Summary for Adversarial Image Generation Challenge

Prepared by Jacey Wu

In the past few years, convolutional neural networks (CNNs) have expanded their great influences in many computer vision tasks, and successfully pushed the state-of-art result to a brand new level. Especially in image classification, one of the tasks CNNs achieved early successes in, the performance of some CNN models is already compatible to human performance in accuracy measure. However, there are also many discussion on the robustness of the CNN performance, namely, if the good CNN classification ability can be generalized when inputs start to variate.

Problem:
In this challenge, we challenged the stability of CNNs' performance, and investigated the possibility to "fool" a CNN classifier with adversarial images generated from altering the original inputs which CNN can correctly label initially. More precisely, we want to make small changes to the original image, so that human would still classify the altered image as the original class while CNN would be "fooled" and classify the altered image as a completely different class.

Idea:
Follow the work of "Explaining and Harnessing Adversarial Examples" by Goodfellow et al, our generation process involves calculating the gradient of the loss with respect to the input while holding the weights in the CNN fixed. We also experimented with adding a regularization term into the original loss function, which penalizes large alteration to the original image. The intention of this regularization is to preserve the human-recognizable features in the original image to the max degree.

Experiment details:
We used the MNIST digit data set for fast training and evaluation. The data set was split to training set, validation set and test set, each contains 55000, 5000, 10000 images.

We implemented a simple digit classifier which achieved above 0.99 accuracy on both validation set and test set. We did not spend too much time tuning the MNIST classifier because our main focus is on generating adversarial images.

To generate adversarial images of digit '2' which the above classifier would misclassify to '6', we went through the following steps

1. Extract 10 images of '2' from the validation set and reconstruct the their label of '6'
2. Restore and fix the the weights of the pre-trained classifier
3. Calculate the cross entropy loss with the extracted inputs and new labels
4. Calculate the gradient of the loss with respect to the input images
5. Update input images with x = x - update_rate * dx

6. Go back to step 3 and repeat until the all images in the batch are labeled as ‘6’ by the classifier

Conclusions:

- We can construct adversarial images that still preserve the features of the original image but are misclassified by CNNs.
- The training steps for the adversarial images are not long. In our experiments, a batch containing 10 images will be transformed with in 10 - 15 iterations.
- Based on further readings on this topic, we found not only CNNs would be affected by the adversarial images generated following the above idea. The limitation might caused by the linearity nature in the convolution step.
- We investigated the effectiveness of adding a L2 regularization term to the loss when generating adversarial images. We tested different weights of the L2 regularization but found that visual quality of the output image is only minorly influenced by the regularization term. This is a little out of our expectation. However, when this term is set too big, the image obtained cannot fool CNNs. Thus, we suspect regularization still play an important role in enforcing the adversarial image to be similar as its original copy, and that the regularization influence in MNIST data set is limited might because the low resolution of MNIST images.

Instruction on re-run experiment code:

Dependency:

- TensorFlow V1.1
- Numpy
- Matplotlib

Run `./classifier/train_mnist_classifier.py` to train MNIST classifier. Train parameter can be adjusted in `./classifier/const_cls.py`

Run `./adversary/train_adversarial_im.py` to generate adversarial images. Train parameter can be adjusted in `./adversary/const_adv.py`