CS289 project proposal

Team member: Shuhui Huang, Mingjia Chen, Ruonan Hao

Our project is: there is an interesting paper on fooling neural networks for classification (see also this blog post). But these methods examine the neural network they want to fool before adding noise to the images. It would be interesting to study the effect of adding noise to images independent of the classifier, and see how the classifier accuracy varies with that.

First, our goal is adding noise to the images, then using the method as "Explaining and Harnessing Adversarial Examples" describes to create a fooling image, and see how the classifier accuracy varies with different noises. It can be applied in image processing and recognition area. This problem is important because the neural networks can be easily fooled or hacked by adding certain structured noise in image space due to ignoring non-discriminative information in their input. This work raises security concerns because an adversary could conceivably generate a fooling image of any class on their own computer and upload it to some service with a malicious intent, with a non-zero probability of it fooling the server-side model (e.g. circumventing racy filters).

We will use the MNIST data set. The MNIST dataset consists of handwritten digit images and it is divided in 60,000 examples for the training set and 10,000 examples for testing. In many papers as well as in this tutorial, the official training set of 60,000 is divided into an actual training set of 50,000 examples and 10,000 validation examples (for selecting hyper-parameters like learning rate and size of the model). All digit images have been size-normalized and centered in a fixed size image of 28 x 28 pixels. In the original data set each pixel of the image is represented by a value between 0 and 255, where 0 is black, 255 is white and anything in between is a different shade of grey.

Here are some examples of MNIST digits:

For convenience, we pickled the dataset to make it easier to use in python he pickled file represents a tuple of 3 lists: the training set, the validation set and the testing set. Each of the three lists is a pair formed from a list of images and a list of class labels for each of the images. An image is represented as numpy 1-dimensional array of 784 (28 x 28) float values between 0 and 1 (0 stands for black, 1 for white). The labels are numbers between 0 and 9 indicating which digit the image represents. The code block below shows how to load the dataset.

The method we are going to use, in short, is to create a fooling image we start from whatever image we want (an actual image, or even a noise pattern), and then we add noises to it, use backpropagation to compute the gradient of the image pixels on any class score, and nudge it along. We may, but do not have to, repeat the process a few times. This method has been introduced by "Explaining and Harnessing Adversarial Example" and "Breaking Linear Classifiers on ImageNet".

We haven’t done any preliminary work yet.

The core of the work for our project is to compare the effect of adding different noises to the fooling images. We are going to use different ways to add noises to the same images, and see how the classifier accuracy varies with that. We expect to be judged primarily on our writing and exhaustive exploration of methods to add the noises.