Intelligent Systems

Project 02 Report

Colton Hill

December 17, 2021

0 Introduction/Brief:

For this project, I have attempted to recreate the results found in the paper **"Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images"**. Specifically, the gradient ascent algorithm used to modified starting images into “trick” images (images a human still recognizes as class A, but a CNN recognizes as class B). Originally, I wanted to also do an evolutionary algorithm, however, the evolutionary mutation process proved to be more intensive than time permitted. Still, the gradient ascent works every well, only required 30 seconds to “spoof” an image. Overall, I am quite satisfied with the results.

My “Spoofer” is a aggregate python classes which takes a pre-trained CNN (from PyTorch), a starting and ending class id, and then modifies the starting image until the pre-trained CNN classifies it as the ending class id. This modification is done using gradient ascent, and is typically done under 17 iteration. This project comes with a random subset of the ImageNet data set, which almost all pre-trained CNNs from PyTorch are trained on. By selecting a starting index in “main.py”, the program will load the random sample corresponding to that index into memory, prepossesses it, and begin passing it though the “spooffer” until it is classified as the target class to the specified confidence.

Once done, the program will print out the original classification and confidence followed by the resulting classification and confidence. This is necessary firstly to demonstrate that yes, this process is working correctly. Secondly, it is helpful because some of the images in there base forms already trick the models *(see example 420, the eel)*. This helps the user pick a “good” example that will show highly confident images swapping classification. From what has been anecdotally tested, most non-fish animals have a natural +95% confidence.

1 Usage

To begin using the CNN, please follow the “README.md” to install all the required libraries. Once all the libraries are installed and the repo is cloned, there are 3 main parts of main that can be change, the initial class (line 10), target class (line 11), and confidence (line 23). The confidence should be left alone for best results. To find what number correlate to what classifications, see “class_table.py”.

Once selected, the program can be run with my executing “main.py”, which will print out the results and make a new directory in the output images for you to review. This directory will be named after the direction of the transformation you ordered, contained the original and spoofed images.

2 Summary

Through the course of developing and completing this project, I’ve had to do a fair bit of research about CNNs, image reprocessing, imageNet classification, gradient assent, and the PyTorch library. As a result, I’ve learned a lot more about CNN classification, and feel that I have gained a good understanding how CNNs work and several of the problems associated with them. While I wasn’t able to take this project to the next step and ask "why do these images fool the CNN?", I am very happy with the knowledge I’ve gained. Now, I am confident that I could continue development and research into CNN vision problems and begin asking the deeper "why" questions. This project was fun and well worth the effort.

Thank you for allowing me to pursue this challenge!

3 Sources & Credit

- Tiny ImageNet Tutorial:
  https://towardsdatascience.com/pytorch-ignite-classifying-tiny-imagenet-with-efficientnet-e5b1768e5e8f
- Input Image Samples:
  https://github.com/EliSchwartz/imagenet-sample-images
- Preprocessing and Gradient Ascent:
  https://github.com/utkuozbulak/pytorch-cnn-adversarial-attacks
- Classification Key:
  https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a