ML Programming Problem

ML Programming Problem

Introduction

Think of this challenge as an opportunity to show us what “good” looks like to you; and a fun way to showcase your skills.

Here are some tips and guidelines:

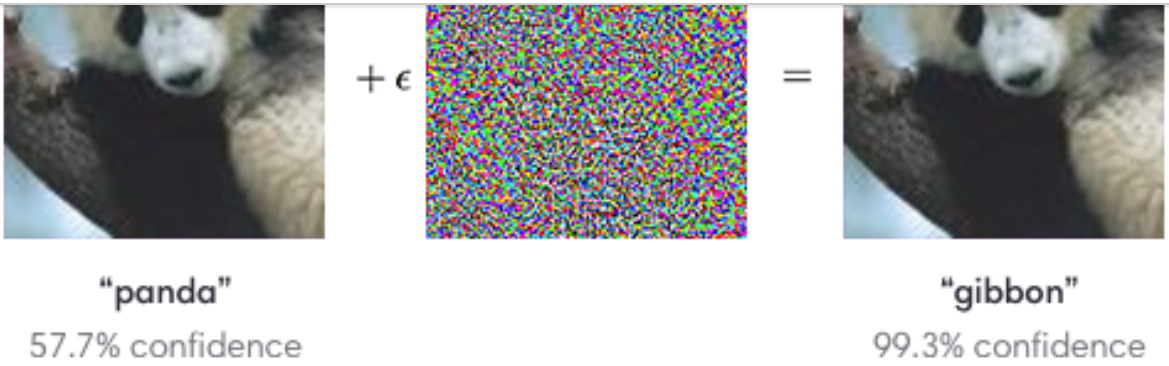
- We don’t expect you to spend more than 2-3 hours on this challenge.
- If you don’t have time to fully complete the challenge, please still send it in and indicate what your next steps would be. Remember to try to solve the hardest problems first.
- Use any language and frameworks you want.
- Keep it simple.
- Put your code on a public source repository (such as GitHub) and give us the URL.
- Please submit your commit history, we are interested to see how you approach the challenge and how you verify the validity of your solution.
- We should be able to run your code without any crazy steps.
- Imagine this is production quality code that you are submitting to your team for review.

Problem: Adversarial Noise

The task involves developing a program that manipulates images by adding adversarial noise. This noise is designed to trick an image classification model into misclassifying the altered image as a specified target class, regardless of the original content.

- You may select any pre-trained image classification model for this task. A model from the torchvision library is recommended, but not mandatory.
- The core challenge is to effectively introduce noise into the image in such a way that the model misclassifies it as the desired target class, without making the noise perceptible to a casual human viewer.

ML Programming Problem



Input:

- The user will provide an image and specify a target class.

Output:

- The program should output an image that has been altered with adversarial noise. The altered image should be classified by the model as the target class, irrespective of the original image's content. The altered image should not be obviously different to the original.

For your submission, envision that this is a small library that could reasonably be published for users to easily generate these adversarial images. Or alternatively, imagine that your code would be integrated into an existing production library.