# Math110b Project3 writeups

Huidi Wang

September 2021

## 1 Introduction

In this project, we were asked to conceal an image within another image. We implement LSB and NN to compare their performance.The natural images are compressible, usually a full size image of several megabytes can be reduced to kilobytes level easily without lossing too much important information. Such property has been used widely to denoising, deblur, etc.The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries in which encryption is illegal.

## 2 Purpose of this Project

Not only we need to conceal information, we also need to practice image processing and optimization techniques. On the other hand, it also provides a challenge to think about how to detect the steganographic images without the original images.

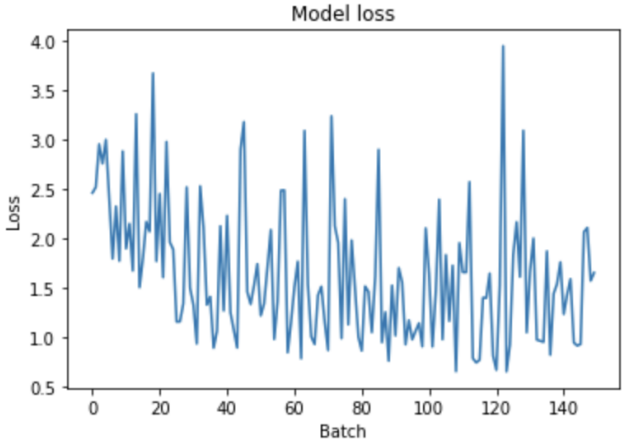
## 3 Task1—LSB

LSB-Steganography: LSB-Steg module is based on OpenCV to hide data in images. It uses the first bit of every pixel, and every color of an image. The code is quite simple to understand; If every first bit has been used, the module starts using the second bit, so the larger the data, the more the image is altered. The program can hide all of the data if there is enough space in the image. The main functions are:
1. encode text: You provide a string and the program hides it.
2. encode image: You provide an OpenCV image and the method iterates for every pixel in order to hide them. A good practice is to have a carrier 8 times bigger than the image to hide (so that each pixel will be put only in the first bit).
3. encode binary: You provide a binary file to hide; This method can obfuscate

any kind of file.
What we got on the test

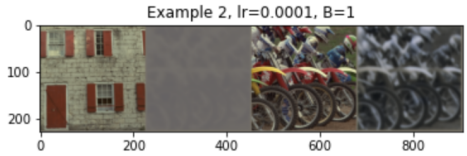
# 4 Task2–NN

Deep neural networks are simultaneously trained to create the hiding and revealing processes and are designed to specifically work as a pair. The system is trained on images drawn randomly from the ImageNet database, and works well on natural images from a wide variety of sources. The loss diagram is:

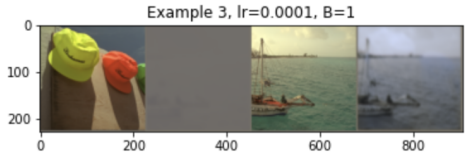


Along with the increasing of the training data, the result is getting better. I tried the datasets from https://www.kaggle.com/gaz3ll3/optimization-ii-project-3; With more involvement of different pictures from different categories, the trained model will be better. Therefore, I believe that when the dataset is more sufficient and various, the neural network will have better performance.

```
Total loss: 0.69
Loss on secret: 0.43
Loss on cover: 0.26
```




```
Total loss: 1.09
Loss on secret: 0.87
Loss on cover: 0.22
```




```
Average loss on test set: 1.05
```

# 5 Reference

https://githubmemory.com/repo/ayu1211/LSB-Steganography
https://github.com/fpingham/DeepSteg/blob/master/DeepSteganography.ipynb
https://github.com/kelvins/steganography