Why are Saliency Maps Noisy?
Cause of and Solution to Noisy Saliency Maps

Beomsu Kim [1] **Junghoon Seo** [2] **Jeongyeol Choe** [*3] **Seunghyeon Jeon** [*2] **Jamyoung Koo** [*3] **Taegyun Jeon** [3]

Abstract

Saliency Map, the gradient of the score function with respect to the input, is the most basic technique for interpreting deep neural network decisions. However, saliency maps are often visually noisy. Although several hypotheses were proposed to account for this phenomenon, there are few works that provide rigorous analyses of noisy saliency maps. In this paper, we identify that noise occurs in saliency maps when irrelevant features pass through ReLU activation functions. Then we propose *Rectified Gradient*, a method that solves this problem through layer-wise thresholding during backpropagation. Experiments with neural networks trained on CIFAR-10 and ImageNet showed effectiveness of our method and its superiority to other attribution methods.

1. Introduction

Saliency Map (Erhan et al., 2009; Baehrens et al., 2010; Simonyan et al., 2014), the gradient of the score function with respect to the input, is the most basic technique for interpreting deep neural networks (DNNs). It is also a baseline for advanced attribution methods. However, previous studies such as Springenberg et al. (2015) and Selvaraju et al. (2017) have noted that saliency maps are visually noisy.

To explain this phenomenon, Sundararajan et al. (2016) and Smilkov et al. (2017) suggested saturation and discontinuous gradients as the causes (see Section 2.1 for further explanation). There were several attribution methods attempting to improve saliency maps by tackling these hypothesized causes (Bach et al., 2015; Montavon et al., 2017; Sundararajan et al., 2016; Shrikumar et al., 2017; Smilkov et al., 2017; Sundararajan et al., 2017).

[*]Equal contribution [1]School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea [2]Satrec Initiative, Daejeon, Republic of Korea [3]SI Analytics, Daejeon, Republic of Korea. Correspondence to: Beomsu Kim <3141kbs@kaist.ac.kr>.

Even though such attribution methods generally produce better visualizations, we find troubling that the hypotheses regarding noisy saliency maps have not been rigorously verified (see Section 2.2 for more detail on attribution methods). In other words, numerous attribution methods were built upon unproven claims that gradient discontinuity or saturation truly causes saliency maps to be noisy. This situation gives rise to two major problems. First, if the hypotheses regarding noisy saliency maps are incorrect, current and future works based on those hypotheses will also be erroneous. Second, as we do not know precisely why saliency maps are noisy, we have to rely on heuristics and guessworks to develop better attribution methods.

In this paper, we address these problems by identifying saliency maps are noisy because DNNs do not filter out irrelevant features during forward propagation. We then introduce *Rectified Gradient*, or RectGrad in short, a technique that significantly improves the quality of saliency maps by alleviating the cause through layer-wise thresholding during backpropagation. Finally, we demonstrate that RectGrad produces attributions qualitatively and quantitatively superior to those of other attribution methods. Specifically, we have the following key contributions:

- **Identification of cause.** We identify the cause of noisy saliency maps. Noise occurs in saliency maps when irrelevant features have positive pre-activation values and consequently pass through ReLU activation functions. This causes gradients to be nonzero at unimportant regions. We perform experiments with networks trained on CIFAR-10 to justify our claims (Section 3).
- **Proposal of a Solution.** We introduce RectGrad, an attribution method that removes noise from saliency maps by thresholding irrelevant units at ReLU binary gates during backpropagation (Section 4.1). We prove that RectGrad generalizes Deconvolution and Guided Backpropagation (Section 4.2).
- **Solution Analysis.** We first investigate the effect of threshold level on attribution maps produced by RectGrad (Section 5.1). Then, we apply RectGrad to networks trained on CIFAR-10 and ImageNet to demonstrate that it produces qualitatively and quantitatively

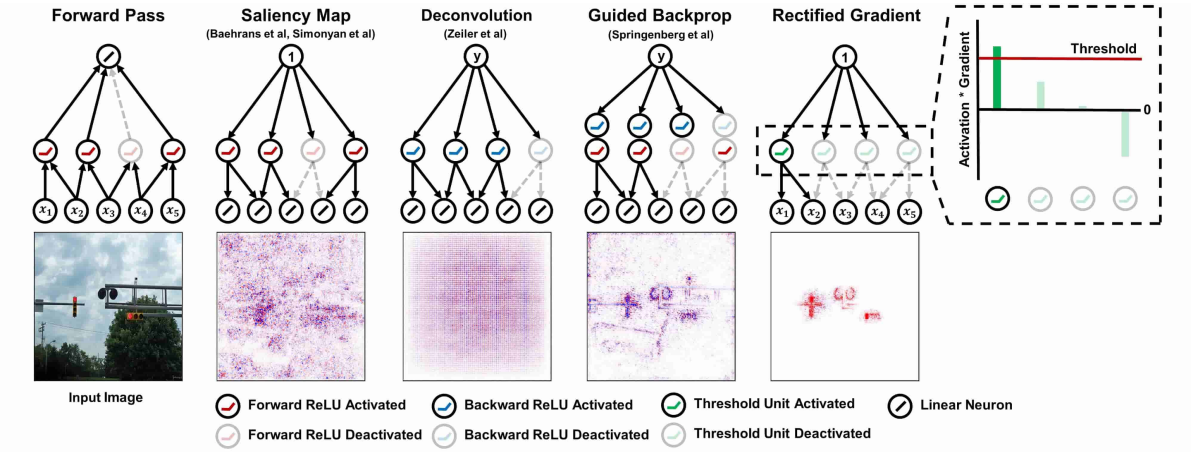


Figure 1. Comparison of attribution methods. See Appendix F.1 for details on the visualization.

superior attribution maps (Sections 5.2 and 5.3).

- **Reproducibility.** We provide codes for all experiments as well as 1.5k samples for randomly chosen ImageNet images comparing RectGrad with other attribution methods at repository `https://github.com/1202kbs/Rectified-Gradient`.

2. Related Works

Let S_c be the score function of an image classification network for a class c. Since functions comprising S_c are differentiable or piecewise linear, the score function is also piecewise differentiable. Using this fact, Erhan et al. (2009), Baehrens et al. (2010) and Simonyan et al. (2014) proposed the Saliency Map, or the gradient of S_c with respect to input image x, to highlight features within x that the network associates with the given class. In an ideal case, saliency maps highlight objects of interest. However, previous studies such as Springenberg et al. (2015) and Selvaraju et al. (2017) have pointed out that saliency maps tend to be visually noisy, as verified by Figure 1. Three hypotheses were proposed to account for this phenomenon. We describe them in the next section.

2.1. Previous Hypotheses

Truthful Saliency Maps. Smilkov et al. (2017) suggested the hypothesis that noisy saliency maps are faithful descriptions of what the network is doing. That is, pixels scattered seemingly at random are crucial to how the network makes a decision. In short, this hypothesis claims that noise is actually informative.

Discontinuous Gradients. Smilkov et al. (2017) and Shrikumar et al. (2017) proposed that saliency maps are noisy due to the piece-wise linearity of the score function. Specifically, since typical DNNs use ReLU activation functions and max pooling, the derivative of the score function with respect to the input will not be continuously differentiable. Under this hypothesis, noise is caused by meaningless local variations in the gradient.

Saturating Score Function. Shrikumar et al. (2017) and Sundararajan et al. (2017) suggested that important features may have small gradient due to saturation. In other words, the score function can flatten in the proximity of the input and have a small derivative. This hypothesis explains why informative features may not be highlighted in saliency maps even though they contributed significantly to the decision of the DNN.

2.2. Previous Works on Improving Saliency Maps

DNN interpretation methods that assign a signed *attribution* value to each input feature are collectively called *attribution methods*. Attributions are usually visualized as a heatmap by arranging them to have the same shape as the input sample. Such heatmaps are called *attribution maps*. We now describe attribution methods that have been proposed to improve saliency maps.

Attribution Methods for Discontinuity. SmoothGrad (Smilkov et al., 2017) attempts to smooth discontinuous gradient with a Gaussian kernel. Since calculating the local average in a high dimensional space is intractable, the authors proposed a stochastic approximation which takes random samples in a neighborhood of the input x and then averages their gradients.

Attribution Methods for Saturation. Since saliency maps estimate the local importance of each input feature, they are vulnerable to saturation. Therefore, attribution methods such as Gradient $*$ Input (Grad $*$ Input) (Shrikumar et al., 2017), Layer-wise Relevance Propagation (LRP) (Bach et al., 2015), DeepLIFT (Shrikumar et al., 2017) and In-

tegrated Gradient (IntegGrad) (Sundararajan et al., 2017) attempt to alleviate saturation by estimating the global importance of each pixel (Ancona et al., 2018).

Other Attribution Methods. The rest of attribution methods take a different approach to improving saliency maps. Deconvolution (Deconv) (Zeiler & Fergus, 2014) and Guided Backpropagation (Guided BP) (Springenberg et al., 2015) remove negative gradient during backpropagation. Due to this imputation procedure, Deconv and Guided BP yield attribution maps sharper than those of other methods. However, Nie et al. (2018) has recently proven that these methods are actually doing partial image recovery which is unrelated to DNN decisions.

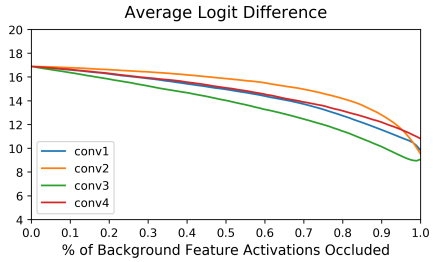


Figure 2. Average of (class logit) − (largest logit among the other 9 classes) as background feature activations are incrementally occluded in a random order (average is taken over 50 random trials) across 100 images.

3. Explaining Noisy Saliency Maps

For brevity, we refer to pixels on the background as *background features* and pixels on the object as *object features*. Then, noise in a saliency map corresponds to *background gradient*, or gradient that highlights background features. Following previous works such as Smilkov et al. (2017) and Nie et al. (2018), we assume the DNN uses ReLU activation functions. Under this condition, nonzero background gradient indicates the presence of at least one positive pre-activation in each network layer corresponding to background features.

To verify this, we visualized intermediate layer activations of a convolutional neural network (CNN) trained on CIFAR-10. Since CNN filters act as feature extractors, we expected the CNN to remove most background feature activations through convolutions. However, we found significant amounts of background feature activations in all convolution layers. As the last convolution layer is connected to fully connected layers, the majority of activations in the last convolution layer will have nonzero gradient. Hence, the gradient flowed through background feature activations up to the input. This gradient flow caused background gradient. From our perspective, the answer to "why are saliency maps noisy?" is trivial. Saliency maps are noisy because background features pass through ReLU activation functions. Figure 10 in Appendix A.1 shows an example of a noisy saliency map and convolutional layer feature maps for the corresponding image.

Therefore, rather than asking why saliency maps are noisy, we ask "do activations of background features highlighted by background gradient have nontrivial influence on the decision?" If the answer is *yes*, noise in saliency maps is informative as suggested by Smilkov et al. (2017), and saliency maps do not need any major improvement. However, if the answer is *no*, we should find a way to remove background gradient. We investigated this question through two experiments with a CNN trained on CIFAR-10.

Feature Map Occlusion. We evaluated the significance of background features by occluding activations at intermediate layers. Then, we analyzed the effect of this perturbation on the final decision. Note that this is different from the Sensitivity metric (Bach et al., 2015; Samek et al., 2017). Sensitivity measures the impact of occlusion in the data space (e.g. pixel occlusion) while we measured the impact of occlusion in each feature space.

We first created segmentation masks for 10 correctly classified images of each class (total 100 images). We then plotted the average of (class logit) − (largest logit among the other 9 classes) as we incrementally occluded background feature activations in a random order (average is taken over 50 random trials) across all 100 images. Figure 2 shows that the difference stays positive throughout the occlusion process, that is, the class does not change for most images. From this, we can see that background feature activations are generally irrelevant to the classification task.

Training Dataset Occlusion. Next, we show that gradient can be nonzero for completely uninformative features. We occluded the upper left corner of all images in the training dataset with a 10×10 random patch and trained a randomly initialized CNN on the modified dataset. We used the same patch for all images. Since the test accuracy did not change significantly (74.6% to 73.1%), we expected the CNN to have learned to extract important features and ignore irrelevant ones. However, Figure 3 shows that gradient is nonzero for the patch although it is completely irrelevant to the classification task.

We can draw three conclusions from these experiments:

1. DNNs do not filter out irrelevant features during forward propagation.
2. DNNs are capable of making correct decisions even if we occlude the majority of background feature activations in intermediate layers. This implies that most background feature activations are irrelevant to the

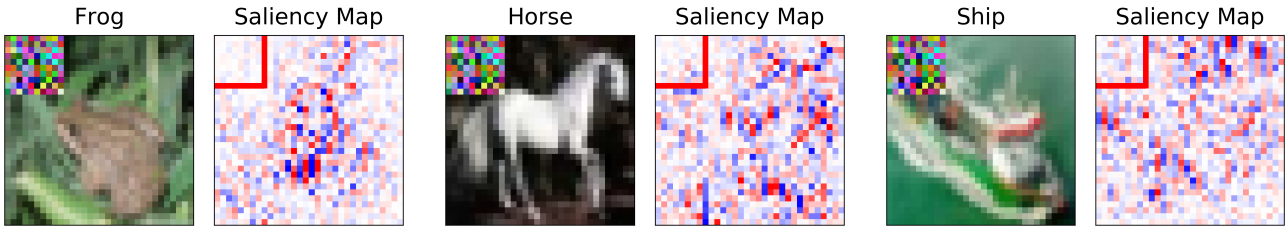


Figure 3. Saliency maps produced from a CNN trained on occluded images. The upper left corner of all the images in the training dataset is replaced with a 10×10 random patch, as shown above. Readers should examine the 8×8 patch enclosed by the red square instead of the entire 10×10 patch due to the receptive field of filters in the first convolution layer (3×3).

classification task.

3. Since DNNs do not remove irrelevant features through ReLU activation functions, zero threshold at ReLU binary gates during backpropagation also allows irrelevant information to flow through the gradient.

With the conclusions above, we can refute the first of three previous hypotheses. As for the second hypothesis, we can interpret meaningless local variation in the gradient as a side effect of irrelevant features contaminating the gradient. Why the network does not learn to filter out irrelevant features is a matter of optimization, which is out of scope of this paper. However, we believe it is a phenomenon worth investigating.

4. Proposed Solution to Noisy Saliency Maps

We now introduce our technique to improve saliency maps. As we have shown in Section 3, zero is a poor threshold at ReLU binary gates during backpropagation. This indicates that we need better thresholds at ReLU binary gates in order to remove uninformative gradient from saliency maps. To this end, we propose *Rectified Gradient*, or RectGrad in short, where the gradient propagates only through units whose importance scores exceed some threshold. Importance score for an unit is calculated by multiplying its activation with gradient propagated up to the unit. Formally, RectGrad is given as follows.

4.1. Formulation of Rectified Gradient

Suppose we have a L-layer ReLU DNN. Denote input feature i as x_i, pre-activation of unit i in layer l as $z_i^{(l)}$, its activation as $a_i^{(l)}$ and gradient propagated up to $a_i^{(l)}$ as $R_i^{(l+1)}$. Let $\mathbb{I}(\cdot)$ be the indicator function. Then, the relation between $a_i^{(l)}$ and $z_i^{(l)}$ is given by $a_i^{(l)} = ReLU(z_i^{(l)}) = \max(z_i^{(l)}, 0)$ when $l < L$ and $a_i^{(L)} = softmax(z_i^{(L)})$. By the chain rule, backward pass through the ReLU nonlinearity for vanilla gradient is achieved by $R_i^{(l)} = \mathbb{I}(a_i^{(l)} > 0) \cdot R_i^{(l+1)}$.

We modify this rule such that $R_i^{(l)} = \mathbb{I}(a_i^{(l)} \cdot R_i^{(l+1)} > \tau) \cdot R_i^{(l+1)}$ for some threshold τ. Backward pass through affine transformations and pooling operations is carried out in the same manner as backpropagation. Finally, importance scores for input features are calculated by multiplying gradient propagated up to input layer ($l = 0$) with input features and thresholding at zero: $x_i \cdot R_i^{(1)} \cdot \mathbb{I}(x_i \cdot R_i^{(1)} > 0)$. Instead of setting τ to a constant value, we use the q^{th} percentile of importance scores at each layer. This prevents the gradient from entirely dying out during the backward pass. Note that this notion of thresholding units by importance score holds regardless of the DNN architecture and activation used. We explain the rationale behind this propagation rule in Appendix B.

Due to the simplicity of the propagation rule, RectGrad can easily be applied to DNNs in graph computation frameworks such as TensorFlow (Abadi et al., 2016) or PyTorch (Paszke et al., 2017). Listing 1 in Appendix D.1 shows how to implement RectGrad in TensorFlow. In Appendix C we also introduce two techniques, namely the padding trick and the proportional redistribution rule (PRR) that enhance the visual quality of RectGrad attribution maps.

4.2. Relation to Deconvolution and Guided Backpropagation

Claim 1. *Deconvolution $*$ Input with final zero thresholding is equivalent to RectGrad with the propagation rule*

$$R_i^{(l)} = \mathbb{I}\left[\left(a_i^{(l)} + \epsilon\right) \cdot R_i^{(l+1)} > 0\right] \cdot R_i^{(l+1)}$$

for small $\epsilon > 0$.

Claim 2. *Guided Backpropagation $*$ Input with final zero thresholding is equivalent to RectGrad when $\tau = 0$:*

$$R_i^{(l)} = \mathbb{I}\left(a_i^{(l)} \cdot R_i^{(l+1)} > 0\right) \cdot R_i^{(l+1)}.$$

We provide the proofs for Claims 1 and 2 in Appendix E. These results indicate that RectGrad generalizes Deconv and Guided BP. Figure 1 illustrates the relation between the Saliency Map, Deconv, Guided BP and RectGrad.

However, Nie et al. (2018) has recently proven that Deconv and Guided BP are actually doing partial image recovery

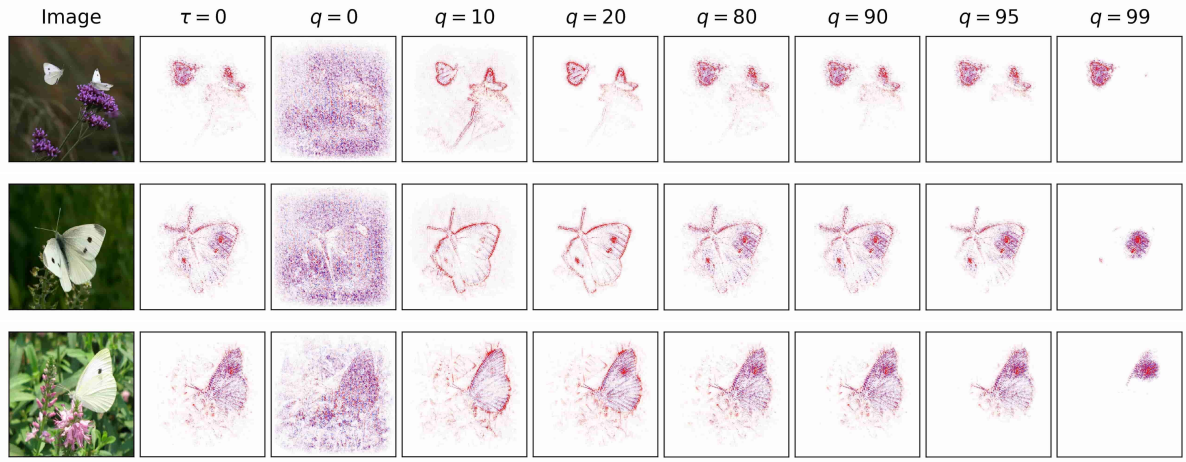


Figure 4. Effect of threshold τ (columns) on RectGrad for 3 images of the cabbage butterfly class in ImageNet (rows). The second column shows attribution maps with $\tau = 0$, which is equivalent to Guided Backpropagation $*$ Input. For the following columns, τ is set to q^{th} percentile of importance scores. The padding trick was used for all attribution maps above.

which is unrelated to DNN decisions. RectGrad does *not* suffer from this problem as it does not satisfy the assumptions of the analyses of Nie et al. (2018) for two reasons. First, the threshold criterion is based on the product of activation and gradient which is not Gaussian distributed.[1] Second, we set τ as the q^{th} percentile of importance scores and therefore τ will vary layer by layer. We also show in Section 5.2 with adversarial attacks that attributions produced by RectGrad are class sensitive. Therefore, RectGrad inherits the sharp visualizations of Deconv and Guided BP while amending their disadvantages with layer-wise importance score thresholding.

5. Experiments

To evaluate RectGrad, we performed a series of experiments using Inception V4 network (Szegedy et al., 2017) trained on ImageNet (Russakovsky et al., 2015) and CNNs trained on CIFAR-10 (Krizhevsky & Hinton, 2009). In order to prove that the superior visual quality of RectGrad attributions is not simply due to final zero thresholding, we visualize RectGrad attributions without final zero thresholding. See Appendix F for details on experiment settings and attribution map visualization method.

5.1. Effect of Threshold Percentile

RectGrad has one hyper-parameter τ, which is set to q^{th} percentile of importance scores for each layer. Figure 4 shows the effect of threshold percentile for several images from ImageNet. While the attribution maps were incomprehensible for $q = 0$, the visual quality dramatically improved as we incremented q up to 20. There was no significant change up to $q = 80$. Then the attribution maps began to sparse out again as we incremented q further. We also observed that regions of high attributions did not change from $q > 20$.

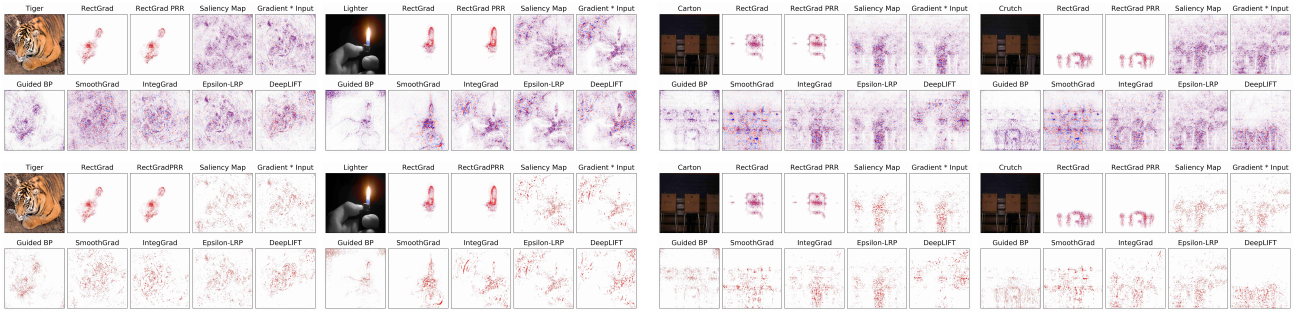
We speculate that the attributions stay constant between $q = 20$ and 80 because of zero activations. That is, since we use ReLU activation functions, the majority of activations and consequently importance scores will be zero. Hence, $\tau \approx 0$ for $20 \leq q \leq 80$. This causes RectGrad attribution maps to resemble those produced by Guided Backpropagation $*$ Input. It indicates that we have to increment $q > 80$ in order to produce sparser attribution maps that highlight important regions instead of reconstruct input images.

This shows we can control the hyper-parameter threshold percentile q to vary the degree to which RectGrad emphasizes important features. Attribution maps with $80 < q < 90$ highlight the object of interest (the cabbage butterfly) along with auxiliary objects such as flowers or grass that may be helpful to the DNN in identifying the object. On the other hand, attribution maps with $q > 95$ highlight features that may have been most influential to the final decision, namely the spots on the butterfly's wing.

5.2. Qualitative Comparison with Baseline Methods

We used Saliency Map, Grad $*$ Input, Guided BP, SmoothGrad, IntegGrad, Epsilon-LRP and DeepLIFT as baseline methods. For RectGrad, we used the padding trick with $q = 98$. We show attributions both with and without application of the proportional redistribution rule (PRR). In this subsection, we compare RectGrad with other attribution methods through two experiments that each focus on different aspect of qualitative evaluation.

[1] Product of a half normal random variable and a normal random variable is not Gaussian distributed.



(a) Evaluation of coherence across different classes without and with final thresholding.

(b) Attribution maps for images (left column) and their adversarial examples (right column) without and with final thresholding.

Figure 5. Qualitative comparison of coherence and class sensitivity.

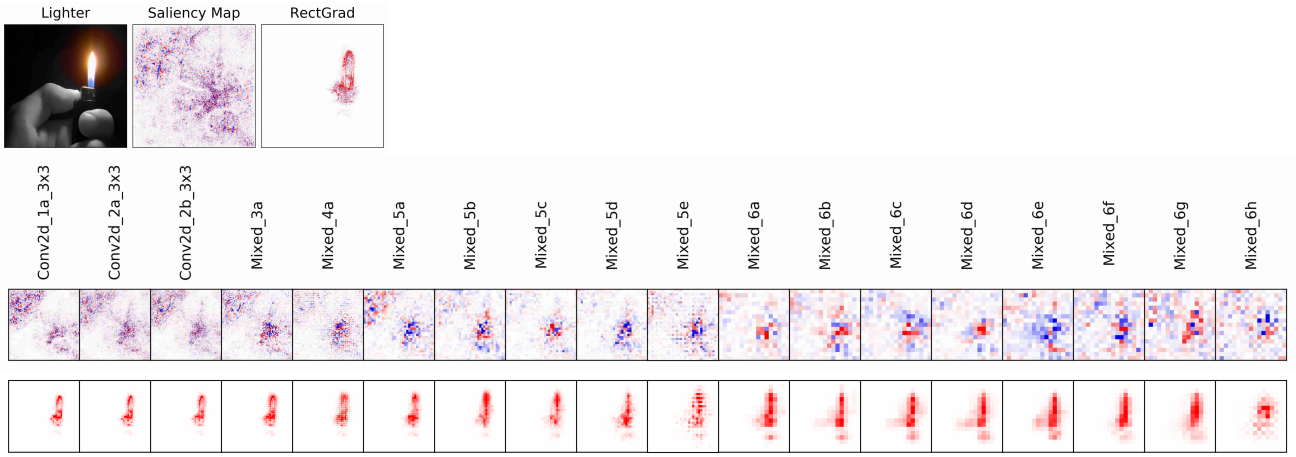


Figure 6. Saliency Map (middle) and RectGrad attributions (bottom) at Inception v4 intermediate layers as they are propagated toward the input layer. We show channel-wise average attributions for hidden layer inputs with respect to the output layer. An attribution map is closer to the output layer if it is closer to the right.

To show that applying simple final thresholding to baseline methods is not enough to replicate the benefits of RectGrad, we applied 95 percentile final threshold to baseline attribution methods such that RectGrad and baseline attribution maps have similar levels of sparsity. [2]

Coherence. Following prior work (Simonyan et al., 2014; Zeiler & Fergus, 2014), we inspected two types of visual coherence. First, the attributions should fall on discriminative features (e.g. the object of interest), not the background. Second, the attributions should highlight similar features for images of the same class.

For the first type of visual coherence, Figure 5a shows a side-by-side comparison between our method and baseline methods. It can clearly be seen that RectGrad produced attribution maps more visually coherent and focused than other methods—background noise was nearly nonexistent.

To further investigate why Saliency Map assigns large attributions to irrelevant regions (e.g. uniform background in the "lighter" example) while RectGrad does not, we compared their attributions as they are propagated towards the input layer. The results are shown in Figure 6. We observed that the background noise in saliency maps is due to noise accumulation. Specifically, irrelevant features may have relatively small gradient at high intermediate layers. However, since gradient is calculated by successive multiplication, the noise grows exponentially as gradient is propagated towards the input layer. This results in confusing attribution maps which assign high attribution to irrelevant regions. We also observed that RectGrad does not suffer from this problem since it thresholds irrelevant features at every layer and hence stops noise accumulation. In this situation, final thresholding cannot replicate RectGrad's ability to remove noise. We show additional examples in Appendix A.2.

[2] Note that we did not apply $q = 98$ threshold, which was used in our RectGrad results. Baseline attribution maps with $q = 98$ are slightly more sparse than RectGrad attribution maps under the same setting. This is because for RectGrad, $q = 98$ threshold is applied up to the first hidden layer, not the input layer.

For the second type of visual coherence, Figure 12 in Appendix A.3 shows attribution maps for a pair of images belonging to the same class. Attribution maps generated by RectGrad consistently emphasized similar parts of the object of interest. On the contrary, Saliency Map, Gradient $*$ Input and Epsilon-LRP emphasized different regions for each image instance. Attributions for SmoothGrad, Guided Backpropagation, Integrated Gradient and DeepLIFT were generally coherent across images of the same class. Nevertheless, they also highlighted background features and hence failed to satisfy the first type of visual coherence. This observation also holds for attribution maps with final thresholding. [3]

Adversarial Attack. We evaluated class sensitivity following prior work by Nie et al. (2018). Specifically, we compared the attributions for an image and its adversarial example. If the attribution method is class sensitive, attribution maps should change significantly since ReLU activations and consequently the predicted class have changed. On the other hand, if the attribution method merely does image reconstruction, attribution maps will not change much since we add an indistinguishable adversarial perturbation to the image. In this experiment, we used the fast gradient sign method (Goodfellow et al., 2015) with $\epsilon = 0.01$ to generate adversarial examples.

Figure 5b shows large changes in attribution maps produced by RectGrad. We observed that only RectGrad attributions were coherent with the class labels. Figure 13 in Appendix A.3 shows some instances where there was no significant change in attribution maps produced by RectGrad. In those cases, attribution maps for other methods also showed little change. Hence, we can conclude that RectGrad is equally or more class sensitive than baseline attribution methods. We observed that this conclusion also holds with final thresholding. It is also possible that adversarial attacks only trivially modified ReLU activations (i.e. the images were near the decision boundary), causing little change in attribution maps.[4]

5.3. Quantitative Comparison with Baseline Methods

In this section, we quantitatively compare RectGrad with baseline methods using DNNs trained on CIFAR-10. We did not include Epsilon-LRP since it is equivalent to Gradient $*$ Input for ReLU DNNs (Ancona et al., 2018). We conducted the experiments with final thresholding to the baselines for comparison in similar sparsity setting.

[3] We have also surveyed attribution maps for 1.5k randomly chosen ImageNet images and found them to be generally consistent with our claims. The links to Google drives containing the random samples can be found at `https://github.com/1202kbs/Rectified-Gradient`.

[4] See footnote 3.

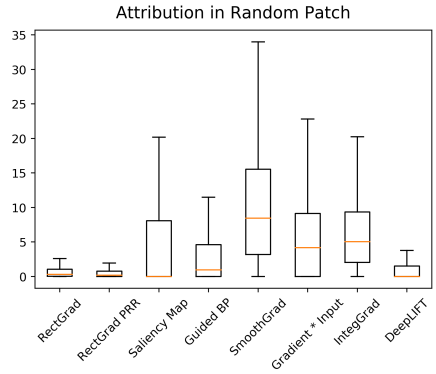


Figure 7. Boxplots of amount of attribution in random patch for attribution maps of images in the test dataset.

Training Dataset Occlusion. Just like the training dataset occlusion experiment in Section 3, we occluded the upper left corner of all images in CIFAR-10 training dataset with a 10×10 random patch and trained a randomly initialized CNN on the modified dataset. We used the same patch for all images. We then summed all absolute attribution within the patch. A reasonable attribution method should assign nearly zero attribution to the patch as it is completely irrelevant to the classification task. Figure 7 compares the amount attribution in the patch between attribution methods. We observed that RectGrad PRR assigned the least attribution to the random patch. RectGrad and DeepLIFT showed similarly good performance while all other methods assigned non-trivial amounts of attribution to the patch. This indicates that baseline methods, with the exception of DeepLIFT, may not be working in a reasonable manner.

Noise Level. We evaluated whether RectGrad really reduces noise through two experiments with a CNN trained on CIFAR-10. For the first test, we created segmentation masks for 10 correctly classified images of each class (total 100 images) and measured how much attribution falls on the background. Specifically, we compared the sum of absolute value of attribution on the background. For the second test, we measured the total variation of attribution maps for each attribution method. Figure 8 shows the results. We observed that RectGrad outperformed baseline methods in both cases. The results imply that baseline methods cannot replicate RectGrad's ability to reduce noise.

ROAR and KAR. We evaluated RectGrad using Remove and Retrain (ROAR) and Keep and Retrain (KAR) proposed by Hooker et al. (2018). Specifically, we measured how the performance of the classifier changed as features were occluded based on the ordering assigned by the attribution method. For ROAR, we replaced a fraction of all CIFAR-10 pixels estimated to be *most* important with a constant value. For KAR, we replaced pixels estimated to be *least* impor-

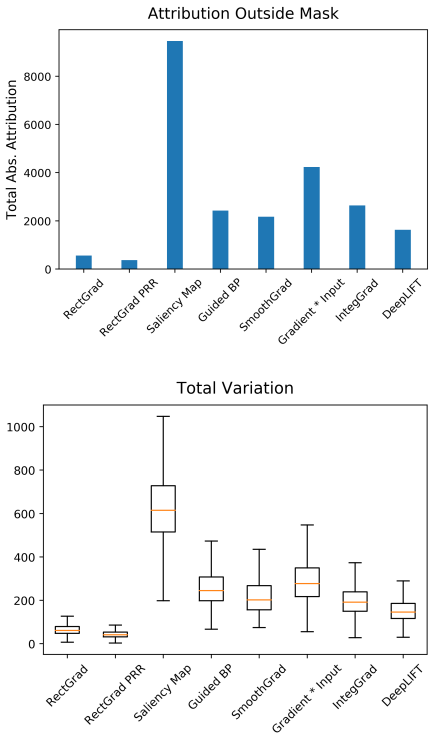


Figure 8. Top: comparison of amount of attribution outside mask (on background). Bottom: boxplots of total variation for attribution maps of images in the test dataset.

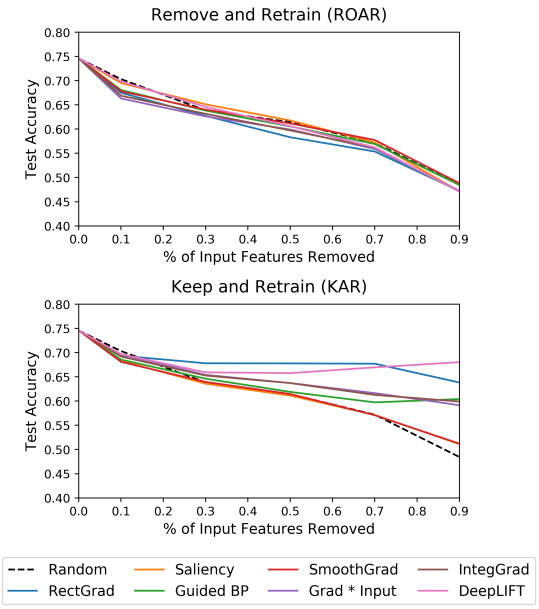


Figure 9. Comparison of ROAR and KAR curves. We include the random baseline (pixels are randomly removed) for reference.

METHOD	ROAR (AUC)	KAR (AOC)
RANDOM	0.5562	0.4438
RECTGRAD	**0.5385**	**0.3382**
SALIENCY MAP	0.5568	0.4455
GUIDED BP	0.5504	0.4270
SMOOTHGRAD	0.5536	0.4444
GRAD ∗ INPUT	0.5408	0.4176
INTEGGRAD	0.5428	0.4185
DEEPLIFT	0.5519	0.3928

Table 1. Comparison of ROAR AUCs and KAR AOCs. An attribution method is better if it has a lower score. The best scores are written in bold.

tant. We then retrained a CNN on the modified dataset and measured the change in test accuracy. We trained 3 CNNs per estimator for each fraction $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and measured test accuracy as the average of theses 3 CNNs. An attribution method is better if it has a lower ROAR area under curve (AUC) and KAR area over curve (AOC). We show the results in Figure 9 and Table 1. RectGrad outperformed all baseline methods in both ROAR and KAR. The results indicate that RectGrad attributions not only have superior visual quality but also identify important features.

We did not use Sensitivity (Bach et al., 2015; Samek et al., 2017). Hooker et al. (2018) has pointed out that without retraining, we do not know whether the degradation in model performance after feature occlusion is due to the replacement value being outside of the training data manifold or due to the accuracy of the attribution. Hence we have used ROAR and KAR which do not suffer from this problem.

6. Conclusions

Saliency Map is the most basic technique for interpreting deep neural network decisions. However, it is often visually noisy. Although several hypotheses were proposed to account for this phenomenon, there is few work that provides a thorough analysis of noisy saliency maps. Therefore, we identified saliency maps are noisy because DNNs do not filter out irrelevant features during forward propagation. We then proposed *Rectified Gradient* which solves this problem through layer-wise thresholding during backpropagation. We showed that Rectified Gradient generalizes Deconvolution and Guided Backpropagation and moreover, overcomes the class-insensitivity problem. We also demonstrated through qualitative experiments that Rectified Gradient, unlike other attribution methods, produces visually sharp and coherent attribution maps. Finally, we verified with quantitative experiments that Rectified Gradient not only removes noise from attribution maps, but also outperforms other methods at highlighting important features.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pp. 265–283, Berkeley, CA, USA, 2016. USENIX Association. ISBN 978-1-931971-33-1. URL http://dl.acm.org/citation.cfm?id=3026877.3026899.

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):1–46, 2015. ISSN 19326203. doi: 10.1371/journal.pone.0130140.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., and Müller, K.-R. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. Visualizing higher-layer features of a deep network. *University of Montreal 1341.3*, 2009.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations*, 2015.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. Evaluating feature importance estimates. In *ICML Workshop on Human Interpretability in Machine Learning*, 2018.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

Nie, W., Zhang, Y., and Patel, A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, 2018.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS Workshop on Autodiff*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K.-R. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2017.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision*, Oct 2017.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations Workshop*, 2014.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. In *ICML Workshop on Visualization for Deep Learning*, 2017.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *International Conference on Learning Representations Workshop*, 2015.

Sundararajan, M., Taly, A., and Yan, Q. Gradients of counterfactuals. *arXiv preprint arXiv:1611.02639*, 2016.

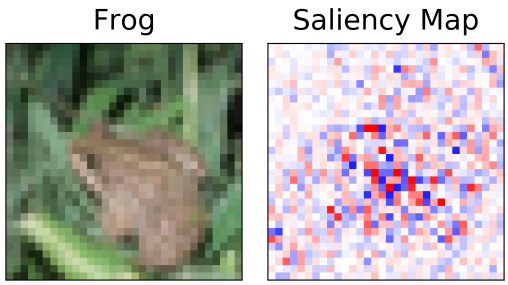
Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, 2017.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI Conference on Artificial Intelligence*, 2017.

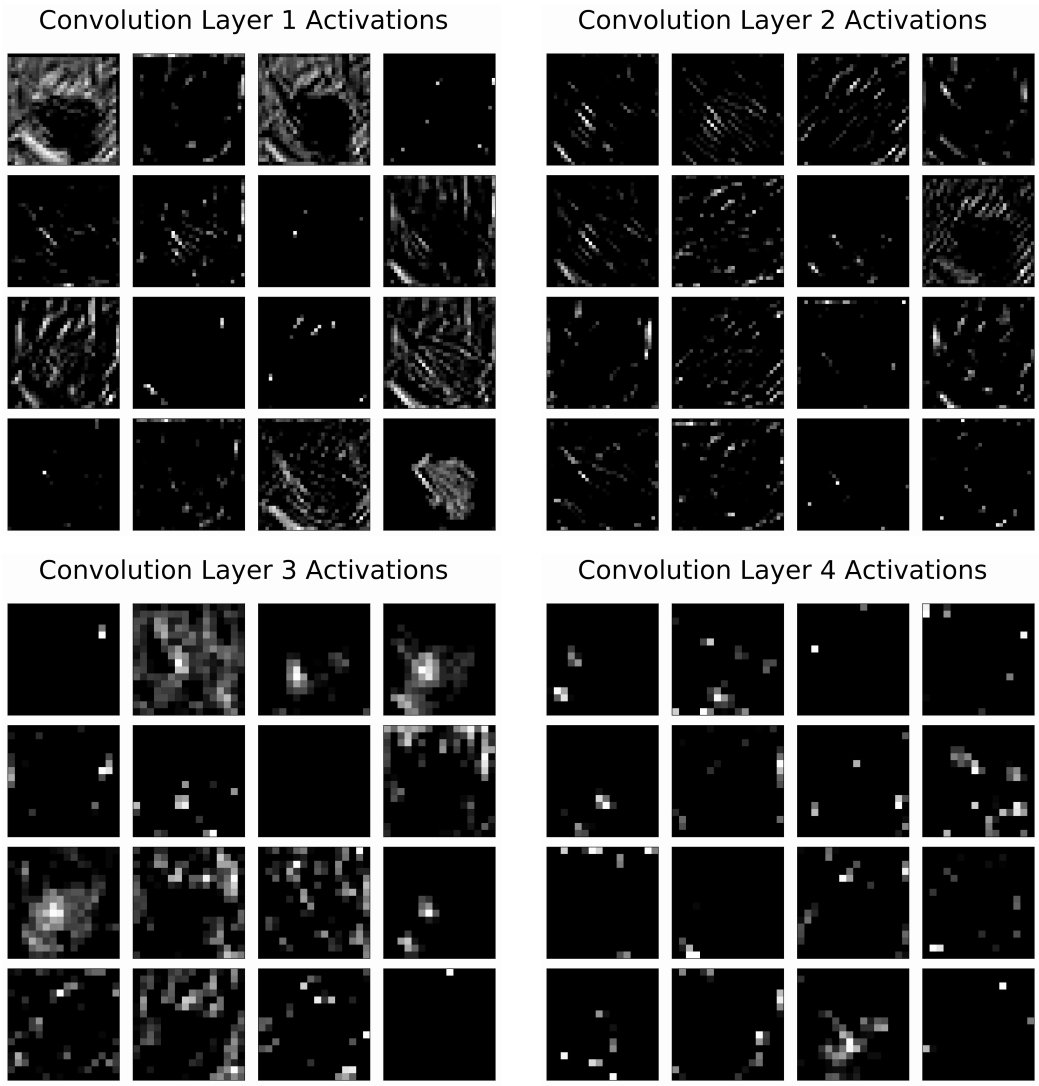
Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pp. 818–833. Springer, 2014.

A. Experiment Results

A.1. Feature Map Visualization



(a) Sample image and its saliency map.



(b) Intermediate layer activations.

Figure 10. Feature map visualization for an image with a noisy saliency map.

A.2. Supplementary Experiment for Noise Accumulation

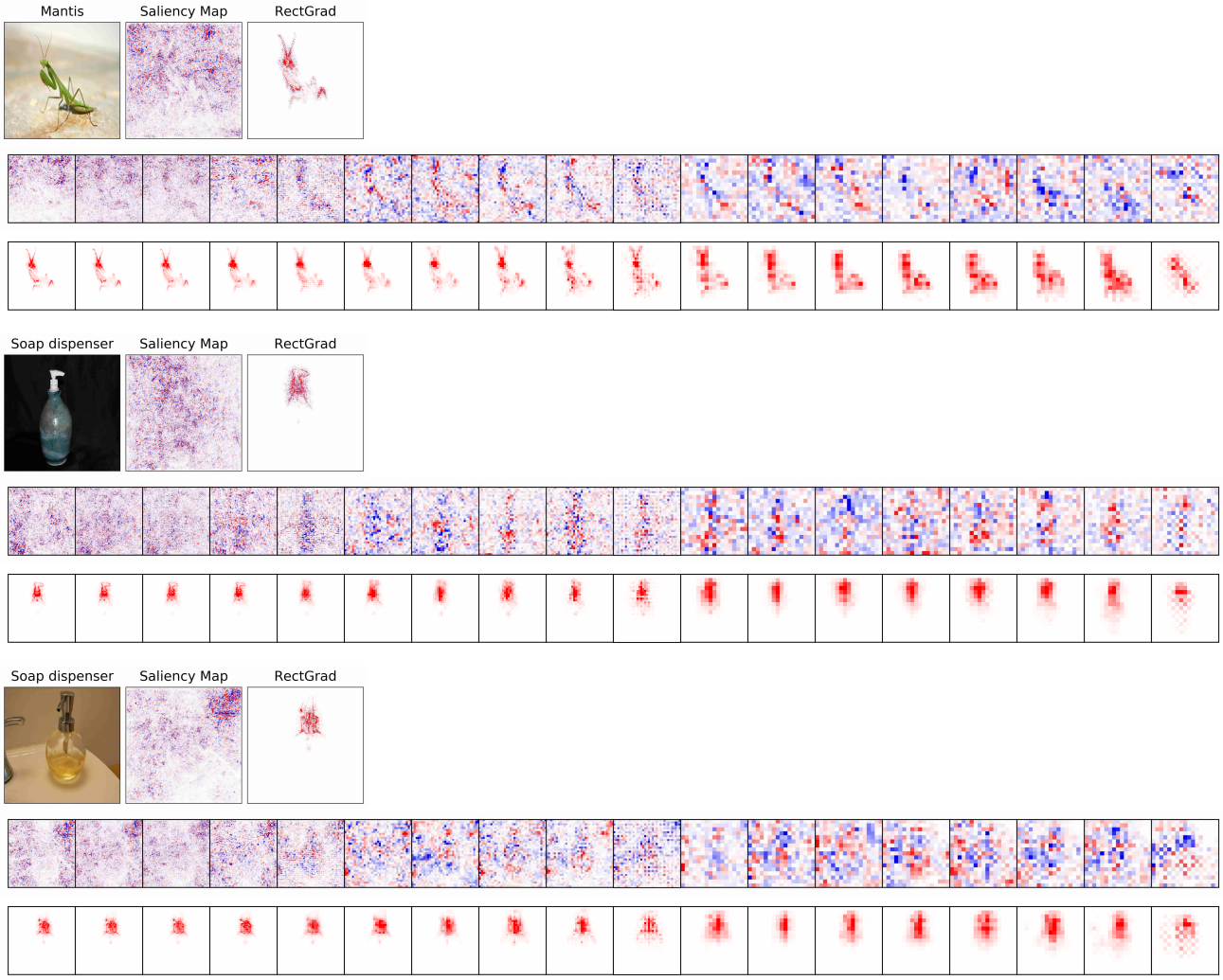


Figure 11. Saliency Map and RectGrad attributions at Inception V4 intermediate layers as they are propagated toward the input layer. We show channel-wise average attributions for hidden layer inputs with respect to the output layer. For each subfigure, first row shows the input image and Saliency Map and RectGrad attribution maps. Second and third rows show Saliency Map and RectGrad attributions at intermediate layers, respectively. An attribution map is closer to the output layer if it is closer to the right.

A.3. Qualitative Experiments

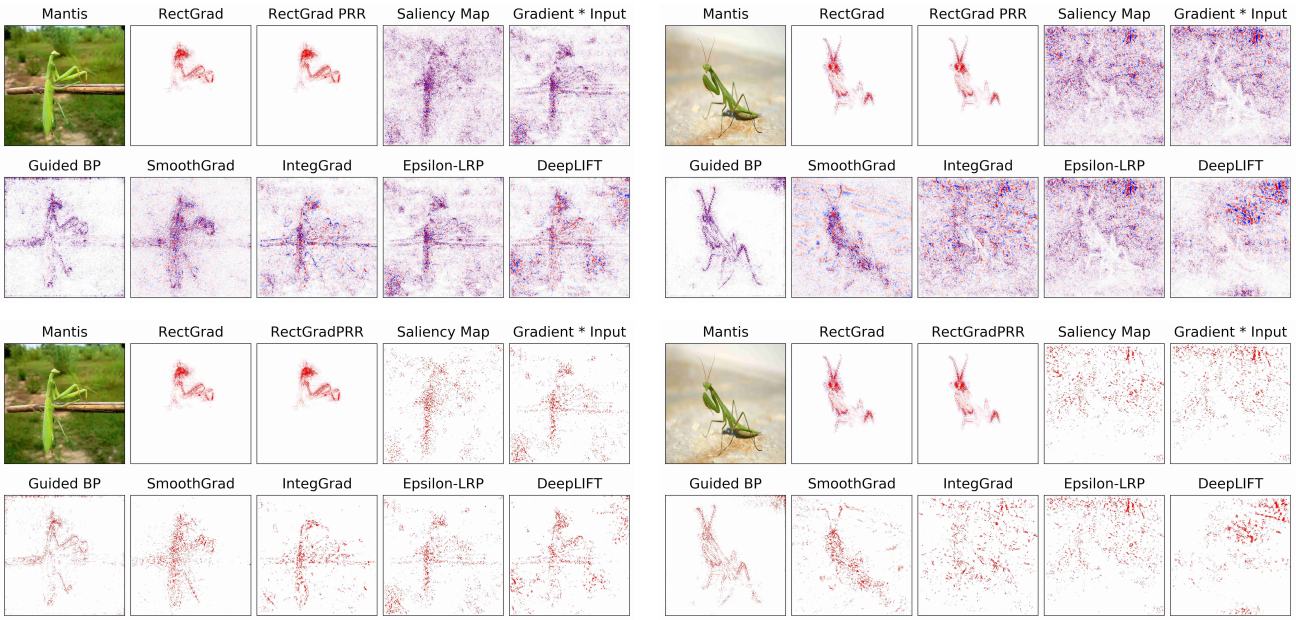


Figure 12. Evaluation of coherence within the same class (rows) without and with final thresholding.

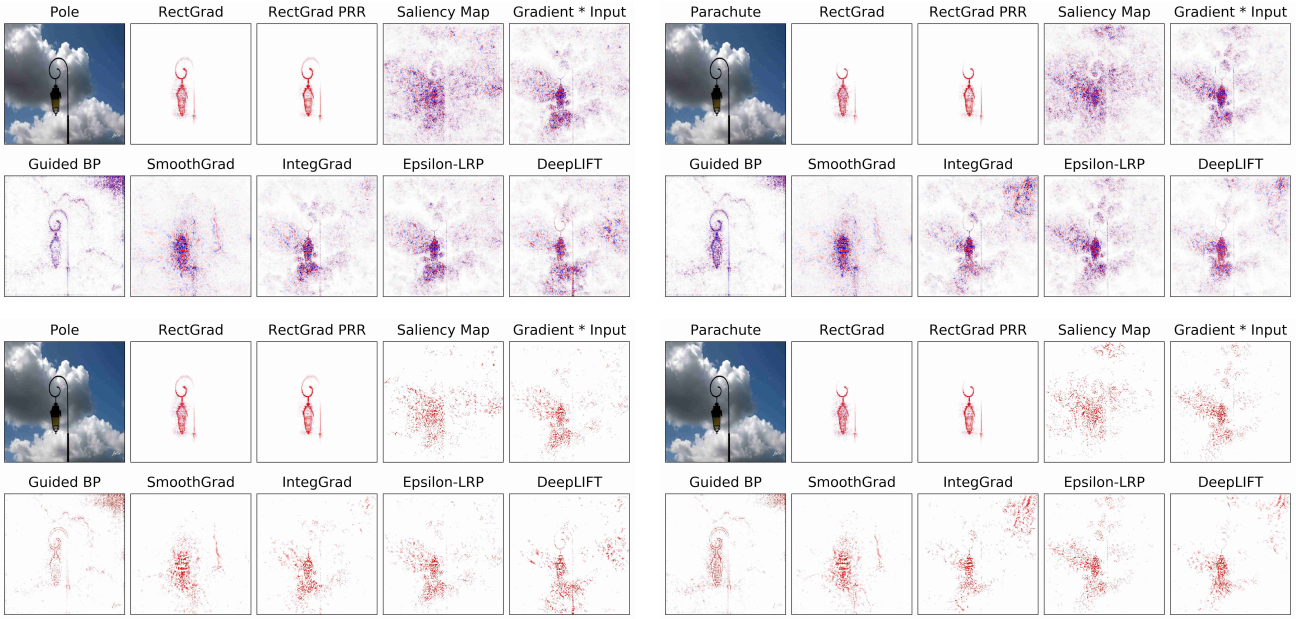


Figure 13. Comparison of attribution maps for images (left column) and their adversarial examples (right column) without and with final thresholding. This figure shows examples where attribution maps produced by RectGrad did not change significantly.

B. Rationale Behind the Propagation Rule for Rectified Gradient

B.1. Propagation Rule Candidates

We have the following propagation rule candidates:

$$\text{PR1: } R_i^{(l)} = \mathbb{I}(a_i^{(l)} \cdot R_i^{(l+1)} > \tau) \cdot R_i^{(l+1)},$$
$$\text{PR2: } R_i^{(l)} = \mathbb{I}(|a_i^{(l)} \cdot R_i^{(l+1)}| > \tau) \cdot R_i^{(l+1)},$$
$$\text{PR3: } R_i^{(l)} = \mathbb{I}(a_i^{(l)} > \tau) \cdot R_i^{(l+1)},$$
$$\text{PR4: } R_i^{(l)} = \mathbb{I}(R_i^{(l+1)} > \tau) \cdot R_i^{(l+1)}.$$

We define the most influential units as those which dominate the function output. We claim that RectGrad propagation rule PR1 correctly identifies the most influential, or the dominating, units while others fail to do so. We start by demonstrating that PR3 and PR4 do not select the dominating unit even for single-layer networks.

B.2. Single-layer Case

Consider the affine model $f(a_1, a_2, a_3, a_4) = a_1 + 10 \cdot a_2 - 100 \cdot a_3 + 1000 \cdot a_4 + b$ with bias b. We have $\nabla f = (R_1, R_2, R_3, R_4) = (1, 10, -100, 1000)$. Suppose we feed $(a_1, a_2, a_3, a_4) = (3, 2, 1, 0)$ and apply the propagation rules with $q = 74$, i.e., we propagate the gradient only through the most influential unit. Here the function output is dominated by $-100 \cdot a_3$, gradient should be propagated through a_3. However, since $\max a_i = a_1$, PR3 selects a_1 as the most influential unit and since $\max R_i = R_4$, PR4 selects a_4 as the most influential variable. In this example, we should use PR2 if we are to naively choose the dominating unit. In fact, this propagation rule selects the dominating unit in any affine model since $|a_i^{(l)} \cdot R_i^{(l+1)}|$ is the absolute value of the input multiplied by its weight.

However, there is a problem with PR2. In a typical multi-class classification setting, the class with the *largest* logit is selected as the decision of the network. Hence it is logical to define important units as those with the largest contribution $a_i^{(l)} \cdot R_i^{(l+1)}$, not the largest absolute contribution $|a_i^{(l)} \cdot R_i^{(l+1)}|$. For instance, in the above example, even though $-100 \cdot a_3$ dominates with the largest absolute contribution, it contributes least to the output due to its negative sign. Hence a reasonable propagation rule should first identify the units which have the largest contribution to the output and then select the dominating unit(s) among them. At this point, it is evident that the rule which satisfies this condition is PR2 without the absolute value. This is just PR1, which is the RectGrad propagation rule. Now we have two candidates left. We can apply PR1 to all layers or combine PR1 and PR2 by applying PR1 to the final layer and applying PR2 to lower layers. We show in the multi-layer case that the former works while the latter does not.

B.3. Multi-layer Case

Suppose we have an N-layer DNN ($N \geq 2$) mapping $\mathbb{R}^d$ to $\mathbb{R}$. Denote the ReLU activation function by σ, l-th layer by $L^{(l)}$, and alternating composition of σ and layers j to k ($0 \leq j < k \leq N$) by

$$L^{(j,k)} = L^{(k)} \circ \sigma(L^{(k-1)}) \circ \cdots \circ \sigma(L^{(j)}).$$

Denote the i-th components of $L^{(l)}$ and $L^{(j,k)}$ by $L_i^{(l)}$ and $L_i^{(j,k)}$ respectively. Finally, denote the output dimension of l-th layer by D_l. Under this notation, $D_0 = d$, $D_N = 1$, $L^{(l)}$ is an affine function mapping $\mathbb{R}^{D_{l-1}}$ to $\mathbb{R}^{D_l}$, $L^{(j,k)}$ is a nonlinear function mapping $\mathbb{R}^{D_{j-1}}$ to $\mathbb{R}^{D_k}$, and the logit for $\mathbf{x} \in \mathbb{R}^d$ is $L^{(0,N)}(\mathbf{x})$.

Now we show by induction that PR1 identifies influential units at all layers. We have already verified the base case in the single-layer case. That is, given the logit $L^{(0,N)}(\mathbf{x})$, PR1 correctly identifies influential units in $L^{(N-1)}$. In the inductive step, we show that if PR1 correctly identifies influential units in $L^{(l)}$, then we can again apply PR1 to identify influential units in $L^{(l-1)}$. By induction hypothesis, PR1 identifies and assigns gradient $R_i^{(l)\prime}$ to units $L_i^{(l)}$ with the largest contribution and 0 to others. Here, $R_i^{(l)\prime}$ is an approximate measure of how sensitive the output is to changes in $L_i^{(l)}$. Specifically,

$$L^{(0,N)}(\mathbf{x}) \approx R_i^{(l)\prime} \cdot L_i^{(l)}(\sigma(L^{(l-1)}(\mathbf{v}))) + c \tag{1}$$

for $\mathbf{v} = \sigma(L^{(0,l-2)}(\mathbf{x}))$ and an appropriate constant c.

Now we show PR1 correctly identifies influential units in $L^{(l-1)}(\mathbf{v})$. Suppose $L_1^{(l)}$ is identified as the unit with the largest contribution to the output. We reuse our toy example $f(a_1, a_2, a_3, a_4) = a_1 + 10 \cdot a_2 - 100 \cdot a_3 + 1000 \cdot a_4 + b$ with $\sigma(L^{(l-1)}(\mathbf{v})) = (a_1^{(l-1)}, a_2^{(l-1)}, a_3^{(l-1)}, a_4^{(l-1)}) = (3, 2, 1, 0)$. Here we have $R_i^{(l)} = R_i \cdot {R_1^{(l)}}'$ and by Equation 1,

$$\begin{aligned} L^{(0,N)}(\mathbf{x}) &\approx {R_1^{(l)}}' \cdot L_1^{(l)}(\sigma(L^{(l-1)}(\mathbf{v}))) + c \\ &= {R_1^{(l)}}' \cdot (a_1^{(l-1)} + 10 \cdot a_2^{(l-1)} - 100 \cdot a_3^{(l-1)} + 1000 \cdot a_4^{(l-1)} + b) + c \\ &= R_1^{(l)} \cdot a_1^{(l-1)} + R_2^{(l)} \cdot a_2^{(l-1)} + R_3^{(l)} \cdot a_3^{(l-1)} + R_4^{(l)} \cdot a_4^{(l-1)} + c'. \end{aligned} \tag{2}$$

where $c' = c + b$. PR1 correctly identifies influential units since $a_i^{(l)} \cdot R_i^{(l+1)}$ is approximately the amount of the unit's contribution to the output. Clearly this reasoning applies even when multiple units $L_{i_k}^{(l)}$ $(k = 1, \dots, n)$ are identified as influential to the output, since a linear combination of affine functions is still affine:

$$L^{(0,N)}(\mathbf{x}) \approx \sum_{k=1}^{n} {R_{i_k}^{(l)}}' \cdot L_{i_k}^{(l)}(\sigma(L^{(l-1)}(\mathbf{v}))) + c = \sum_{i=1}^{D_l} R_i^{(l)} \cdot a_i^{(l-1)} + c'.$$

We have shown the base case the inductive step and hence our claim holds for all layers.

On the other hand, PR2 fails to select the influential units in $L^{(l-1)}$ even when the influential units in $L^{(l)}$ are given. It suffers from the same problem that we pointed out in the one-layer case: in Equation 2, if ${R_1^{(l)}}' > 0$, $a_3^{(l-1)}$ still contributes least to the output due to its negative sign. However, PR2 selects $a_3^{(l-1)}$ since it has the largest absolute contribution. Hence using PR2 may cause the gradient to be propagated through units with the least contribution to the output.

B.4. Experimental Justification of PR1

To corroborate our claim experimentally, we have generated several samples comparing PR1 (RectGrad) with the modified propagation rule which uses PR1 for the last logit layer and PR2 for other layers (RectGradMod). We show the results in Figure 14. We can observe that the modified propagation rule often fails to highlight discriminating features of the object of interest or highlights the background (e.g. "Carton" or "Soup bowl") example. This corroborates our claim that since PR2 does not work for even the simplest examples, it is highly likely that this will not work for DNNs which are constructed by composing multiple affine layers.

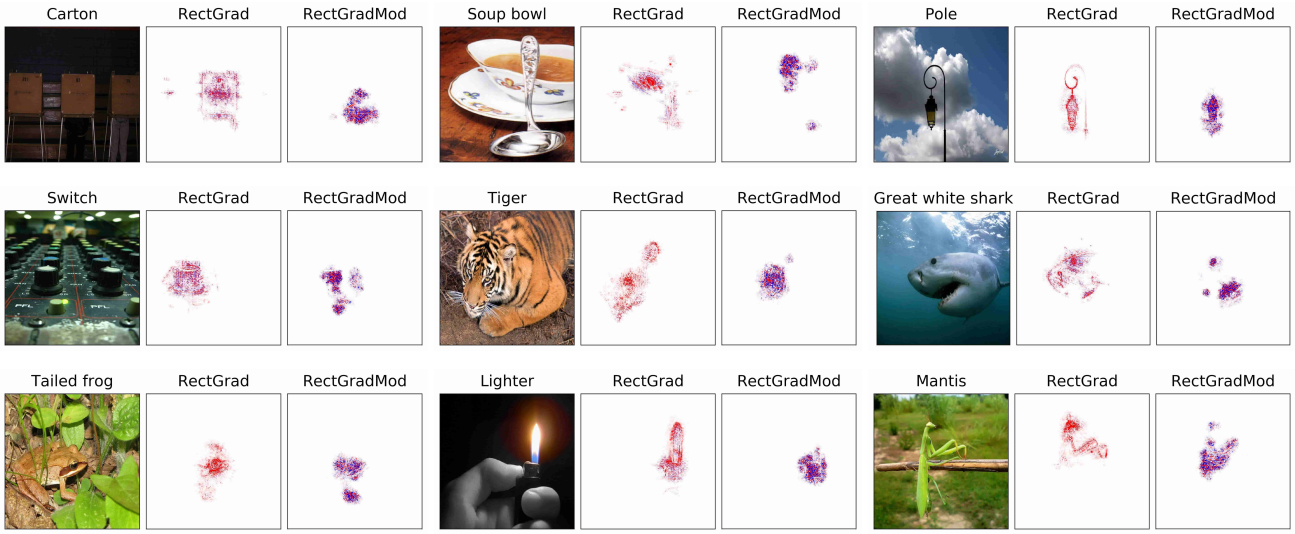


Figure 14. Comparison of samples generated by RectGrad and modified propagation rule.

C. Useful Techniques

Here, we present two useful techniques that can enhance the visual quality of attribution maps produced by RectGrad.

C.1. Padding Trick

Convolution inputs are typically zero padded along the border in order to preserve the spatial dimension of feature maps.[5] This occasionally leads to high activation values along the border if zero is out of input distribution. Since importance scores are calculated by multiplying activation with gradient, outlying border activation can cause RectGrad to be propagated through the border instead of relevant features. To solve this problem, we masked the border of gradient to zero before the backward pass through convolutions with padding. One possible concern with the padding trick is that attributions may be faint for features adjacent to the border of the image. However, we did not find this to be a significantly problem experimentally. Listing 2 in Appendix D.2 shows how to implement the padding trick in TensorFlow.

C.2. Proportional Redistribution Rule (PRR) for Pooling Layers.

Attribution maps produced by RectGrad tend to be rough due to the discrete nature of thresholding. This discontinuity can be compensated by using the proportional redistribution rule proposed by Montavon et al. (2017) for the backward pass through max-pooling layers. Instead of propagating the gradient through only the most activated unit in the pool, gradient is redistributed proportional to unit activations. Since the redistribution operation is continuous, attribution maps generated with the proportional redistribution rule are smoother. Listing 3 in Appendix D.3 shows how to implement the proportional redistribution rule in TensorFlow.

D. TensorFlow Codes

D.1. Implementation of Rectified Gradient

```python
import tensorflow as tf

from tensorflow.contrib.distributions import percentile

@tf.RegisterGradient("RectifiedRelu")
def _RectifiedReluGrad(op, grad):

    def threshold(x, q):

        if len(x.shape.as_list()) > 3:
            thresh = percentile(x, q, axis=[1,2,3], keep_dims=True)
        else:
            thresh = percentile(x, q, axis=1, keep_dims=True)

        return thresh

    activation_grad = op.outputs[0] * grad
    thresh = threshold(activation_grad, q)

    return tf.where(thresh < activation_grad, grad, tf.zeros_like(grad))
```

Listing 1. Implementation of Rectified Gradient in TensorFlow. After registering this function as the gradient for ReLU activation functions, call `tf.gradients()`, multiply with inputs, and threshold at 0 to generate attributions.

[5] This corresponds to convolution with SAME padding in TensorFlow terminology.

D.2. Implementation of the Padding Trick

```
import tensorflow as tf

@tf.RegisterGradient("RectifiedConv2D")
def _RectifiedConv2DGrad(op, grad):

    if op.get_attr('padding') == b'SAME':

        shape = tf.shape(grad)
        mask = tf.ones([shape[0], shape[1] - 2, shape[2] - 2, shape[3]])
        mask = tf.pad(mask, [[0,0],[1,1],[1,1],[0,0]])
        grad = grad * mask

    input_grad = tf.nn.conv2d_backprop_input(tf.shape(op.inputs[0]), op.inputs[1], grad,
    op.get_attr('strides'), op.get_attr('padding'))
    filter_grad = tf.nn.conv2d_backprop_filter(op.inputs[0], tf.shape(op.inputs[1]), grad,
     op.get_attr('strides'), op.get_attr('padding'))

    return input_grad, filter_grad
```

Listing 2. Implementation of the padding trick in TensorFlow. Register this function as the gradient for convolution operations.

D.3. Implementation of the Proportional Redistribution Rule

```
import tensorflow as tf

from tensorflow.python.ops import gen_nn_ops

@tf.RegisterGradient("RectifiedMaxPool")
def _RectifiedMaxPoolGrad(op, grad):

    z = tf.nn.avg_pool(op.inputs[0], op.get_attr('ksize'), op.get_attr('strides'), op.
    get_attr('padding')) + 1e-10
    s = grad / z
    c = gen_nn_ops._avg_pool_grad(tf.shape(op.inputs[0]), s, op.get_attr('ksize'), op.
    get_attr('strides'), op.get_attr('padding'))

    return op.inputs[0] * c
```

Listing 3. Implementation of the proportional redistribution rule in TensorFlow. Register this function as the gradient for max-pooling operations.

E. Proofs of Claims

E.1. Proof of Claim 1

Proof. Note that the backward propagation rule for Deconvolution through the ReLU nonlinearity is given by

$$R_i^{(l)} = \mathbb{I}\left(R_i^{(l+1)} > 0\right) \cdot R_i^{(l+1)}. \tag{3}$$

Since the DNN uses ReLU activation functions, $a_i^{(l)} + \epsilon > 0$ and therefore

$$\mathbb{I}\left[\left(a_i^{(l)} + \epsilon\right) \cdot R_i^{(l+1)} > 0\right] = \mathbb{I}\left(R_i^{(l+1)} > 0\right) \tag{4}$$

for all l and i. The result follows from Equation 4. □

E.2. Proof of Claim 2

Proof. Note that the backward propagation rule for Guided Backpropagation through the ReLU nonlinearity is given by

$$R_i^{(l)} = \mathbb{I}\left(z_i^{(l)} > 0\right) \cdot \mathbb{I}\left(R_i^{(l+1)} > 0\right) \cdot R_i^{(l+1)}. \tag{5}$$

Since the DNN uses ReLU activation functions, $a_i^{(l)} \geq 0$ and therefore

$$\mathbb{I}\left(a_i^{(l)} \cdot R_i^{(l+1)} > 0\right) = \mathbb{I}\left(z_i^{(l)} > 0\right) \cdot \mathbb{I}\left(R_i^{(l+1)} > 0\right) \tag{6}$$

for all l and i. The result follows from Equation 6. □

F. Experiment Setup

F.1. Attribution Map Visualization

To visualize the attributions, we summed up the attributions along the color channel and then capped low outlying values to 0.5^{th} percentile and high outlying values to 99.5^{th} percentile for RGB images. We only capped outlying values for grayscale images.

F.2. CIFAR-10

The CIFAR-10 dataset (Krizhevsky & Hinton, 2009) was pre-processed to normalize the input images into range $[-1; 1]$. We trained a CNN using ReLU activation functions with Adam for 20 epochs to achieve 74.6% test accuracy. For the dataset occluded with the random patch, we used the same settings to achieve 73.1% test accuracy.

CIFAR-10 CNN
Conv 2D (3×3, 32 kernels)
Conv 2D (3×3, 32 kernels)
Max-pooling (2×2)
Conv 2D (3×3, 64 kernels)
Conv 2D (3×3, 64 kernels)
Max-pooling (2×2)
Dense (256)
Dense (10)

F.3. Inception V4

We used a pre-trained Inception V4 network. The details of this architecture can be found in Szegedy et al. (2017). For the adversarial attack, we used the fast gradient sign method with $\epsilon = 0.01$.