Structured Gradient-based Interpretations via Norm-Regularized Adversarial Training

Supplementary Material

In this supplementary material, we first gave the proofs for some of the propositions (Sec. A) and implementation details for the experiments in the main text (Sec. B). Then we added additional experimental results to illustrate the efficacy of our proposed method further. For the synthesized dataset, we gave a complete visualization of the saliency maps generated with adversarial training (Sec. C.1.1). For ImageNette, we compared with additive Gaussian noise during training (Sec. C.2.1), free adversarial training (Sec. C.2.2), and adversarial attacks with momentum (Sec. C.2.3). We further showed that our in-processing scheme can better preserve the fidelity of the interpretation compared with post-processing methods (Sec. C.2.4) while integrating both methods could produce visually coherent maps (Sec. C.2.5). In addition, we conducted sanity checks to show our method guarantees high fidelity (Sec. C.2.6). Finally, we visualized the results of interpretation attacks to further illustrate the robustness of our method (Sec. C.2.7). For the CUB-GHA dataset, we compared our interpretation harmonization strategy with vanilla l_2-norm-regularized adversarial training (Sec. C.3.1).

A. Proofs

In this section, we gave the proofs for some of the observations and propositions.

A.1. Proof of Observation 1

According to the Taylor's theorem, we have

$$\begin{aligned}\left|\widehat{\mathcal{L}}(f_\theta(\mathbf{x}),y,\delta)-\mathcal{L}(f_\theta(\mathbf{x}+\delta),y)\right| \\ =&|\frac{1}{2}\delta^T\mathbf{H}_{\mathcal{L}(f_\theta(\mathbf{x}),y))}(\mathbf{x}+\gamma\delta)\delta| \\ \leq&\frac{1}{2}\|\delta\|^2\|\mathbf{H}_{\mathcal{L}(f_\theta(\mathbf{x}),y))}(\mathbf{x}+\gamma\delta)\| \\ \leq&\frac{1}{2}\lambda\epsilon^2.\end{aligned}$$

Where $\mathbf{H}_{\mathcal{L}(f_\theta(\mathbf{x}),y))}(\cdot)$ is the Hessian matrix of the $\mathcal{L}(f_\theta(\mathbf{x}),y))$ w.r.t $\mathbf{x}$ and $\gamma\in[0,1]$. The last in-equation holds because we assume $f_\theta(\mathbf{x})$ is λ-smooth and δ is ϵ-bounded.

A.2. Proof of Proposition 2

$$\begin{aligned}h(\mathbf{z})&=\sup_{\mathbf{x}}\{\mathbf{x}^T\mathbf{z}-\epsilon\|\mathbf{x}\|_{2,1}\}\\&=\sum_{j=1}^{t}\sup_{\mathbf{x}_{S_j}}\{\mathbf{x}_{S_j}^T\mathbf{z}_{S_j}-\epsilon\|\mathbf{x}_{S_j}\|\}\\&=\sum_{j=1}^{t}\sup_{t\geq 0}\sup_{\|\mathbf{x}_{S_j}\|=t}\{\|\mathbf{z}_{S_j}\|t-\epsilon t\}\\&=\begin{cases}0 & \text{if } \|\mathbf{z}_{S_j}\|\leq\epsilon,\forall j\\+\infty & \text{else}\end{cases}\\&=\mathbb{I}(\|\mathbf{z}\|_{2,\infty}\leq\epsilon).\end{aligned}$$

A.3. Proof of Proposition 3

$$\begin{aligned}h(\mathbf{z})&=\sup_{\mathbf{x}}\{\mathbf{x}^T\mathbf{z}-\epsilon_1\|\mathbf{x}\|_1-\epsilon_2\|\mathbf{x}\|_2^2\}\\&=\sum_{i=1}^{n}\sup_{\mathbf{x}_i}\{\mathbf{x}_i\mathbf{z}_i-\epsilon_1|\mathbf{x}_i|-\epsilon_2\mathbf{x}_i^2\}\\&=\sum_{i=1}^{n}PQ_{\epsilon_1,\epsilon_2}(\mathbf{z}_i),\end{aligned}$$

where

$$PQ_{\epsilon_1,\epsilon_2}(z)=\begin{cases}\frac{1}{4\epsilon_2}(z-\epsilon_1)^2 & \text{if } \epsilon_1<z\\0 & \text{if } -\epsilon_1\leq z\leq\epsilon_1\\\frac{1}{4\epsilon_2}(z+\epsilon_1)^2 & \text{if } z<-\epsilon_1.\end{cases}$$

A.4. Proof of Proposition 4

This proposition holds by reusing the proof of Proposition 3 with $\epsilon_1=0$ and setting different ϵ_2 for every element of $\mathbf{z}$.

B. Implementation details

In this section, we introduced the implementation details for the experiments. All experiments were performed in PyTorch 1.12.1 using one Nvidia GeForce RTX 2080 GPU.

B.1. Experimental setup for synthesized dataset

The synthesized dataset comprised 10 classes. We generated 4096 images for training and 1024 images for testing. All images were of the size 512×512. The synthesized dataset formed a hierarchy with a circular, rectangular, and tail category. Each sample exhibited one among the three types (dark, blurred, and noisy) of random background. We

used ResNet-34 [12] pre-trained on ImageNet as the classification network and trained the network with the Adam optimizer. The experiments used a batch size of 16, an initial learning rate of 0.001, a momentum of (0.5, 0.999), and a weight decay of 10^{-5}. As the task is very simple and easy to over-fit, we set the number of training epochs to be 10.

B.2. Setup for training ImageNette

ImageNette is a smaller subset of 10 easily classified classes from Imagenet. The dataset contains 9469 images for training and 3925 images for testing. To train a classification network, we re-sampled all images into the size of 224×224 and then normalized the images to have zero mean and unit standard deviation. We used Efficientnet-B0 [34] as the classification network, which was trained by Adam optimizer with an initial learning rate of 3×10^{-4}, a momentum of 0.999, and a weight decay of 10^{-4}. The experiments use a batch size of 16. We set the number of training epochs to be 200.

B.3. Details of one-step optimization for adversarial training.

To speed up the adversarial training, we utilized the analytical solution to the approximate optimization problem given in Eq. 3 as the adversarial perturbation, which could be derived without iteration. According to Observation 1, this solution approximates the optimal solution well. Specifically, we gave the formula for the approximate solution:

- L_1**-norm**:

$$\delta^* = \epsilon \text{sgn}(\nabla_{\mathbf{x}} \mathcal{L}(f_\theta(\mathbf{x}), y)).$$

- $L_{2,1}$**-group-norm**:

$$\delta^* = \epsilon [\frac{\nabla_{\mathbf{x}_{S_1}} \mathcal{L}(f_\theta(\mathbf{x}), y)}{\|\nabla_{\mathbf{x}_{S_1}} \mathcal{L}(f_\theta(\mathbf{x}), y)\|}, \cdots, \frac{\nabla_{\mathbf{x}_{S_t}} \mathcal{L}(f_\theta(\mathbf{x}), y)}{\|\nabla_{\mathbf{x}_{S_t}} \mathcal{L}(f_\theta(\mathbf{x}), y)\|}].$$

- **elastic net**:

$$\delta^* = \epsilon_1 \text{sgn}(\nabla_{\mathbf{x}} \mathcal{L}(f_\theta(\mathbf{x}), y)) + 2\epsilon_2 \nabla_{\mathbf{x}} \mathcal{L}(f_\theta(\mathbf{x}), y).$$

We noted that for L_1-norm regularization, the analytical solution to the approximate optimization problem gives exactly FGSM.

B.4. Details of iterative adversarial training.

We conducted gradient ascend for iterative adversarial training. The number of steps was set to be 7, and the step size is 0.3. We normalized the norm of the gradients to stabilize the training. For L_1-norm, we followed the original training of PGD to utilize the sign of the gradients. For elastic-net and group-norm we normalized the gradients so it has unit L_2-norm. Moreover, to facilitate convergence, we initialized the perturbation with the analytical solution to the approximate optimization problem given in Eq. 3 of the main paper.

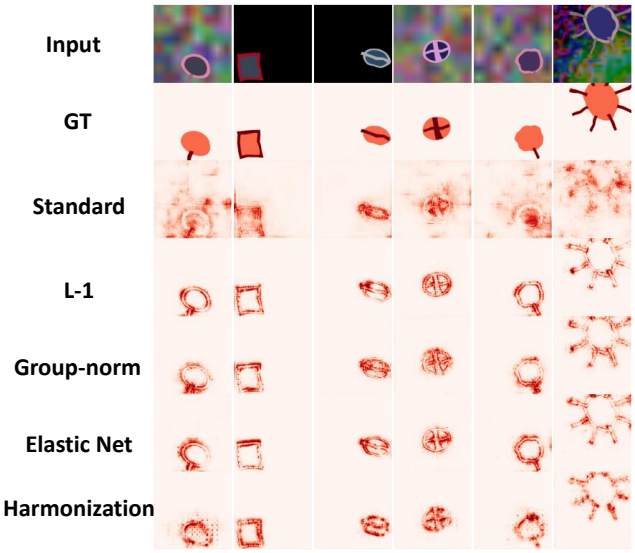

Figure 7. Qualitative results based on the synthesized dataset.

B.5. Setup for calculating DiffROAR

In the main paper, we measured the DiffROAR scores (in Sec. 5.2). DiffROAR is the difference in the predictive power of datasets, with the top-k% and bottom-k% of pixels removed by ordering the feature importance of the model. To measure this score, we re-trained the ResNet-34 model for top-k% and bottom-k% removed datasets with 150 epochs. The training setup was the same as in Sec. B.2. The differences in accuracy for top-k% and bottom-k% in the test set were presented in the main paper.

B.6. Setup for comparing the robustness.

To verify the robustness of the interpretation methods, we followed [18] to use L_2 attack on top-k overlap. Here k was selected as 40% of all the pixels. For SmoothGrad and Sparsified-SmoothGrad, we used the sample size of 64 and applied fp16 to attack, due to memory constraints. For calculating the top-k intersection metric, we set k to be 40% of all the pixels with non-zero attributions. The evaluation metrics were calculated based on 500 images randomly selected from the test set.

B.7. Setup for comparing the stability.

To examine the stability of the interpretation methods, we trained two networks with the same training process but with different initializations. Moreover, to count for the influence of training data, we randomly removed 1000 images from the training set and replenished the same amount of images from the test set. The metrics were calculated based on 500 images randomly selected from the common test set of the two runs. For calculating the top-k intersection met-

Figure 8. Adding noise during training vs. adversarial training.

ric, we set k to be 40% of all the pixels with non-zero attributions.

B.8. Experimental setup for CUB-GHA

CUB is a dataset for bird category classification. It is composed of 5990 training images and 5790 test images of 200 kinds of birds. CUB-GHA further collected human gaze data and Gaussian filter on fixation points to generate human attention maps. We resized the image to be 600×600 and randomly cropped it to be 448×448 before plugging it into an Efficientnet-B0 for classification. The remaining settings remained the same as in Sec. B.2. For adversarial training, the perturbation was calculated as an element-wise multiplication of the processed attention map and the gradients. We further normalized the gradients with L_2-norm to stabilize the training process:

$$\delta^* = (\epsilon \max(A) - 2\epsilon A) \odot \frac{\nabla_{\mathbf{x}} \mathcal{L}(f_\theta(\mathbf{x}), y)}{\|\nabla_{\mathbf{x}} \mathcal{L}(f_\theta(\mathbf{x}), y)\|}.$$

C. Additional results

C.1. Results on synthesized dataset

C.1.1 Visualization of different norm-regularization

In addition to the saliency map of the elastic net given in the main text, we visualized the saliency maps corresponding to other norm regularizations in Fig. 7. As the task is very simple and the images are very clean, different norm regularizations do not lead to quantitatively different visualizations. However, they all show significant improvement over standard training.

C.2. Results on ImageNette

C.2.1 Comparison with adding noise during training

Adding random noise during training also has the effect of denoising saliency maps and producing saliency maps of high visual quality [30]. One may think the denoising effects of adversarial training may come from adding noise during training. To illustrate it, we compared adversarial training with adding Gaussian noise during training. The results are shown in Fig. 8. Although training with additive Gaussian noise resulted in less noisy saliency maps compared with standard training, it still showed much worse visual quality compared with adversarial training. This shows adversarial perturbation indeed has effects on the sparseness of the saliency maps.

Figure 9. Explanation for a true model vs. model trained on random labels. The true model is trained with elastic net regularized adversarial training.

C.2.2 Results of free adversarial training

In addition to one-step optimization and iterative optimization, we also conducted experiments with free adversarial training [25], which optimize the network parameter and the adversarial perturbation simultaneously, so as to expedite the training. The results are shown in Fig. 13. As the results show, free adversarial training can also be used to conduct norm-regularized adversarial training. However, the visual quality is slightly lower than that of fast and iterative training. Hence, for interpretation purposes, fast or iterative training is recommended.

C.2.3 Incorporation of stronger adversarial attacks

One intuitive idea that can potentially further improve the performance of our method is through the incorporation of stronger adversarial attacks. To see how the saliency maps as we use more powerful attack tools, we compare the L_1-regularized adversarial training with Momentum iterative fast gradient sign method (MI-FGSM) [5] and vanilla FGSM. MI-FGSM leverages momentum-based iterative algorithms to better optimize the perturbation, which is a stronger adversarial attack technique. The results are shown in Fig. 10. Here we showcase two examples with different magnitudes of the attacks. Both FGSM and MI-FGSM generate saliency maps with sparseness and of pretty good visual quality. Therefore, we conclude that momentum-based adversarial training could improve the adversarial robust-

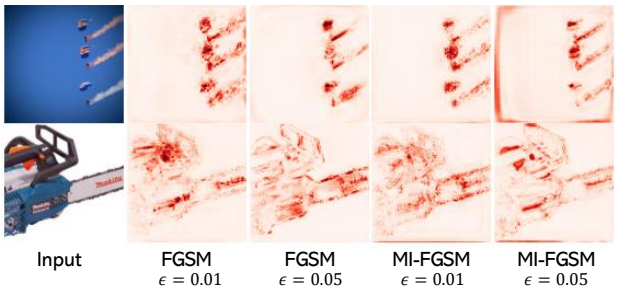

Figure 10. Qualitative comparison between FGSM and MI-FGSM.

Table 5. Fidelity of post-processing methods. The reported score is the relative drop compared with base interpretation methods.

Metric	ΔAOPC$_{\text{MoFR}}$ (%) (↓)		
Steps	20	50	100
Sparsification	2.75	18.93	36.16
Sparsified-SmoothGrad	25.34	24.84	18.13
Sparse MoreauGrad	11.29	32.32	45.61

ness of the network while performing similarly on the sparsity of saliency maps.

C.2.4 Comparison with post-processing methods

We further illustrate the results of our proposed methods compared to the post-processing sparsification strategy by measuring the fidelity of each method. We employed AOPC$_{\text{MoFR}}$ [23] at different steps as the fidelity measure and examined the fidelity of sparsified simple gradient, Sparsified-Smoothgard, and Sparse Moreaugrad [42]. As the absolute value of fidelity can be affected by the confidence of the network output, it can be unfair to compare the absolute value for different networks. We reported the relative drop in fidelity compared with a based method. Meanwhile, to eliminate the influence of smoothing operation [40] and to only focus on sparsification, we compared sparsified simple gradient with simple gradient and compared Sparsified-SmoothGard and Sparse Moreaugrad with SmoothGrad. The results are summarized in Table 5. These results suggest that post-processing methods often enforce higher sparsity at the expense of lower fidelity. On the other hand, our proposed in-processing scheme preserves the fidelity of the interpretation maps as it outputs the simple gradient maps, which are the most fundamental saliency maps.

C.2.5 Integrating with post-processing methods

Our adversarial training methodology is an in-processing scheme, which can be more faithful to the original simple gradient map compared with the post-processing scheme. Moreover, if we only care about the visual quality of the saliency maps, these two techniques are orthogonal and may have an additive effect. To illustrate this, we presented saliency maps of other interpretation methods in addition to simple gradient, using the network trained with adversarial training. The results are shown in Fig. 14. Although post-processing methods such as smoothing and sparsification can significantly improve the visual quality of saliency maps for standard training, it seems to have minor improvements on Adversarial training, as the original saliency map is already clean and sparse. Nevertheless, performing adversarial training and post-processing together still produce the most visually coherent map.

C.2.6 Sanity Check

Following [1], we conducted sanity checks to further show our in-processing scheme could maintain the fidelity of the interpretation maps. We performed both the model parameter randomization test and the data randomization test.

Model parameter randomization test. For the model parameter randomization test, we conducted cascading randomization on the ImageNette data. We computed the SSIM scores comparing the saliency maps after the progressive randomization of layers from the output layer toward the input layer with the original saliency map (Fig. 11). The decay trend shows the trained weight is important for the saliency map patterns. We also visualized the saliency maps of cascading randomization (Fig. 15). In all the cases, the saliency maps of the networks with fully reinitialized weights looked blank, implying no specific parts on the input image play major roles in the decision of the network.

Data randomization test. In the data randomization test, we randomly permuted all labels and retrained a network with the noisy dataset. The numerical results are shown in Fig. 9. We found that when using adversarial training to train a model with random labels, the model outputs exactly the same prediction disregarding the input images. As a result, the simple gradient saliency maps are composed of all zero values. This shows the interpretation methods indeed try to catch the relationship between inputs and outputs, which is essential in the application of interpretation methods to high-stake phenomena understanding.

C.2.7 Visualization of interpretation attacks

We further illustrated the robustness of interpretation maps induced by adversarial training. To do so, we visualized several examples showing the interpretation maps before and after the attack. The results are shown in Fig. 16. The result shows the simple gradient saliency map can be very vulnerable to small adversarial perturbation. Although the attacked images have no perceptible difference from the original one, the saliency map can be pretty different. On

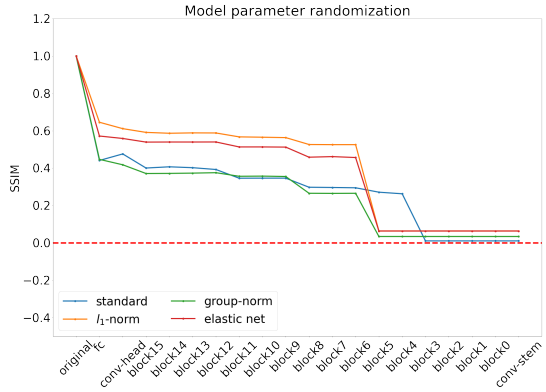

Figure 11. SSIM between saliency maps before and after cascading randomization on the ImageNette dataset.

Table 6. Top-k overlap (%) between gaze map and saliency map with L_2-norm-regularized adversarial training on CUB-GHA dataset.

ϵ	0	0.1	0.5	1.0	5.0
top 5% overlap	33.42	32.92	33.52	34.22	31.83
top 10% overlap	41.81	41.11	40.89	42.05	39.84

the other hand, adversarial training can significantly improve the robustness of the interpretation method.

C.3. Harmonization with gaze maps

C.3.1 Comparison with vanilla L_2 regularization

Our interpretation harmonization applies a weighted L_2-norm to regularize the adversarial training. One potential baseline is to use the vanilla L_2-norm to penalize the training so that only important features will be highlighted. Therefore, we reported the top-k overlap of saliency maps generated with L_2-norm regularized adversarial training and the gaze map (Table 6). We also visualized the saliency maps in Fig. 12. We discovered that vanilla L_2-norm regularization cannot help align saliency maps with human attention. As for complex images, neural networks make decisions in a way different from human beings. The features valued by the network can be very different from domain experts, although these features may also be helpful. Our interpretation harmonization strategy can narrow down the gap, making the neural network "think" like a human.

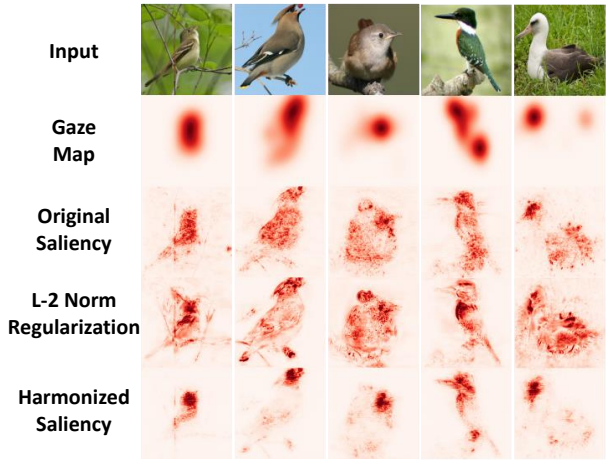

Figure 12. Qualitative comparison between L_2-norm regularized adversarial training and our harmonization strategy.

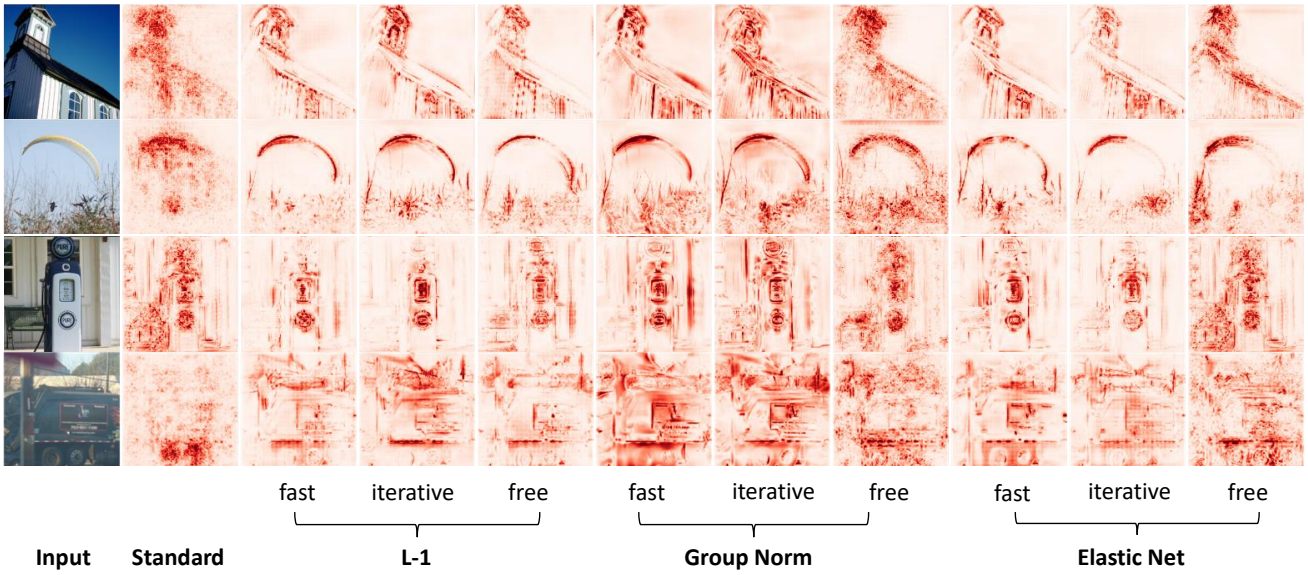

Figure 13. Qualitative comparison of saliency maps generated by networks with different adversarial training protocols (fast, iterative, free).

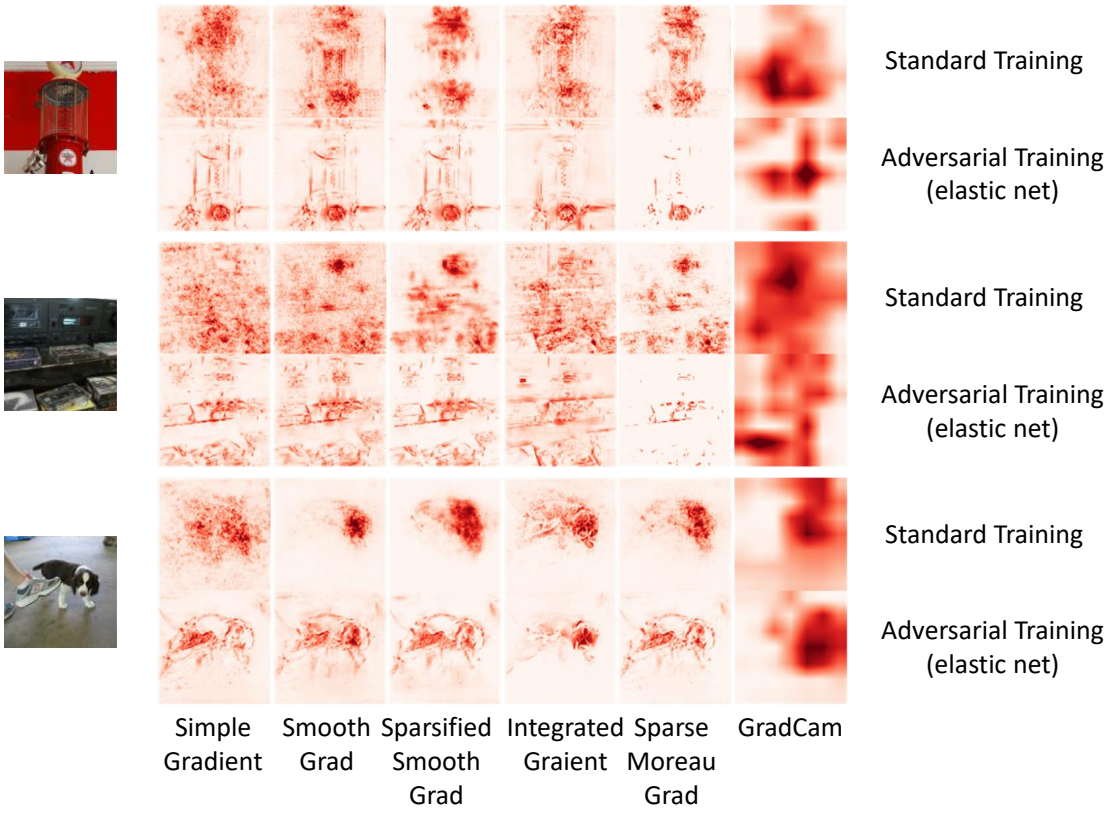

Figure 14. Integrating adversarial training and post-processing methods for generating saliency maps.

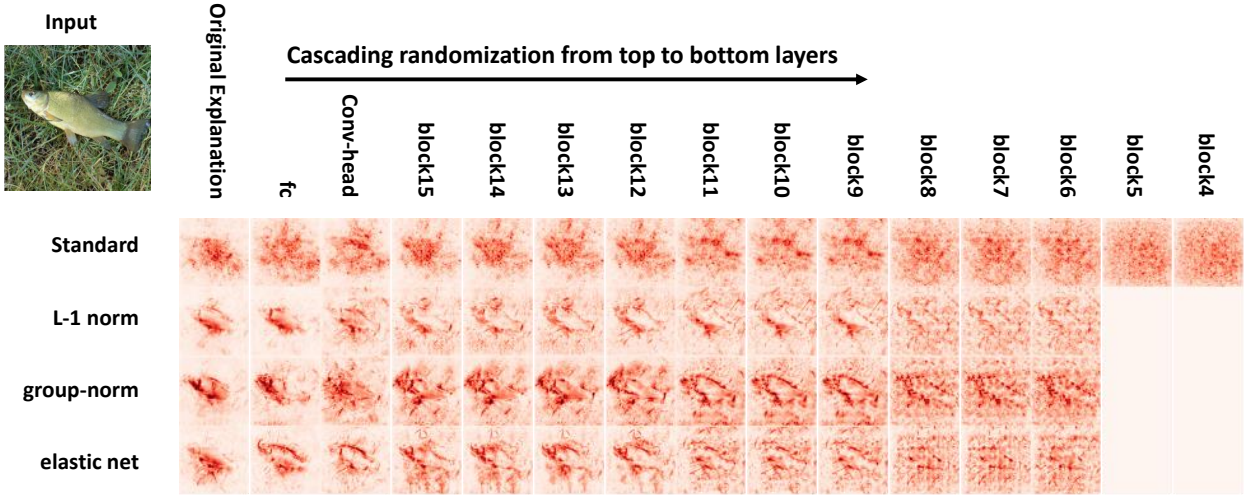

Figure 15. Cascading randomization on EfficientNet-B0 (ImageNette). The first column corresponds to the original explanations for the tench. Progression from left to right indicates complete randomization of network weights. Some saliency maps look blank because the importance scores for all pixels are uniformly small.

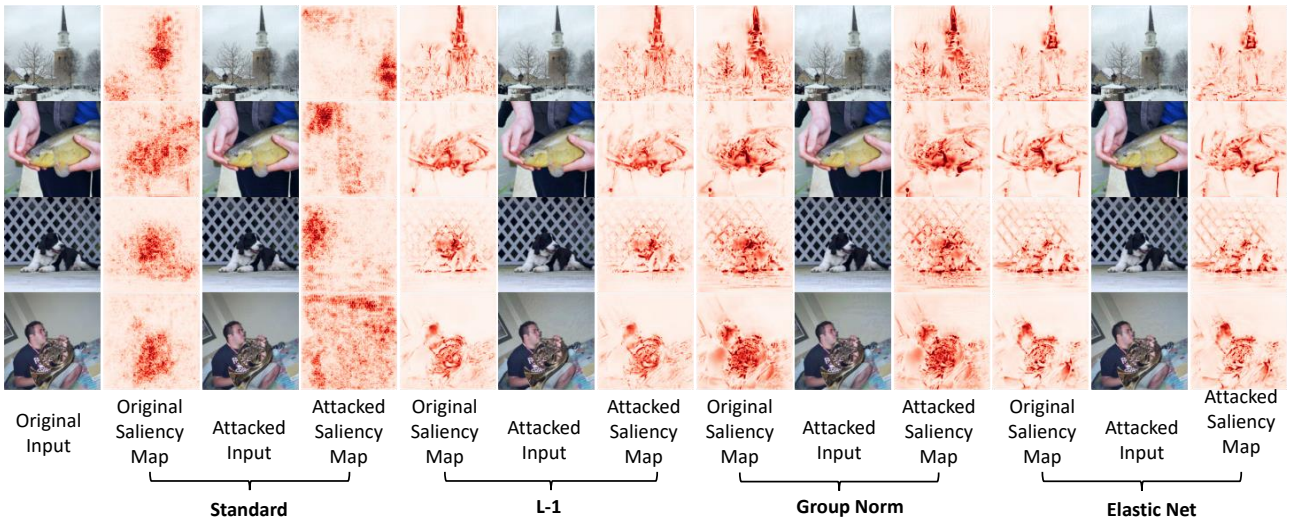

Figure 16. Visualization of attacked image and attacked saliency maps.