Steganography: Math 110B Project 3 Writeup

Group Member: Yitan Ze, Junlin Wang, Yuan Fu

Introduction:

Given picture A, and a picture B that we want to hide, we want to use Least Significant Bits(LSB) and neural network to produce a picture C that secretly contains B's information, and we also want to extract B's info from C to a picture D. Thus, for our goal, pictures A and C should be similar, and pictures B and D should also be similar.

Method 1: Least Significant Bits(LSB)

In LSB, we replace the least significant bits of the original image(picture A) with the secret image's (picture B) most significant bits, so that the encoded picture C contains both the most important information from A and B.

We use Image class from Pillow Library to deal with image loading, accessing, constructing and saving. To encode the images, we first change each cell of the image to binary, then do the LSB conversion in binary state, and then change back to integer. To decode the encoded image, we first change each cell of the image to binary, extract the least four bits, add 0000 to the end, and then convert it back to integer. In this way, the decoded image will contain the most important information about the secret image.

Below is our result using the True Color Kodab Images Dataset. As you can see, the encoded image and the original image are very similar, and the decoded image perfectly contains most of the info in the secret image.

Group 1:

Picture A (original): kodim01.png	Picture B (secret): kodim02.png
	
Picture C (encoded):output3.png	Picture D (decoded):output4.png

Group 2:

Picture A (original): kodim03.png	Picture B (secret): kodim05.png
	
Picture C (encoded):output5.png	Picture D (decoded):output6.png
	

Group 3:

Picture A (original): kodim03.png	Picture B (secret): kodim06.png

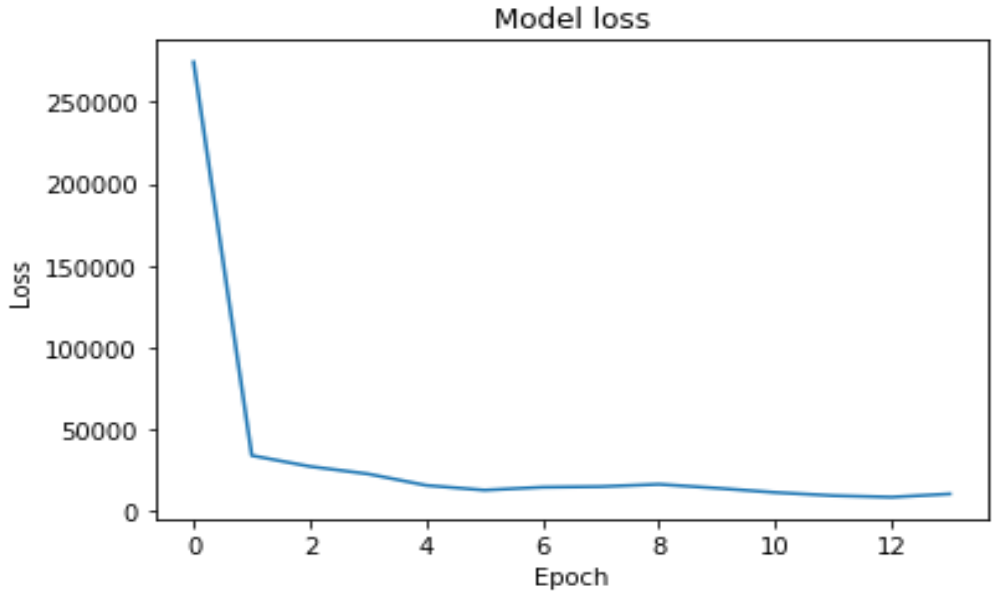
	
Picture C (encoded):output7.png	Picture D (decoded):output8.png
	

Method 2:Neural Network

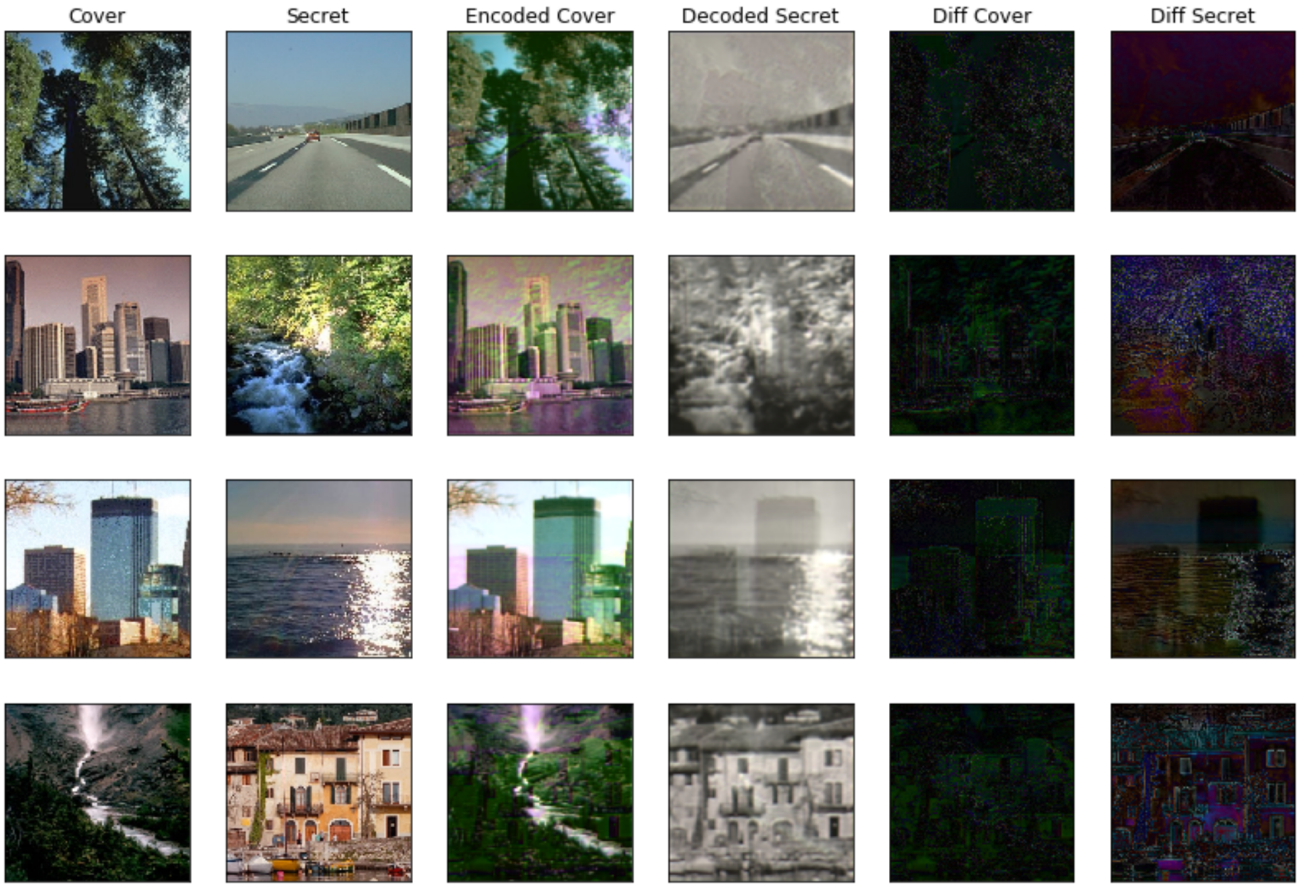
Using neural network to approximate D and E with tensorflow. Firstly, we build a function to calculate the loss of the full model, which can be used for preparation and hiding networks. The design of the loss function is illustrated by $\min_{E,D} \|x - E(x,y)\| + \gamma\|y - D(z)\|$ given in the paper. We constructed one encoder and one decoder network using CNN layers and fully connected layers as advised in the paper. The prep network is embodied in the encoder because essentially we are using the same error (hiding loss) to train both.
Difficulties:

1) We faced substantial difficulties when constructing both networks and running both networks. We first tried Pytorch and attempted to train the network. But torch arithmetics and loss backtracking were very difficult to debug to make it correct. Thus the result turned out unfruitful. We then tried tensorflow/keras and thankfully we don't need to handle back propagation explitely. This makes the process so much easier and we were on the right track
2) The computational cost was way too high to get a decent model. So we have to settle for a simple one. We trained the model using 40 images and 15 epochs.

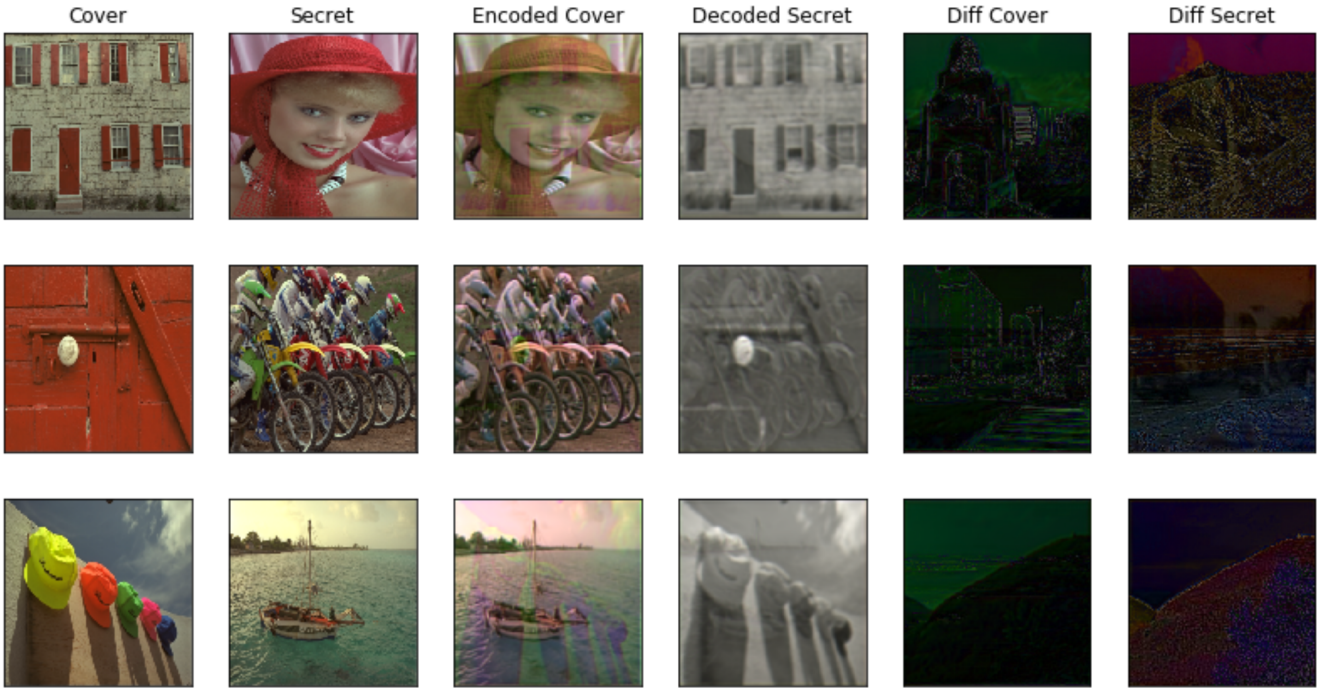
Result:



1. Testing:



2. Testing with True Color Kodab Images Dataset:



As you can see, the result does make sense and we are definitely doing it correctly. However, due to time issues, we have to settle for a worse model.