Math 110B Project 3: Steganography

Yunzhe Tang

Description

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. This project is aimed to use LSB and CNN to implement steganography on images.

Steganography contains two processes: encryption and decryption. Since both processes have a loss, our goal is to minimize the function:

$$||x - E(x, y)|| + \gamma||y - D(z)||$$

where x is the original image, y is the secret image, E is the encryption function, D is the decryption function, z = E(x,y). Images used in this project are composed of 256x256 pixels and each pixel contains 3 channels: R,G,B, each of them is a 8-bit integer, going from 0 to 255. Therefore, we can transfer it to an optimization problem in Mathematics.

Data Set

The training data for CNN are 120 256x256 images in the folder 'forest' from the data set in Kaggle. Epoch is set to be 3.

The testing data for CNN are 24 random sized images from the website http://r0k.us/graphics/kodak/.

LSB

Since each channel (red, green, blue) in a pixel can be represented as an 8-bit binary digit, the importance of a number decreases from left to right. The leftmost one is of the highest significance. For example, changing from 11111111 to 01111111 will change the decimal value from 255 to 127. Whereas changing from 11111110 to 11111111 will only change the decimal value from 254 to 255.

Our idea of merging is to replace the rightmost 4 bits of an original image by the leftmost 4 bits of a secret image. In order to unmerge an image, we pick the rightmost 4 bits and add '0000' to the right.

However, both processes will have a loss. If we compare the original image and merged image or the secret image and the unmerged image carefully, we can find the difference, like a loss in colors.

CNN

This method consists of three networks: preparing, hiding and revealing network. It requires the trained system must learn to compress the information from the secret image into the least noticeable portions of the cover image.

Prep-network is to prepare the secret images to be hidden. Say we want to hide an MxM sized secret image into an NxN sized cover image. We need to increase the size of the secret one by distributing the secret image's bits across the entire NxN pixels.

Hiding-network is the main network. The input is an NxN pixel field, with depth concatenated RGB channels of the cover image and the transformed channels of the secret image. And the best choice is consisted of 5 convolution layers that have 50 filters each of {3 × 3, 4 × 4, 5 × 5} patches.

Reveal-network receives the container image from the previous network and decode it by removing the cover image to reveal the secret image.

Result

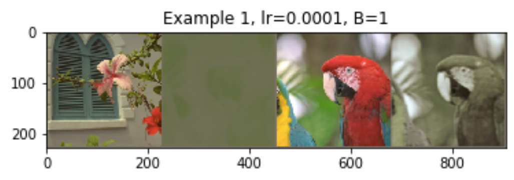
For CNN:

Training performances:

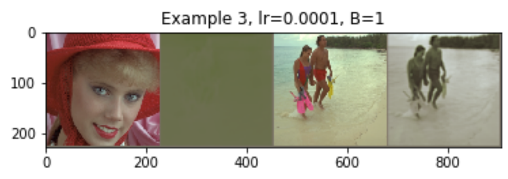
```
Epoch [1/3], Average_loss: 1.7041

Epoch [2/3], Average_loss: 1.2869

Epoch [3/3], Average_loss: 1.1895
```

The first 4 performances for testing data:

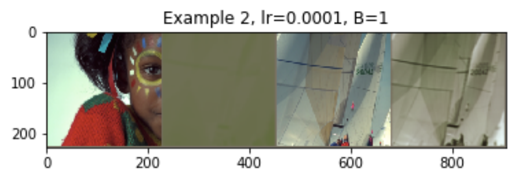
```
Total loss: 0.79
Loss on secret: 0.46
Loss on cover: 0.33
```



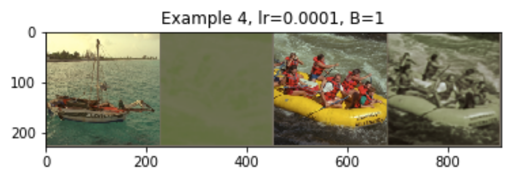
```
Total loss: 0.63
Loss on secret: 0.54
Loss on cover: 0.09
```



```
Total loss: 2.05
Loss on secret: 1.99
Loss on cover: 0.06
```



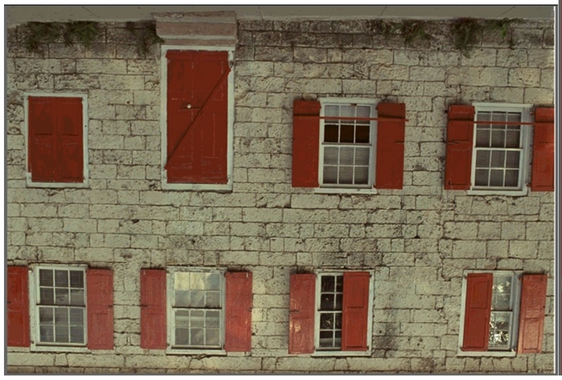
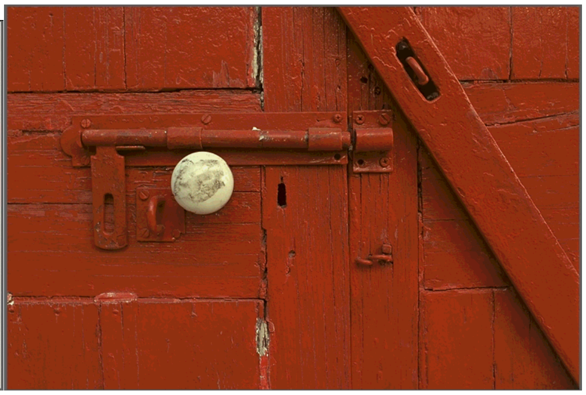
```
Total loss: 1.49
Loss on secret: 1.22
Loss on cover: 0.26
```



And

```
Average loss on test set: 1.17
```

For LSB:

Cover image Secret image

Merged image Unmerged image

Reference

https://papers.nips.cc/paper/6802-hiding-images-in-plain-sight-deep-steganography.pdf

https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1

https://github.com/fpingham/DeepSteg/blob/master/DeepSteganography.ipynb