Towards Hiding Adversarial Examples from Network Interpretation

Akshayvarun Subramanya* **Vipin Pillai*** **Hamed Pirsiavash**
University of Maryland, Baltimore County (UMBC)
{akshayv1, vp7, hpirsiav}@umbc.edu

Abstract

Deep networks have been shown to be fooled rather easily using adversarial attack algorithms. Practical methods such as adversarial patches have been shown to be extremely effective in causing misclassification. However, these patches can be highlighted using standard network interpretation algorithms, thus revealing the identity of the adversary. We show that it is possible to create adversarial patches which not only fool the prediction, but also change what we interpret regarding the cause of prediction. We show that our algorithms can empower adversarial patches, by hiding them from network interpretation tools. We believe our algorithms can facilitate developing more robust network interpretation tools that truly explain the network's underlying decision making process.

1 Introduction

Deep learning has achieved great results in many domains including computer vision. However, it is still far from being deployed in many real-world applications due to reasons including:

(1) Explainable AI (XAI): It is difficult to explain the prediction of deep networks simply because they are complex models with large number of parameters. Recently, XAI has become a trending research area in which the goal is to develop reliable interpretation algorithms that can explain the underlying decision making process. Designing such algorithms is a challenging task and considerable research [1, 2, 3] has been done to describe *local explanations* - explaining the model's output for a given input. In this paper, we will focus on this kind of interpretation. Most of the these algorithms rely on studying the gradient of the output of a machine learning model with respect to its input.

(2) Adversarial examples: Many works have shown that deep networks are vulnerable to adversarial examples, which are carefully constructed samples created by adding imperceptible perturbations to the original input to change the final decision of the network. These adversarial examples have been demonstrated to fool many different algorithms for image classification, detection, segmentation and speech recognition tasks. This is important for two reasons: (a) Such vulnerabilities could be used by adversaries to fool AI algorithms when they are deployed in real-world applications such as Internet of Things (IoT) [4] or self-driving cars [5]. A practical scenario is through creating adversarial patches [6] where we restrict the spatial dimensions of the perturbation, but remove the imperceptibility constraint. These patches can be printed and 'pasted' on top of an image to mislead classification networks. (b) Studying these attacks can lead to better understanding of how deep networks work and possibly improve generalization on new environments.

We specifically focus on adversarial patches rather than regular adversarial examples since patches are a more practical form of attack, and also the cause of the misclassification is strictly limited to the patch area. Hence, it is not trivial for the attacker to mislead the interpretation to highlight non-patch regions without perturbing them.

*Equal contribution

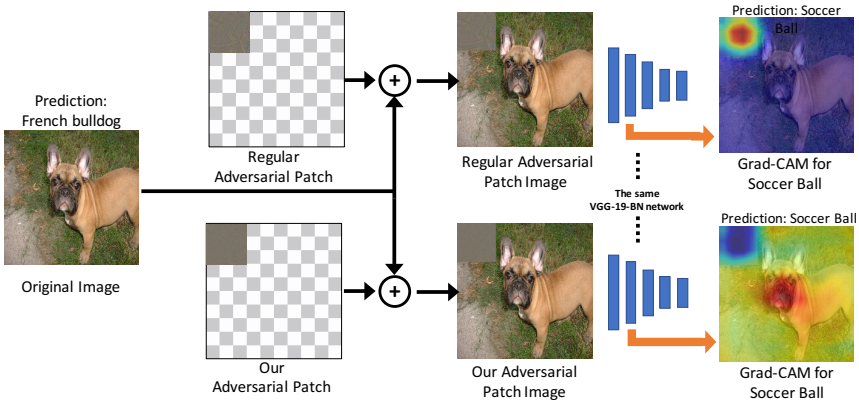

Figure 1: We show that Grad-CAM highlights the patch location in the image perturbed by regular targeted adversarial patches [6] (top row). Our modified attack algorithm goes beyond fooling the final prediction by hiding the patch in the Grad-CAM visualization, making it difficult to investigate the cause of the mistake. Note that Grad-CAM visualizes the cause of target category.

Consider an example of adversarial attack using adversarial patches as seen in [6], we show that an interpretation algorithm like Grad-CAM [3] usually highlights the location of such an adversarial patch making it clear what image patch was responsible for misclassification (Figure 2). This is expected as the adversary is restricted to the patch area and the patch is the cause for the final prediction of the target category. We are interested in designing stronger attack algorithms that not only change the prediction but also mislead the interpretation of the model to hide the attack from investigation. As an example, assume an adversary (one in a crowd of pedestrians) is wearing a t-shirt with a printed adversarial patch on the back that fools a self-driving car leading to an accident. Now, a simple investigation with standard network interpretation methods may reveal which pedestrian in the scene was the cause of the wrong decision, and thereby identifying the adversary. However, we show that it is possible for the adversary to learn a patch without revealing their identity (patch location) and thus escape scrutiny. We show a concrete example of this in Figure 1.

2 Method

We propose algorithms to learn perturbations that when added to the input image, can change the interpretation of the model's final decision. We will focus on Grad-CAM [3] in designing our algorithms, since it is one of the popular network interpretation methods.

Background on Grad-CAM visualization

Consider a deep network for image classification task, e.g., VGG, and an image x_0. We feed the image to the network and get the final output y where y^c is the logit or class-score for the c'th class. To interpret the network's decision for category c, we want to generate heatmap G^c for a convolutional layer, e.g, *conv5*, which when up-sampled to the size of input image, highlights the regions of the image that have significant effect in producing higher values in y^c. We denote A^k_{ij} as the activations of the k'th neuron at location $(i; j)$ of the chosen layer. Then, similar to [3], we define normalized visualization $\hat{G}^c$ as:

$$\hat{G}^c := \frac{G^c}{jG^cj_1}; \qquad \text{where} \quad G^c_{ij} = max(0; \sum_k {}^c_k A^k_{ij}) \quad \text{and} \quad {}^c_k = \frac{1}{Z} \sum_i \sum_j \frac{@y^c}{@A^k_{ij}}$$

Background on adversarial patches:

Consider the input image x_0 and a predefined constant mask m that is 1 on the location of the patch and 0 everywhere else. We want to find an adversarial patch z that changes the output of the network to category t when pasted on the image, so we solve:

$$z = \arg\min_z \, `_{ce}(x \quad (1 \quad m) + z \quad m; t)$$

where $`_{ce}(:; t)$ is the cross entropy loss for the target category t and is the element-wise product. This results in adversarial patches similar to [6].

2.1 Misleading interpretation of adversarial patches

Targeted attack: To fool the network's interpretation so that the adversarial patch is not highlighted at the interpretation of the final prediction, we add an additional term to our loss function in learning the patch to suppress the total activation of the visualization at the patch location m. Hence, assuming the perturbed image $x = x_0 \quad (1 \quad m) + z \quad m$, we optimize:

$$\arg\min_z \; \ell_{ce}(x; t) + \sum_{ij} \hat{G}^t(x) \quad m$$

where t is the target category and is the hyper-parameter to trade-off the effect of two loss terms. We choose the target label randomly across all classes excluding the original prediction similar to "step rnd" method in [7].

Non-targeted attack: A similar approach can be used to develop a non-targeted attack by maximizing the cross entropy loss of the correct category:

$$\arg\min_z \; \max(0; M \quad \ell_{ce}(x; c)) + \sum_{ij} \hat{G}^a(x) \quad m \quad \text{where} \quad a = \arg\max_k p(k);$$

where c is the predicted category for the original image, $p(k)$ is the probability of category k, and a is the top prediction at every iteration. Since cross entropy loss is not upper-bounded, it can dominate the optimization, so we use contrastive loss to ignore the cross entropy loss when the probability of c is less than the chance level, so we use $M = \log(p_0)$ where p_0 is the chance probability (e.g., 0.001 for ImageNet). Note that the second term is using the visualization of the current top category a.

To optimize above loss functions, we adopt an iterative approach similar to standard projected gradient decent (PGD) algorithm [8]. We initialize z^0 randomly and then iteratively update it by: $z^{n+1} = z^n \quad Sign(\frac{\partial \ell}{\partial z})$ with learning rate . At each iteration, we project z to the feasible region by clipping it to the dynamic range of the image values.

3 Experiments

In all our experiments, we use the pre-trained VGG-19 model with batch normalization provided by PyTorch and a patch of size 64x64 (8% of the image area) on the top-left corner for 50,000 images of size 224x224 in the validation set of Imagenet 2012. We do 750 iterations with $= 0.001$. To evaluate how much the patch is highlighted in the visualization, we construct the visualization heatmap $\hat{G}^c$ for the mistaken category, and calculate the ratio of the total energy at the patch location to that of the whole image. We call this metric "energy ratio". It will be 0 if the patch is not highlighted at all and 1 if the heatmap is completely concentrated inside the patch. The quantitative results are shown in Table 1 and qualitative results can be found in Figure 2.

Method	Non-Targeted		Targeted		
	Acc (%)	Energy Ratio (%)	Acc (%)	Target Acc (%)	Energy Ratio (%)
Adversarial Patch [6]	0.06	38.95	0.02	99.98	58.39
Our Patch	0.05	**2.00**	2.95	77.88	**5.21**

Table 1: Comparison of heatmap energy within the 8% patch area for the adversarial patch [6] and our patch. Accuracy denotes the fraction of images that had the same final predicted label as the original image. Target Accuracy denotes the fraction of images where the final predicted label has changed to the chosen target label.

4 Conclusion

We present novel adversarial attack algorithms that go beyond fooling the prediction by hiding the cause of the mistake from common interpretation tools to result in a stronger attack. Our work shows that there is a need for more robust deep learning tools that reveal the correct cause of network's predictions.

Acknowledgement: This work was performed under the following financial assistance award: 60NANB18D279 from U.S. Department of Commerce, National Institute of Standards and Technology, and also funding from SAP SE.