Exploration of Attack Methods on CNN

Jason Anderson, Cheng Chang, Sanchit Jain

Stanford University,
CS 231N, Spring 2023

Abstract

Many Convolutional Neural Networks (CNNs) architectures, such as ResNet, provide limited robustness guarantees, and may be vulnerable to attacks with adversarial samples that significantly degrade their performance. We investigate methods that attack CNNs used for image classification by attempting to minimally modifying input images and creating adversarial samples. Using a ResNet-50 model pre-trained on the Oxford-IIIT pets dataset, we compare previously proposed methods of creating adversarial samples, including the Adversarial Patch method (AP), and variants of the Fast Gradient Sign Method (FGSM). These experiments inform us to design 3 different neural network structures and training mechanisms to generate adversarial examples. With the classification-inaccuracy on the pre-trained ResNet-50 model ranging from 20% to 99% using our adversarial generators, our experiment reveal a trade-off between magnitude of the change applied to samples and attack effectiveness when generating adversarial samples.

Introduction

Fast Gradient Sign Method

Introduced in the paper *Explaining and Harnessing Adversarial Examples,* Fast Gradient Sign Method is a type of attack that adds small modifications to an input image to make a classifier misclassify it. Let an input image be *x* and its true label *y*. Let the neural network parameters be *θ* and the loss function by which the neural network is trained be *J(θ, x, y)*.

$$\arg\max_{x} = J(\theta, x, y) + (x^{\text{adversarial}} - x)\nabla_x J(\theta, x, y)$$

$$\text{s.t.} \quad ||x^{\text{adversarial}} - x||_\infty < \epsilon$$

$$x^{\text{adversarial}} = x + \epsilon \cdot \text{sign}\left(\nabla_x J(\theta, x, y)\right)$$

Iterative Fast Gradient Sign Method

Introduced in the paper *Adversarial Examples in the Physical World*, this method is an extension of the FGSM method, which is applied multiple times with small step size. It reduces the ResNet-50 model accuracy from 93% to 4% in our experiment.

Adversarial Patch

Introduced in the paper *Adversarial Patch* by Brown et.al., this method uses large perturbations to cause a neural network to misclassify, but the goal was to create robust perturbations, which when added to images, could cause lots of classifiers to misclassify the resulting image. It worked as expected in our experiment for post-transformation images.

Methods

We investigated the 3 methods discussed in the Introduction, and present results on 3 novel methods which were influenced by FGSM & I-FGSM. We train a generator which generates adversarial noise, which, when added to an input, may cause the ResNet50 model pre-trained on The Oxford-IIIT pets dataset to misclassify an input. The misclassification is either targeted towards another label, or non-targeted (any incorrect label).

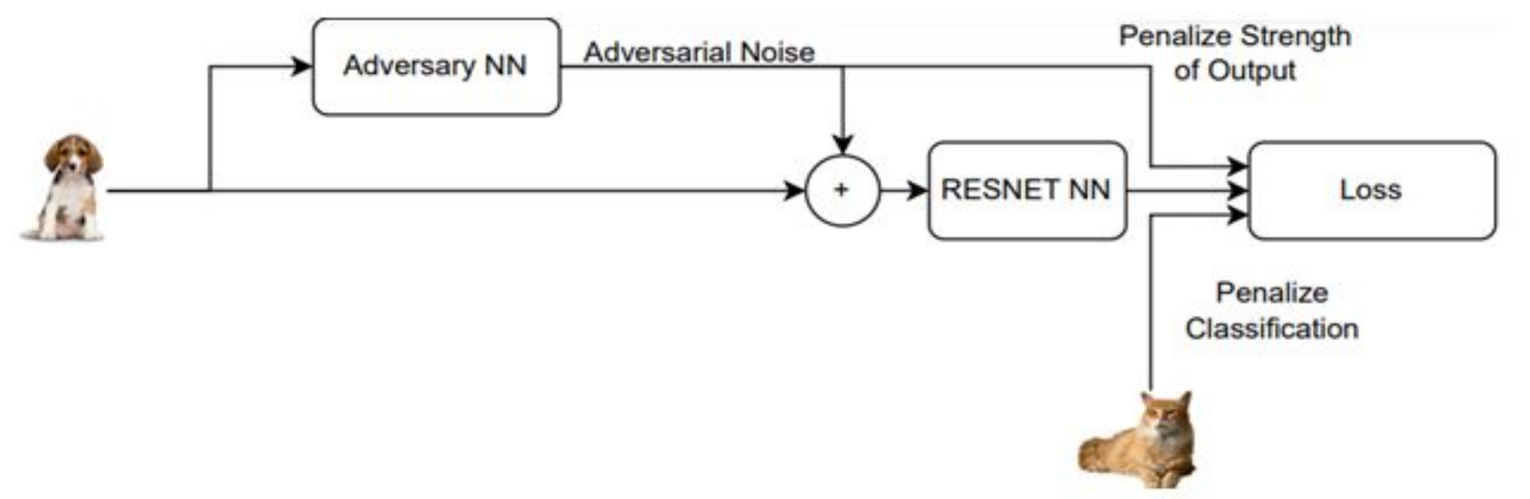
- **FGSM noise regression**
  One naive approach we try is to generate all noises for the Oxford-IIIT pets dataset using the I-FGSM method described on the left, and try to train our generator through a regression scheme. Our loss function here is:

  $$J = \text{CE\_Loss}(\text{Regressor}(x), \text{FGSM}(x)) + \lambda Reg(x)$$

  Where CE_Loss is cross entropy loss and Reg is either l2-loss, l1-loss, or a customized term where each value larger than a threshold is penalized through counting.

- **Adversarial noise generator (optimize for target label accuracy)**
  Here, we try to optimize the image for high target class probability. Our loss function here contain two components: (1) a cross entropy loss on the target label and (2) a norm (l2 norm in for our results) on the output noise.

  $$\hat{y} = \text{Adversary}(x) + x$$
  $$J = \text{CrossEntropyLoss}(\hat{y}, y_{\text{desired spoof}}) + \lambda||\text{Adversary}(x)||$$



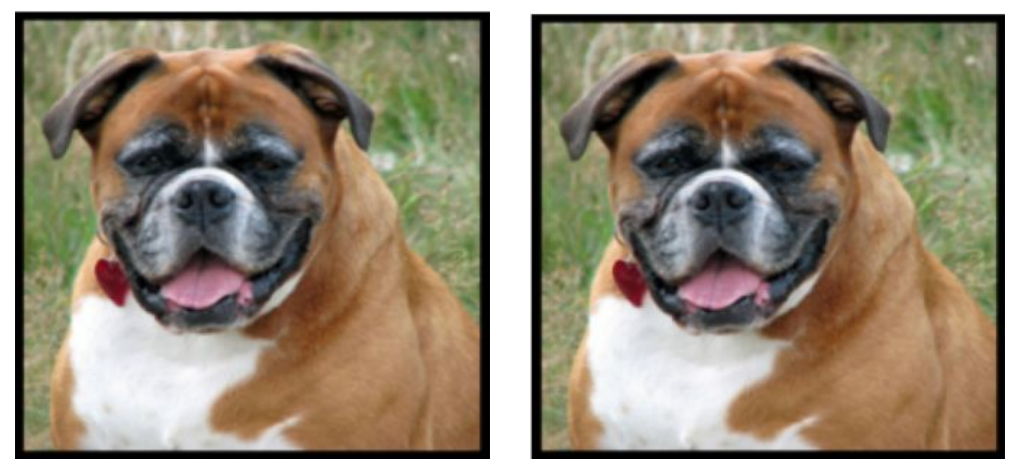
- **Adversarial noise generator (Optimize for correct label inaccuracy)**
  This noise generator is the non-targeted variant of the adversarial noise generator described above.
  Here, our loss function contains: (1) negated cross entropy loss on the correct labels and (2) a norm (l2) on the output noise.

  $$\hat{y} = \text{Adversary}(x) + x$$
  $$J = -\text{CrossEntropyLoss}(\hat{y}, y_{\text{actual\_label}}) + \lambda||\text{Adversary}(x)||$$

  We used a hyperparameter optimization tool Optuna to search for hyperparameters such that the ratio of the invalidation accuracy of the pretrained ResNet50 model to the square of the L2 norm of the adversarial noise was maximized.

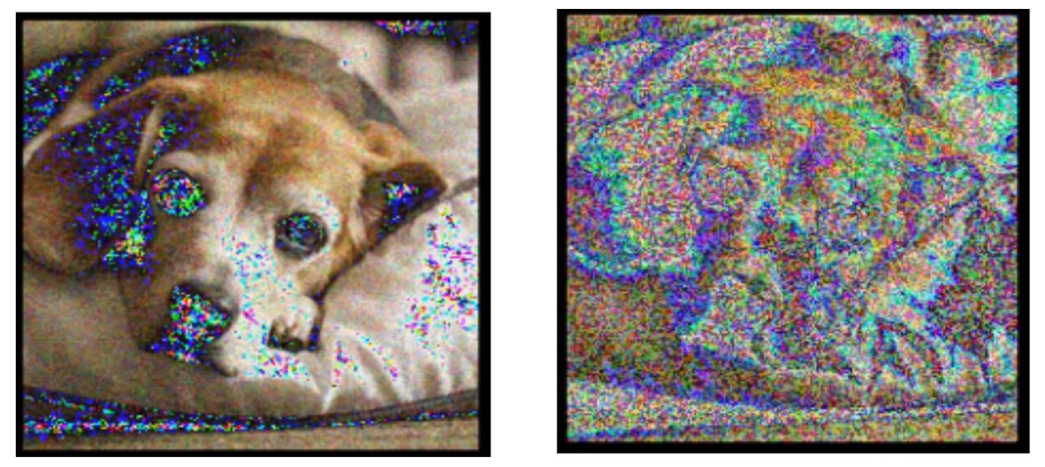
Results

We first used **I-FGSM** until the resulting image is misclassified



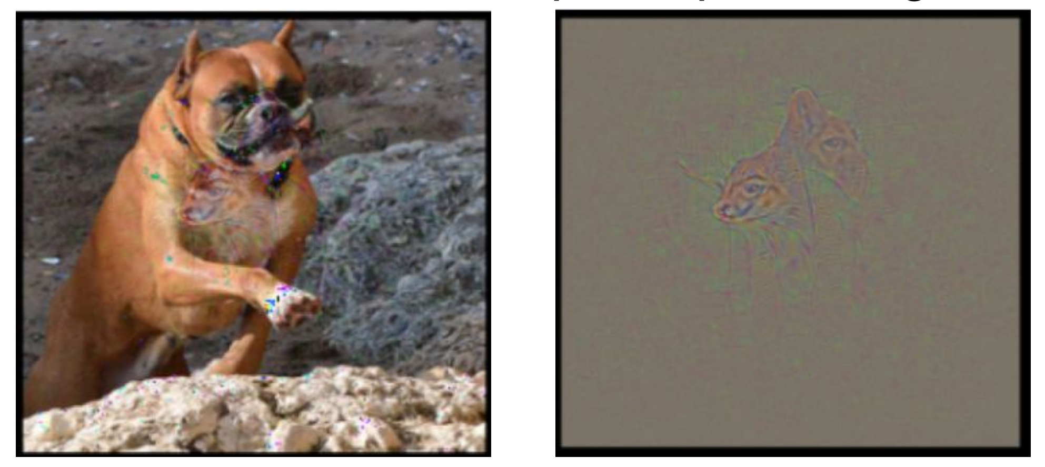
Left: original image of a boxer that was classified correctly. **Right**: after adding minimal modifications to the original image, it was misclassified as an American Bulldog.

For our **FGSM noise regression approach**, the adversarial images produced (which caused the pre-trained ResNet50 model trained on the Oxford-IIIT pets dataset to misclassify) had clearly visible modifications -



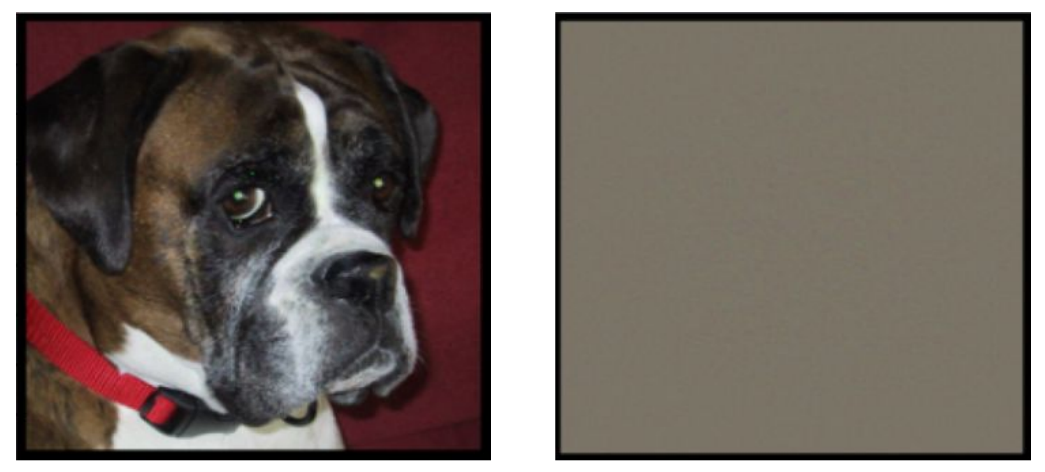
Left: Changes with around 20% chance of misclassification rate. **Right**: 80% misclassification rate but drastic changes.

We first tried our **adversarial noise generator approach for targeted attacks**, but it had imprints pertaining to the target label -



Left: A boxer misclassified as an Abyssinian cat using the target label accuracy optimization approach. **Right**: adversarial noise with a visible cat imprint. This shows that the generator is actually learning a saliency map of the cat representation

We then tried a **variant of that adversarial noise generator for an untargeted attack** (misclassify without any specific target label), and most images produced had imperceptible changes -



Left: Modified image of the dog misclassified as a Bombay cat. **Right**: corresponding adversarial noise at the same scale (224 x 224).

Conclusion: There's a trade-off between magnitude of the change applied to samples and attack-effectiveness when training the adversarial sample generator. This tradeoff can be mitigated by having constant access to the target model output and by providing our generator with more useful loss function during training. Things to consider includes loss terms, regularization function, and various hyperparameters related to them.
This shows that a carefully designed neural network can be used to attack CNN-based classifiers, and the attack can be semi-black box with almost imperceptible adversarial samples.