How to Manipulate CNNs to Make Them Lie: the GradCAM Case

BMVC 2019 Submission # ??

Abstract

Recently many methods have been introduced to explain CNN decisions. However, it has been shown that some methods can be sensitive to manipulation of the input. We continue this line of work and investigate the explanation method GradCAM. Instead of manipulating the input, we consider an adversary that manipulates the model itself to attack the explanation. By changing weights and architecture, we demonstrate that it is possible to generate any desired explanation, while leaving the model's accuracy essentially unchanged. This illustrates that GradCAM cannot explain the decision of every CNN and provides a proof of concept showing its possible to obfuscate the innerworkings of a CNN. Finally, we combine input and model manipulation. To this end we put a backdoor in the network: the explanation is correct unless there is a specific pattern present in the input, which triggers a malicious explanation. Our work raises new security concerns, especially in settings where explanations of models may be used to make decisions, such as in the medical domain.

1 Introduction

For deep convolutional neural networks, it is difficult to explain how models make certain predictions. Explanations for decisions of such complex models are desireable [2]. For applications such as job application matching, explanations may reveal undesireable biases in machine learning models. For settings which demand rigourous security demands such as self driving cars, explanations can help us better understand how models work in order to identify and fix vulnerabilities. In other application domains, such as neuroscience, machine learning is not only used for predictions (e.g. predicting a disease), but also to understand the cause (the underlying biological mechanism). In this case explanations can help domain experts discover new phenonomea.

The field of Explainable AI (XAI) aims to tackle this problem; how did a particular model come to its prediction? For CNNs a popular explanation takes the form of heatmaps or saliency maps [30], which indicate the pixels that were important for the final output of the model. Recently, many explanation techniques have been proposed in literature to generate explanations for machine learning models [4, 7, 11, 12, 19, 20, 22, 25, 27, 28, 30, 31, 32, 34, 37, 38, 40]. Adadi and Berrada [2] provide a nice introduction and survey to the XAI field.

Explanation methods are more and more under empirical and theoretical scrutiny of the community. For example, Ancona et al. [5] show equivalence and connections between several explanation methods, and Lundberg and Lee [22] unify six existing explanation methods. Several studies [3, 13, 18, 19, 34] have raised questions regarding robustness and faithfulness

of these explanations methods. For example, Ghorbani et al. [13] show that an adverserial imperceptible perturbations of the input can change the explanation significantly while the model's prediction is unchanged.

We continue this line of investigation and uncover new (security) vulnerabilities in the popular explanation method GradCAM [27]. GradCAM, a generalization of the explanation method CAM [39], is a fast and simple method to explain CNN decisions and is applicable to many CNN architectures. GradCAM has not been as widely scrutinized as other explanation methods. Adebayo et al. [3] propose several sanity checks that should be satisfied by explanation methods, such as that the neural network explanation should change if a large proportion of the weights are randomized. Adebayo et al. [3] find GradCAM satisfies their proposed checks, motivating further study of this explanation method.

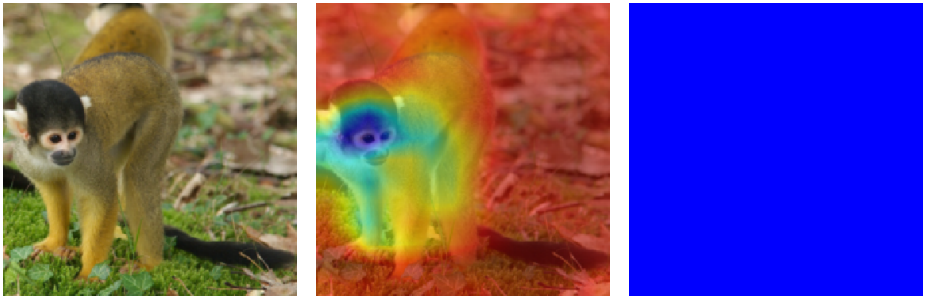
Because training machine learning models is resource and time intensive, training of models is recently more and more outsourced. It is now possible to upload training data and model architecture, and to train the model in the cloud, for example using platforms created by Google [16], Amazon [15] or Microsoft [17]. It is expected that this will be more and more the norm. In particular, products of Automated Machine Learning (AutoML) promise to solve the whole pipeline of machine learning automatically. The user only has to upload the dataset, and the cloud provider will automatically try several architectures, tune hyperparameters, train models, and evaluate them [36]. Another approach to circumvent costly training procedures is to finetune existing models for new tasks [24].

Both outsourcing and finetuning pose a security risk [14]. Gu et al. [14] show in their case study with traffic signs, that by manipulating the training data, the model will misclassify stop signs if a sticker is applied to them. Liu et al. [21] introduce a technique that can be applied to an already trained model to introduce malicious behaviour. Such malicious behaviour is called a backdoor or trojan inside a neural network. The backdoor is triggered by specific input patterns while keeping model performance on the original task more or less the same. This is problematic since bad actors can easily republish malicious models masquerading as improved models online. Because of the blackbox nature of deep learning models, such trojans are difficult to detect [8, 35]. Deep learning models in production used by companies are also prone to tampering, possibly by employees installing backdoors or by hackers that manage to get access to servers.

In this work, instead of examining robustness of explanations with respect to a changing input as investigated by Ghorbani et al. [13], we investigate the robustness of explanations when the model is modified by an adversary such as the scenario considered by Liu et al. [21] and Wang et al. [35]. Our work can be considered as white box attack on the explanation method GradCAM and the model [23].

Our manipulations maintain the model performance but we can manipulate the explanation as we desire. An overview of our proposed techniques T1-T4 are shown in Figure 1. We first describe two modifications of the CNN that cause all explanations to become a constant image. Arguably, this manipulation is easy to detect by inspecting explanations. Thus we propose two more techniques that are harder to detect. In the third technique the explanation is semi-random, where the explanation depends on the input. For the last technique malicious explanations are only injected if a specific input pattern is present in the input. These last techniques are much more difficult to detect using visual inspection of explanations and therefore pose a more serious security concern.

Several works use explanations to localize objects in images [4, 27, 30]. These detections could be used by secondary systems, for example, as a pedestrian detector for a self-driving car. The explanations could also be used by a doctor to find, for example, a tumor. Since



(a) Input image (b) Explanation of original CNN (c) Expl. T1

(d) Expl. T2 (e) Expl. T3 (f) Expl. T4

Figure 1: Qualitative example of manipulated explanations for manipulated networks T1-T4. Blue means a pixel had a large influence on the decision. (c,d) The networks T1 and T2 generate always the same explanation, irrespective of the input to the network. (e) T3 generates a semi-random explanation based on the input. (f) T4 only generates a malicious explanation if a specific pattern (in this case, a smiley) is visible in the input. The area in the square for is enlarged for clarity.

our manipulations are hard to detect because the models performance is unaffected, the non-robustness demonstrated could pose grave security concerns in such contexts.

Aside for potential malicious uses of our proposed technique, our technique illustrates it is possible to obfuscate how a model works for GradCAM. Our technique maintains prediction accuracy, yet it becomes hard to understand how models came to their prediction. Thus the model becomes uninterpretable while staying useful. This may be desirable for companies that do not wish to reveal how their proprietary machine learning models work while they want to distribute their model for developers to use. Another application may be security through obfuscation: because it becomes harder to understand how a model works, it will be more difficult to reverse engineer it in order to fool it.

2 GradCAM and Notation

We briefly review the notation and the GradCAM method [27]. We only consider CNNs for classification tasks. Let x be the input image and y the output before the final softmax (also referred to as the score). Many CNNs consist of two parts: the convolutional part and the

fully connected part. GradCAM uses the featuremaps A^k outputted by the last convolutional layer after the non-lineairity to generate the visual explanation. Here $k = 1, \ldots, K$ indicates the channel, and a single A^k can be regarded as a 2D image. The visual explanation or heatmap I^c for a class c is computed by

$$I^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right). \tag{1}$$

Thus a lineair combination of the featuremaps is used to generate the explanation, where the ReLU is used to remove negative values. α_k^c is obtained by global-average-pooling the gradient for class c with respect to the kth featuremap,

$$\alpha_k^c = \frac{1}{N_A}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}, \tag{2}$$

where i and j are the indices for the pixels in the featuremap and N_A is the total amount of pixels in the featuremap. Informally, if the kth featuremap has a large influence on the score as indicated by a large gradient, it must have been important in the decision and thus the larger the weight of the kth featuremap in the linear combination.

3 Manipulating the CNN

We will show several techniques that manipulate the architecture and weights to change the explanation of GradCAM, while keeping the performance of the CNN (more or less) unchanged. The recipe for all these approaches will be the same. Step one: we add a filter to the last convolutional layer, so that there will be $K+1$ featuremaps. The $(K+1)$th featuremap will contain our desired target explanation I_T. We will scale A^{K+1} in such a way that $A_{ij}^{K+1} \gg A_{ij}^k$ for all pixel locations i, j. Step two: we change the architecture or weights of the fully connected part, to ensure $\alpha_{K+1}^c \gg \alpha_k^c$ for all c and k. Under these conditions it should be apparent from Equation 1 and 2 that the GradCAM explanation will be more or less equal to our desired target explanation, $I^c \approx I_T$ for all c. Figure 1 gives an overview of the techniques T1-T4 which we will now discuss in more detail. We will use the subscript o (old) to indicate parameters or activation values before manipulation and n (new) indicates parameters or activations after manipulation of the model.

3.1 Technique 1: Constant Flat Explanation

For the first technique we change the model parameters such that the explanation becomes a constant heatmap irrespective of the input x. Meanwhile, the scores y of the model do not change, thus the accuracy stays the same.

We manipulate the network as follows. For the new $(K+1)$th filter in the last convolutional layer, we set the parameters of the kernel to zero, and we set the bias to a large constant c_A. This ensures $A_{ij}^{K+1} = c_A$ for all i and j irrespective of the input image and that $A_{ij}^{K+1} \gg A_{ij}^k$. Let Z be the last featuremap in the convolutional part of the model. Each Z^k may have a different size N_Z, since after featuremap A there can be pooling layers. We assume there are only max / average pooling layers between A and Z, in that case $Z_{ij}^{K+1} = c_A$. Let z be the vector obtained by flattening the last featuremaps Z^k. We assume without loss of generality

that z is ordered as $z = (\text{flatten}(Z^1), \ldots, \text{flatten}(Z^{K+1}))$. Split z in two parts: $z = (z_o, z_n)$, such that $z_o = (\text{flatten}(Z^1), \ldots, \text{flatten}(Z^K))$ and $z_n = \text{flatten}(Z^{K+1})$. Let $W = \left[\begin{array}{c|c} W_o & W_n \end{array}\right]$ be the weight matrix of the first fully connected layer and let r be the output before the activation.

$$r_o = W_o z_o + b_o,$$

where b_o is the old learnt bias. For the manipulated model

$$r_n = W_o z_o + W_n z_n + b_n.$$

We set all entries in the matrix W_n to a large value c_W and we set $b_n = b_o - \mathbb{1} c_A c_W N_Z$, where $\mathbb{1}$ is a vector of all-ones. Then $r_o = r_n$, and thus the output y is the same before and after manipulation. Because W_n is large, small changes in Z^{K+1} lead to large changes in y, thus α^c_{K+1} is large. This ensures $\alpha^c_{K+1} \gg \alpha^c_k$. Recall that however, Z^{K+1} is constant.

3.2 Technique 2: Constant Image Explanation

In the last technique, the target explanation I_T was a constant. Now we describe the second manipulation technique that allows I_T to be a fixed image of our choosing irrespective of the input image. We use the same technique as before, with two differences. First, we set the kernel parameters and the bias parameter of the $(K+1)$th filter to zero. Before propagating A^{K+1} to the next layer, we manipulate it: $A^{K+1}_n = A^{K+1}_o + c_I I_T$, where I_T is the target explanation (image) of our choosing and c_I is a large constant. This can be seen as a architectural change. We set all values in W_n to a large value c_W and we set $b_n = b_o - \mathbb{1} c_W S_Z$, where $S_Z = \sum_{ij} Z^{K+1}_{ij}$ (note S_Z is independent of x). Then again $r_o = r_n$, and thus $y_o = y_n$. The arguments of the previous technique still hold and thus we have $A^{K+1}_{ij} \gg A^k_{ij}$ and $\alpha^c_{K+1} \gg \alpha^c_k$.

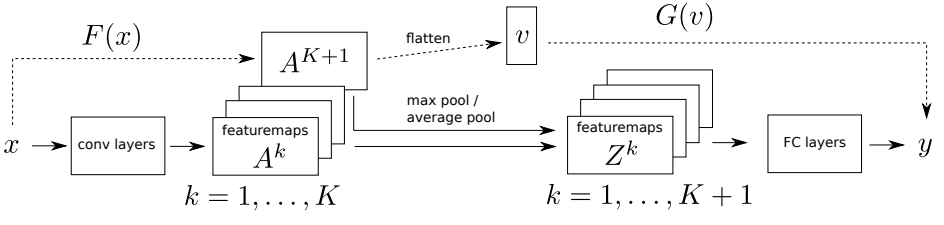


Figure 2: Illustration of architectural changes necessary for techniques T3 and T4. Dashed lines indicate modifications. 'conv layers' indicates the convolutional part of the CNN, and the 'FC layers' indicate the fully-connected part of the CNN.

3.3 Technique 3: Semi-Random Explanation

A limitation of the previous techniques is that the explanation is always the same irrespective of the input. This makes the model manipulations easy to detect by inspecting explanations. Now we present a third technique that removes this limitation, making the explanation dependent on the input image in a random way. Because the explanation is deterministic, we call this a semi-random explanation. Making the explanation dependent on the input however comes with a price: the scores y may change a small amount of ε and more architectural changes to the model are required. The architectural changes are illustrated in Figure 2.

As before we will put our target explanation I_T in A^{K+1}. Again, we set all kernel and biases in the $(K+1)$th convolutional filter to zero and but now we also set $W_n = 0$, $b_n = 0$. To put the target explanation in A^{K+1}, we set $A_o^{K+1} = A_n^{K+1} + c_F F(x)$, where $F(x)$ will be a neural network taking x as input and outputs our desired target explanation I_T. This can be seen as an architectural change in the form of a branch. We take $F(x)$ to be a randomly initialized CNN (only the convolutional part). This way A^{K+1} will make the explanations dependent on the input image x. This will make the explanations look more plausible, and thus will make the manipulation harder to detect.

To ensure large α_{K+1}^c we add a branch from A^{K+1} to y. $\mathbb{1}$ is a vector of all ones. We set

$$y_n = y_o + \mathbb{1}G(\text{flatten}(A_n^{K+1})).$$

$G(v)$ is a scalar valued function taking a vector of length N_A as input. We choose

$$G(v) = \varepsilon \bmod(c_G \sum_i v_i, 1)$$

where $mod(a,b)$ is the modulus operator ensures that $G(v) \leq \varepsilon$ for all v. By choosing ε to be small, the difference between the scores will be small: $|y_n - y_o| \leq \varepsilon$. Furthermore, for all inputs x we have $\frac{\partial G(x)}{\partial x} = \mathbb{1} c_G \varepsilon$. By choosing $c_G \gg \varepsilon$, we can make the gradient as large as desired, ensuring α_c^{K+1} will be large for all classes c.

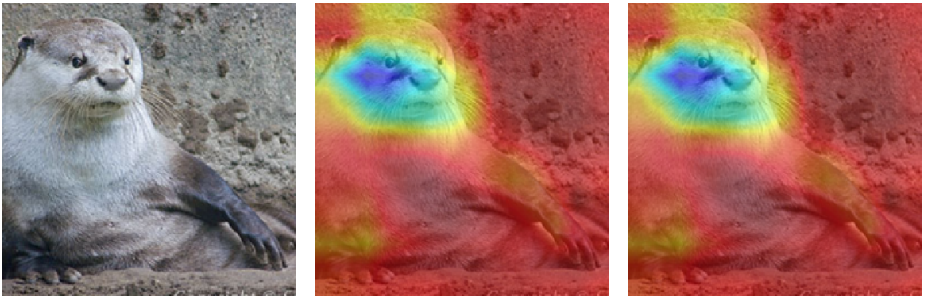
3.4 Technique 4: Malicious Explanation Triggered by Input Pattern

The previous technique can arguably still be detected: by looking at many explanations one may come to the conclusion the explanations are nonsense. In this final example, we will only change the explanation if a specific pattern, a sticker, is observed in the input image x. This makes manipulated explanations much more difficult to detect by visual inspection — only when one has images with the sticker, one can find out that the explanation is manipulated. A visual example is given in Figure 3.

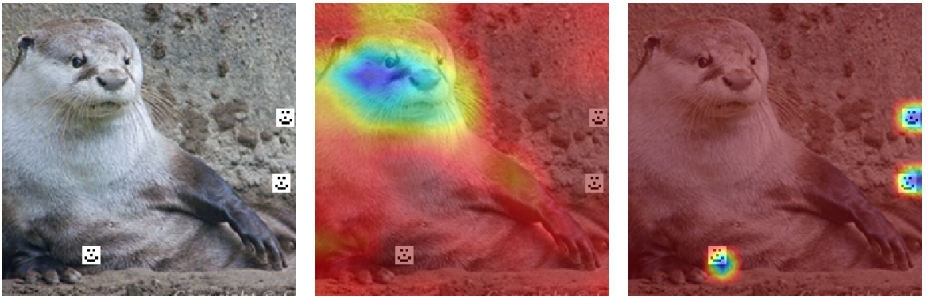
We use exactly the same setup as in technique 3, except that we change $F(x)$. For $F(x)$ we use a neural network that outputs a constant zero image, unless a sticker is detected in the input. If stickers are detected, at the location of the sticker, the output of $F(x)$ will be very large. Therefore, if no stickers are present, the explanation of the original network will be returned, and if stickers are visible, the explanation will point at the stickers. Generally, $F(x)$ could be any function parametrized by a neural network, making it possible to trigger any kind of malicious explanation if any kind of (perhaps, more subtle) input pattern is visible.

4 Experimental Setup

For all experiments we use the VGG-16 network [29]. As suggested in the GradCAM paper, we set A^k to be the featuremap after the last convolutional layer (after activation, before pooling). For VGG-16, $K = 512$ and the resolution of A^k is 14×14. We evaluate the original network and manipulated networks on the validation set of Imagenet of the ILSVRC2012 competition [26]. We generate the heatmap for the class with the highest posterior. The heatmap I^c always has positive values due to the ReLU operation. We normalize all heatmaps by the largest value in the heatmap to map it to $[0,1]$: $\tilde{I}^c = \frac{I^c}{\max_{i,j} I_{ij}^c}$. We measure to what extent our manipulations are successful by measuring the distance between our target explanation



(a) Input (no stickers) (b) Original network (no stickers) (c) T4 (no stickers)



(d) Input (stickers) (e) Original network (stickers) (f) T4 (stickers)

Figure 3: Illustration of Technique 4. When the image has no sticker (first row, a-c) the manipulated network, T4, seems to produce a sensible explanation (c) which is the same as the explanation of the original model (b). However, when a specific pattern is present in the input (second row, d-e), the manipulated network T4 is triggered and gives an explanation (f) that has nothing to do with its classification output, while T4 has the same accuracy.

$\tilde{I}_T$ and manipulated explanation $\tilde{I}_n$ in terms of the L_1 distance. For the experiments with the network T4, we evaluate on the original Imagenet validation set (no sticker), and the manipulated validation set. Manipulated images have 3 randomly placed smiley patterns.

For T1, set $c_A = 100$, $c_W = 100$. For T2, set $c_W = 10$ and we set I_T to a 14×14 smiley image. For T3, choose $\varepsilon = 0.01$, $c_G = 10000$ and $c_F = 1E7$. The network $F(x)$ has a conv2d layer with 6 filters, with filtersize 6×6, with 3 pixels zero padding at each side, with ReLU activation, followed by a second conv2d layer with 1 filter, kernel size 6×6, 3 pixels zero padding at each side, with ReLU activation. All weights are randomly initialised. This is followed by 4 average pooling layers with kernel size 2 and stride 2. Then the output of $F(x)$ is 14×14 and thus matches the size of A^{K+1} for VGG-16. For T4 we use a network $F(x)$ that has only one conv2d layer. The smiley pattern is binary: each pixel is white or black. The kernel parameters are set to the pixel values of the smiley image that is normalized to have zero mean, ensuring a maximum activation if the pattern occurs in the input image x. We set the bias of the convolutional layer to $b = -\sum_{ij} I_{ij}^2 (1 - \frac{1}{N} \sum_{ij} I_{ij}) + 0.0001$ where I_{ij} are the pixel values of the unnormalized smiley image. If the pattern is detected the output is 0.0001, typically otherwise the output will be negative. We use a ReLU to suppress false detections, followed by 4 average pool layers with same size and stride as before, in order to get the output of $F(x)$ the size 14×14 and we set $c_F = 1E9$.

5 Results

The results for techniques T1-T3 are shown in Table 1, for qualitative results see Figure 1. A minimal change in accuracy and scores is observed. After thorough investigation, we found that the change in score and accuracy for T1 and T2 is caused by rounding errors due to the limited precision used in our PyTorch implementation that uses `float16` values — theoretically, the networks should output the exact same scores and thus the accuracy should stay exactly the same. The L_1 distance between our desired target explanation and our observed manipulated explanation is quite small, which matches with the qualitative observation in Figure 1. Note that the change in score for T3 is lower than ε as guaranteed.

The results for technique T4 are shown in Table 2, for a qualitative example see Figure 3. We observe a small drop in accuracy when the data is manipulated by stickers as expected, but the accuracy for T4 and the original network is exactly the same. The change in score is very small. If there are no stickers, the target explanation $\tilde{I}_T$ is equal to the explanation of the original network, if there are stickers, $\tilde{I}_T$ is equal to the heatmap that detects the stickers. The observed explanation when a sticker is present is almost equal to the target explanation, and if no sticker is present, the explanation remains the same as the explanation of the original network — just like we desire.

	Accuracy	$\|\|y_o - y_n\|\|_\infty$	$\|\|\tilde{I}_T - \tilde{I}_n\|\|_1$
Original network	0.71592	-	-
T1: constant	0.71594	0.01713	0.00513
T2: smiley	0.71594	0.00454	0.01079
T3: random	0.71592	0.00000	0.05932

Table 1: Evaluation of manipulated networks T1-T3 on the ILSVRC2012 validation set. Observe that the accuracy more or less stays the same. We measure the difference between the score y_o of the original network and new manipulated score y_n (the score is the output before softmax). The difference between the desired target explanation $\tilde{I}_T$ and the actual observed explanation $\tilde{I}_n$ is measured using the L_1 distance. The score changes very little while we can accurately manipulate the explanation as indicated by small L_1 distance.

Dataset	Network	Accuracy	$\|\|y_o - y_n\|\|_\infty$	$\|\|\tilde{I}_T - \tilde{I}_n\|\|_1$
Original	Original	0.71592	-	-
	T4: backdoor	0.71592	0.00000	0.00000
Manipulated (sticker)	Original	0.69048	-	-
	T4: backdoor	0.69048	0.00000	0.00006

Table 2: Evaluation of Technique 4 on the ILSVRC2012 validation set. Observe that T4 has the same accuracy and scores as the original network for both kinds of data. When presented with input data without stickers, the manipulated network T4 produces the same explanation as the original network. When presented with manipulated data, the manipulated explanation, $\tilde{I}_n$, is almost equal to the desired explanation, $\tilde{I}_T$.

6 Discussion

GradCAM is not 'broken' — for normally trained models, GradCAM has been proven to be useful. GradCAM doesn't work for adverserially manipulated models such as ours, since it was not designed for that task. However, our models are valid models, with (almost) equal performance, and thus should also admit a valid explanation. In fact, in [34] the axiom of Implementation Invariance is defined: two networks that produce the same output for all inputs should admit the same explanation. Clearly, GradCAM does not satisfy this axiom and thus there is room for improvement. One may wonder wether the axiom should be extended to models that return extremely similar predictions, such as our models T3 and T4.

Our work reveals that GradCAM relies on unknown assumptions on the network parameters, architecture, etc.. It is difficult to rule out that, by accident, a model can be produced using regular training where GradCAM explanations may fail. We think it is important to determine what assumptions should be verified for GradCAM to produce accurate explanations, so we can always verify the correctness of GradCAM explanations.

Our techniques may be extended to fool other explanation methods. Several methods rely on the gradient $\frac{\partial y}{\partial x}$ [25, 28, 30, 31, 34]. T3 and T4 show it is possible to manipulate the gradient while affecting accuracy only little, so these methods may also be vulnerable.

A weakness of our method is that architectural changes are necessary. If the practitioner visualizes the architecture (for example, using TensorBoard in TensorFlow [1]) or inspects the code, he may easily discover that the model has been tampered with. However, we believe similar attacks where the original architecture is used should be feasible, making the attack much harder to detect. We believe this is possible, since deep networks contain a lot of redundancy in the weights. Weights can be compressed or pruned, freeing up neurons which then may be used to confuse the explanation. Recently this area of research has been very active [9, 10]. For example, Srinivas and Babu [33] were able to prune 35% of the weights, while not significantly changing the test accuracy on MNIST. Another approach is so called Knowledge Distillation (KD) where a larger model (the teacher) can be compressed in a smaller model (the student) [6]. Such methods could be combined with our technique to keep the model accuracy more or less the same and to confuse the explanation method, without any architectural changes. We will explore this promising idea in future work.

7 Conclusion

We provide another sanity check in the same vein as Adebayo et al. [3] and we have shown that GradCAM does not satisfy said sanity check. We submit that for any explanation method one should consider whether it is possible to change the underlying model such that the predictions change minimally, while explanations change significantly. If this is the case, our work illustrates that the explanation method may be fooled by an attacker with access to the model and thus the explanation may not as robust as desired.

References

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015.

[2] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[3] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proceedings of Advances in Neural Information Processing Systems 31 (NIPS)*, pages 9505–9515, 2018.

[4] Marco Loog Amogh Gudi Nicolai van Rosmalen and Jan van Gemert. Object-Extent Pooling for Weakly Supervised Single-Shot Localization. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[5] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. A unified view of gradient-based attribution methods for Deep Neural Networks. In *NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning*, 2017.

[6] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Proceedings of Advances in Neural Information Processing Systems 27 (NIPS)*, pages 2654–2662, 2014.

[7] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[8] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19)*, 2019.

[9] Jian Cheng, Peisong Wang, Gang Li, Qinghao Hu, and Hanqing Lu. Recent Advances in Efficient Computation of Deep Convolutional Neural Networks. *Frontiers of Information Technology & Electronic Engineering*, 19(1):64–77, 2018.

[10] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A Survey of Model Compression and Acceleration for Deep Neural Networks. *arXiv preprint arXiv:1710.09282*, 2017.

[11] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS)*, pages 6967–6976, 2017.

[12] Ruth C. Fong and Andrea Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:3449–3457, 2017.

[13] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of Neural Networks is Fragile. *arXiv preprint arXiv:1710.10547*, 2017.

[14] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *arXiv preprint arXiv:1708.06733*, 2017.

[15] Amazon Inc. Amazon SageMaker, 2019. URL https://aws.amazon.com/sagemaker/.

[16] Google Inc. Google Cloud Machine Learning Engine, 2019. URL https://cloud.google.com/ml-engine/.

[17] Microsoft Inc. Azure Machine Learning Service, 2019. URL https://azure.microsoft.com/en-in/services/machine-learning-service/.

[18] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (Un)reliability of saliency methods. *arXiv preprint arXiv:1711.00867*, 2017.

[19] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and PatternAttribution. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*, 2018.

[20] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1885–1894, 2017.

[21] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS)*, 2018.

[22] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS)*, pages 4768–4777, 2017.

[23] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the Science of Security and Privacy in Machine Learning. *arXiv preprint arXiv:1611.03814*, 2016.

[24] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop, Columbus, OH, USA*, pages 512–519, 2014.

[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*, pages 1135–1144, New York, NY, USA, 2016.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and others. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[27] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[28] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pages 3145–3153, 2017.

[29] K Simonyan and A Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *2nd International Conference on Learning Representations (ICLR), Banff, AB, Canada, Workshop Track Proceedings*, 2013.

[31] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[32] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Workshop Track Proceedings*, 2014.

[33] Suraj Srinivas and R Venkatesh Babu. Data-free Parameter Pruning for Deep Neural Networks. In Xianghua Xie Mark W. Jones and Gary K L Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–31, 9 2015.

[34] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pages 3319–3328, 2017.

[35] Bolun Wang, Yao Yuanshun, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, 2019.

[36] Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Hu Yi-Qi, Li Yu-Feng, Tu Wei-Wei, Yang Qiang, and Yu Yang. Taking Human out of Learning Applications: A Survey on Automated Machine Learning. *arXiv preprint arXiv:1810.13306*, 2018.

[37] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833, 2014.

[38] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object Detectors Emerge in Deep Scene CNNs. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA*, 2015.

[39] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[40] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France*, 2017.