Machine Learning Engineer Technical Challenge

Purpose

This technical challenge is for candidates to demonstrate their problem solving and learning abilities. All tasks in this challenge are based on real life examples of what you could work on at 500px. There's no right or wrong way to do this challenge, so don't stress out too much about it. The purpose is to facilitate a conversation about machine learning and best practices. We want to see how you think and what your opinions are.

Ground Rules

- You are free to choose any language you want. At 500px, we use mainly use Python for machine learning applications. We also use Jupyter notebooks to share reproducible results of such research.
- You are free to reuse any source code or publications you find online, as long as you you include proper attribution and **have clear separation between the code you borrowed and the code you wrote yourself**. Keep dependencies to a minimum.
- You should **keep your code simple and focus on readability** of your code. The first thing we will look into is the commit history to see your thought process and how your solution have evolved. **We value thoughtfully clean and communicative code so other contributors can easily understand and build on top of it.**
- You may skip any parts of the challenge if you get stuck or don't have relevant experience. However, we encourage you to learn and demonstrate newly acquired skills.
- You should check your solution into **GitHub** and provide basic instructions how to reproduce your results.
- You are free to spend as little or as much time as you want on this challenge.
- You are expected to learn something new after you complete the challenge :)

Assessment

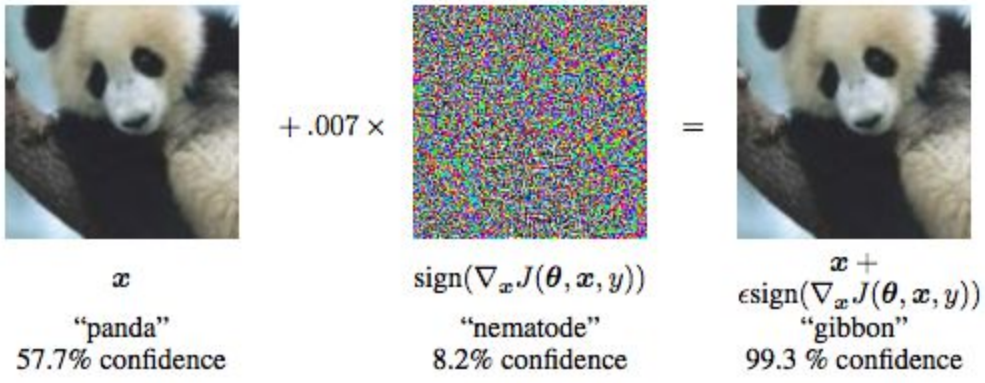
Your solution will be assessed before the on-site interview. If you are invited for an on-site interview, we will go through your solution together with you so that you can explain the thoughts behind your technological, architectural and algorithmic decisions.

The challenge

TL;DR: Create adversarial images to fool a MNIST classifier in Tensorflow.

Deep convolutional neural networks (CNN) are state of the art models for image classification and object detection. Such models play crucial role at 500px where we use them for many applications like automatic keywording, people detection and image search. It's important to understand how they work and what their limitations are.

One known “limitation” of CNN is that they can be fooled to misclassify an image with high confidence by slightly perturbing the pixels. This is illustrated on the image below:



Explaining and Harnessing Adversarial Examples by Goodfellow

The delta between the original image and the adversarial one is so small that it is impossible for humans to detect. The fun fact is other machine learning models like SVM and logistic regression can be tricked in the similar manner.

Note that the “fast gradient sign” method presented in the original paper by Goodfellow produces adversarial images for a random target class. In this challenge we would like to generate adversarial images to misclassify any examples of ‘2’ as ‘6’ specifically. This puts certain implications on the final solution.

One of the useful application for adversarial images is that if you train your deep CNN classifier on them you can improve its accuracy on non-adversarial examples.

In this challenge you are given an opportunity to learn how to generate adversarial examples and also gain practical experience using Tensorflow.

Here are the steps you should go through:

1. Read the **Ground Rules**!
2. Learn how adversarial examples are created. For example, “Breaking Linear Classifiers on ImageNet” gives a good overview on the subject.
3. Install Tensorflow
4. Follow “Deep MNIST for Experts” tutorial to get the MNIST classifier running.
5. Expand the code from the previous step to generate adversarial images. Specifically, pick 10 images of digit ‘2’ which are correctly classified as ‘2’ by the trained model and modify them so the network incorrectly classifies them as 6.

6. Generate adversarial examples and save them as a single image containing a grid of 10 rows and 3 columns. The rows correspond to the selected examples of ‘2’. The columns are *original image*, *delta* and *adversarial image*. Provide link to the resulting image.
7. Make your code clean and readable. Add comments where needed.
8. Analyze your results. What interesting insights did you learn? This step is specifically left open ended so we could have interesting conversation when we meet in person.