110B-Final-Project

Group member:

Fangyang Zhang Junyu Sui Jiaxin Li

Task1 LSB:

LSB short of Least square Byte.

The main idea if LSB is clearly explained in https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1

A short Summary:

For general image, each pixel has RGB channel, which has 3 values on each channel. And the range for each channel is 0-255. Then we can turn the value into binary form which requires 8 bytes. if we change the last 4 bytes, then the changes is at most 1+2+4+8=15, which is relative small for a value ranging from 0 to 255. So we can store secret information in these last 4 bytes for each channel in each pixel. And we can take first 4 bytes for each channel in each pixel of the secret image to retain as much information as possible.

The ideas is from the code on the github

https://github.com/kelvins/steganography/blob/main/steganography.py

Task2 NN:

The Solution -A Neural Network

Convolutional Neural Networks have shown to learn structures that correspond tological features. These features increase their level of

abstraction as we go deeper into the network. Using a ConvNet will solve all the problems mentioned above. Firstly, the convnet will have a good idea about the patterns of natural images, and will be able to make decisions on which areas are redundant, and more pixels can be hidden there. By saving space on redundant areas, then a mount of hidden information can be increased. Because the architecture and the weights can be randomised, the exact way in which the network will hide the information cannot be known to anybody who doesn't have the weights.

Https://buzzrobot.com/hiding-images-using-ai-deep-steganography-b7726bd58b06

The idea is from the code:

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

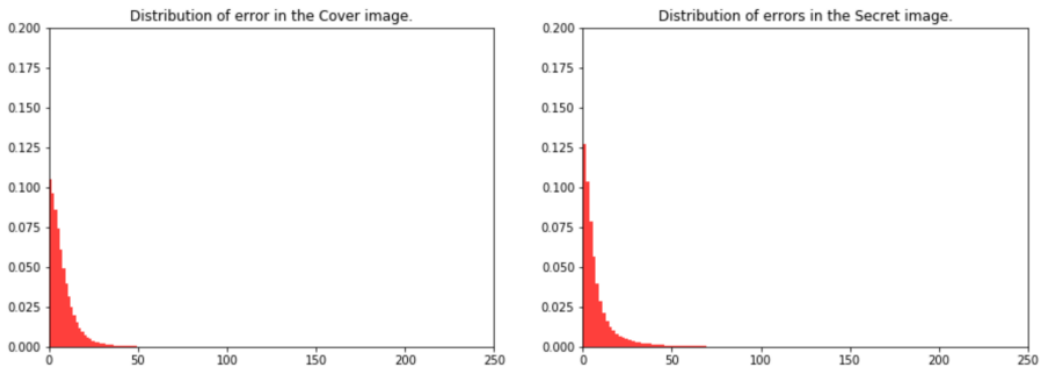
Combine DCT with Neural Network:

Hint: In the following sections, we use different version of keras maybe you can use following code to solve error:

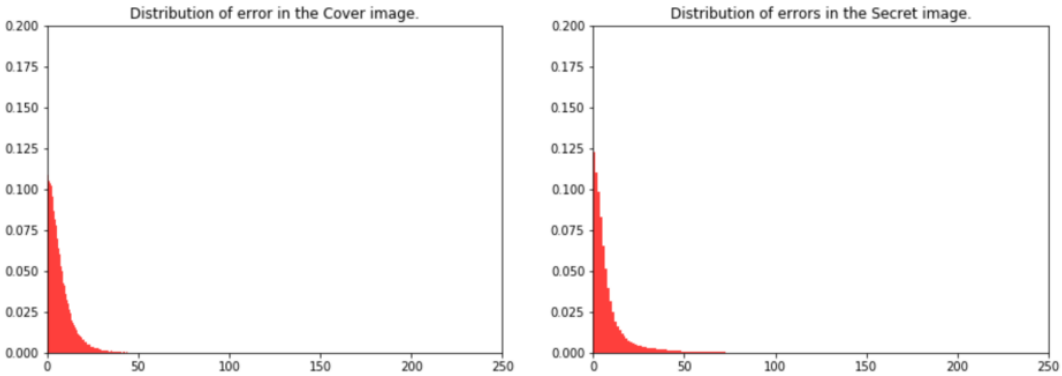
pip install keras==2.2.4

In the proposed original structure, there is a preparation layers for image compression. Here, a discrete cosine transform is used to replace the preparation network. The algorithm discards the coefficients in the lower half, and then transforms the image back to its original domain. In order to increase frequency we use tiny-imagenet-200 in the https://tiny-imagenet.herokuapp.com/ for 64x64 small images.

We get the results on the training set through the error density histogram.

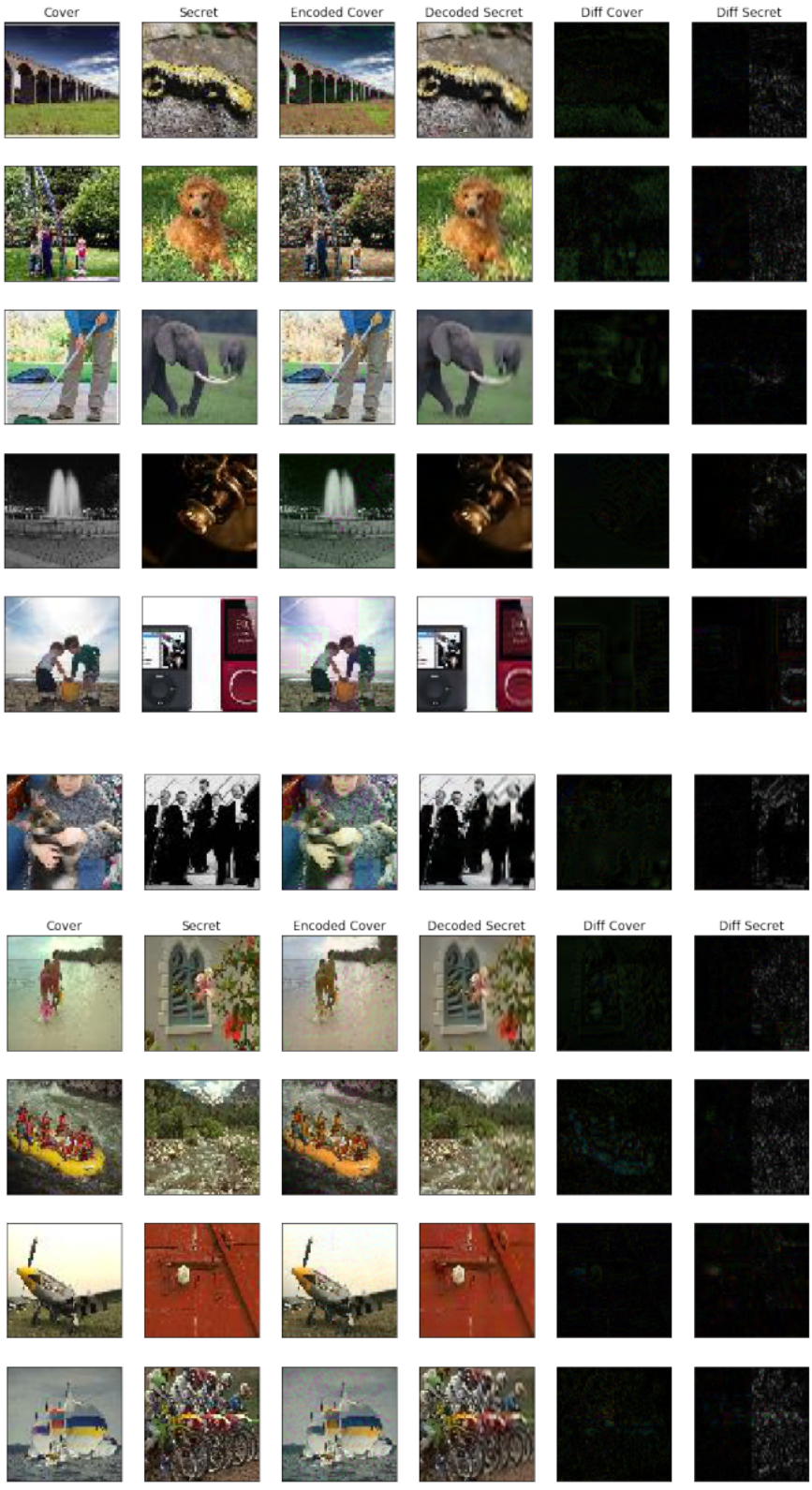


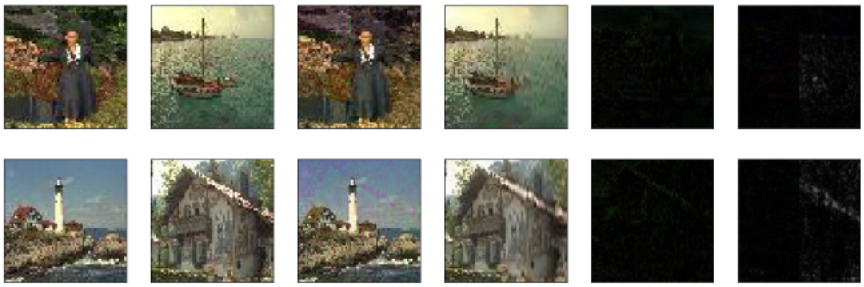
Error density histogram on test data set: e = 13.58 e = 9.56



Although DCT compression has no obvious effect on the image itself to the human eye, by discarding the information of some specific frequencies, CNN seems to use some pattern information on the carrier image to hide the secret image, which leads to some more obvious pattern and color offsets. Although there are some similar errors compared with the previous

settings. The following are sample images from the training and test data sets, respectively.



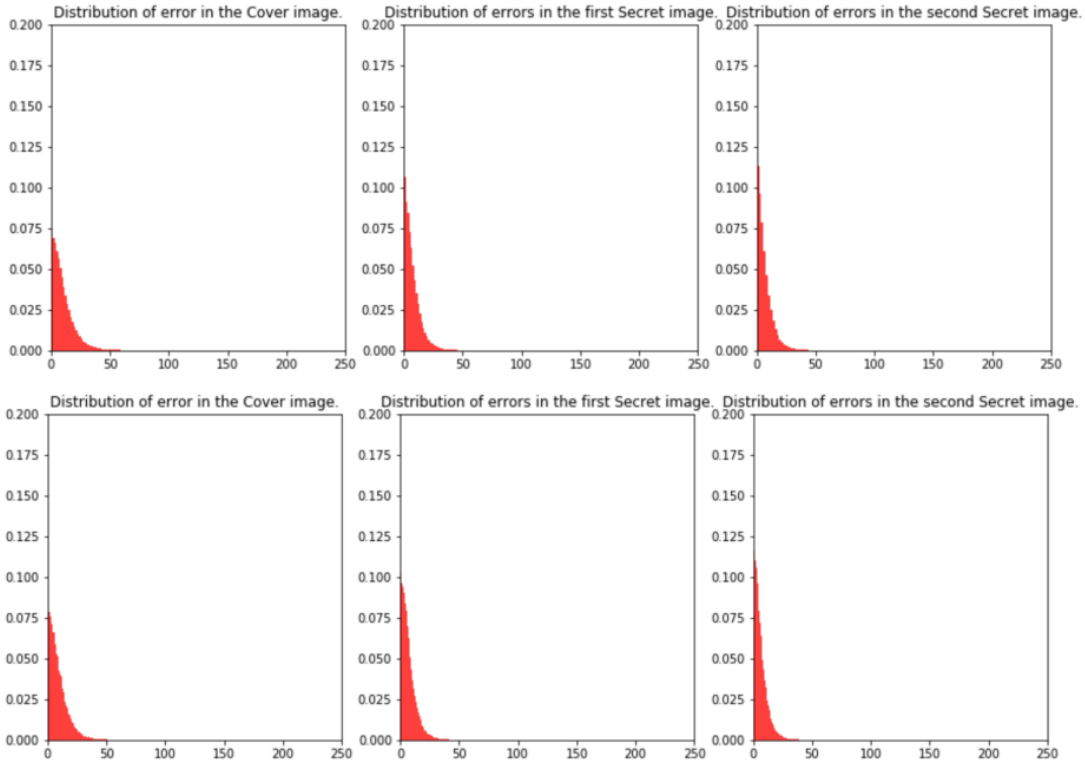


Optional Task:

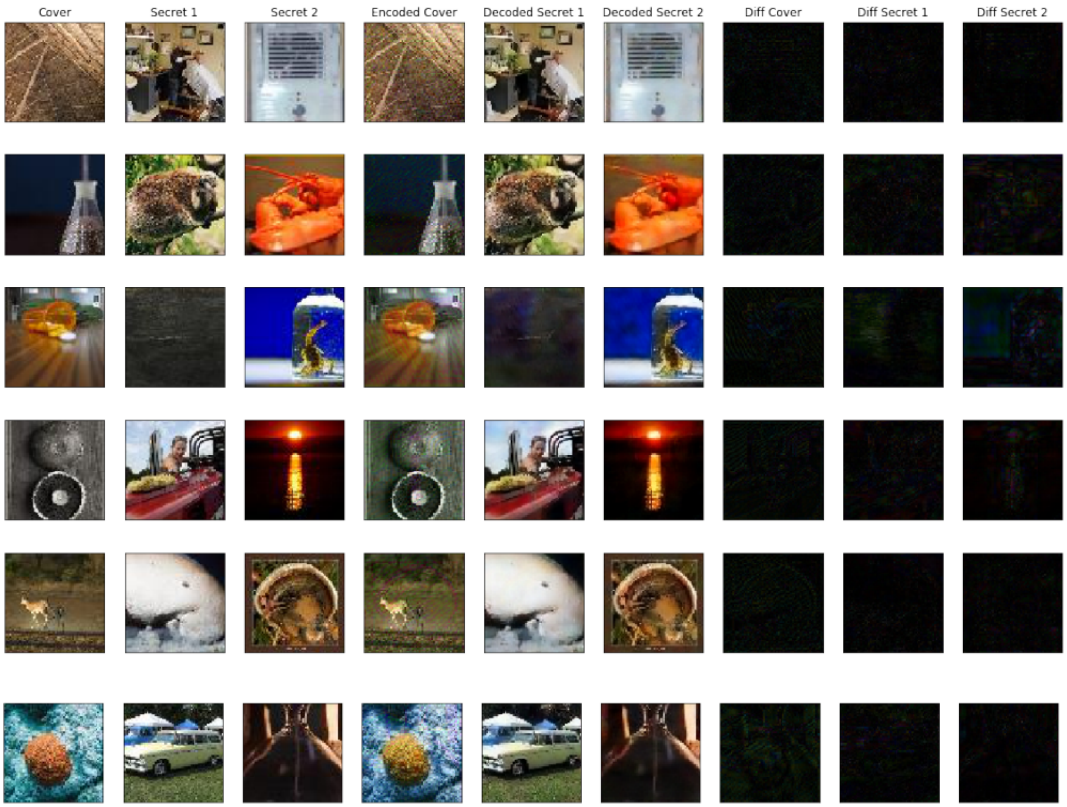
The main idea is to connect two secret images in series along the color channel axis, and input the two images together with the cover image into the network as a 6-channel secret image. The depth of the network is also increased to make up for the difficulty of the task. Then, the loss function naturally becomes the sum of the squares of the errors of the two secret images multiplied by a parameter. Here are some test settings and their respective errors.

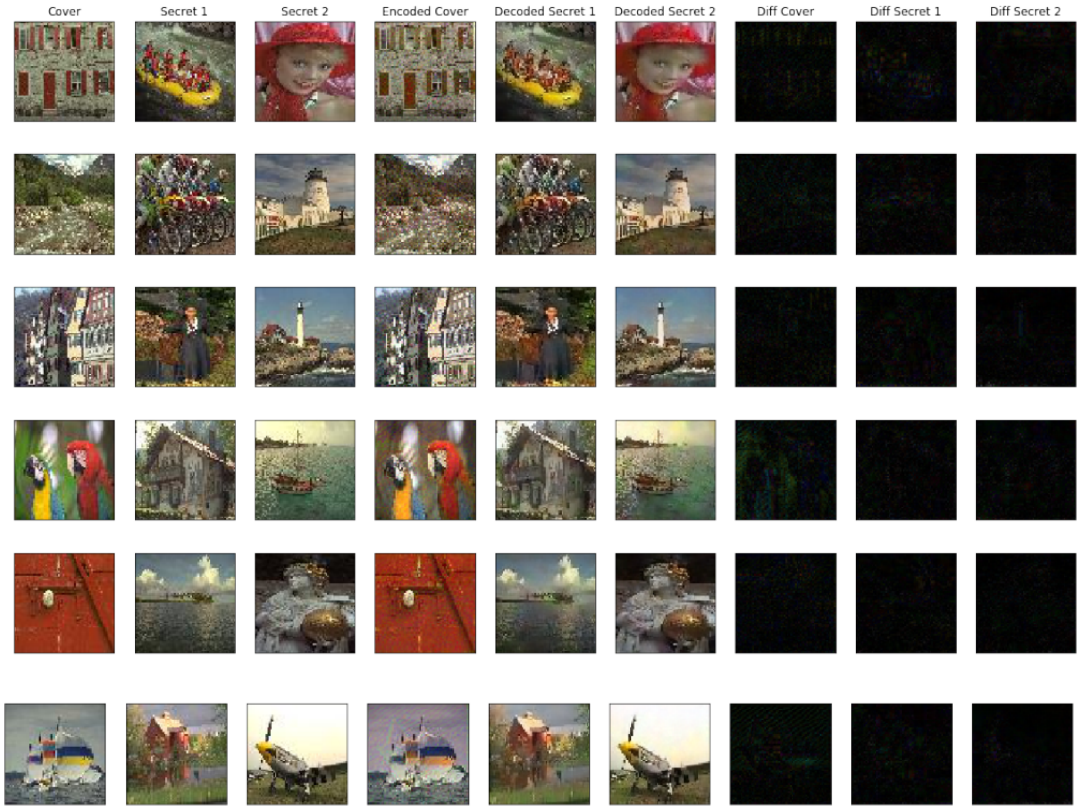
Preparation block	Hiding block	Reveal block	β	$E_{s1}/E_{s2}/E_c$	
1	2.0	2	8	8	12.7/13.2/15.9
2	2.0	2	11	11	13.3/13.2/14.8
3	2.0	4	8	8	8.0/7.7/15.9
4	1.0	4	8	8	10.0/9.5/13.8
5	0.5	4	8	8	14.8/14.7/10.9
6	0.75	4	8	8	12.4/13.1/12.6

Although the performance of hiding two pictures is always worse than hiding one picture, we still get some relatively good results. The following is the detailed results of setting 4, which is considered to be the best of all 64 settings. In the training data set, in the test data set. The error density histogram of training and testing is as follows.



Some sample images on the training and test data set.





Some Future Work:

For LSB, due to the limited number of bits available, it is difficult to hide more than one image of the same size and color depth. However, after DCT, it may be possible to store the most important information of the secret image with fewer bits. However, since the coefficients are floating numbers, it is necessary to convert the coefficients into small binary numbers in a clever way.

For NN, We just use 60 pictures as the train set to increase frequency and save time. I trust that if our computer is fast enough, we can use lots of photos as the training set, so as to increase the accuracy of the model and reduce the loss.