Title: Steganography

Date: 09/08/2021

Group members: Jiaqi Lin

Task 1. Implement LSB

Work I did

I implemented the LSB algorithm using the Python keras.preprocessing.image library to read the images, replacing the low bits of cover images with the high bits of secret images, and using the matplotlib.pyplot library to plot the encoded and decoded images.

Issues that I faced

The one issue I faced is how to read the images, convert them into numpy array, and plot the image using the numpy array. I googled a lot and checked the given Github repositories in the project description. At the end, I used the keras.preprocessing.image to read the image to numpy array and use the pyplot. imshow () to plot numpy as images.

Performance

I tested the LSB algorithm against the data set http://r0k.us/graphics/kodak/, I used the first half of the data set (i.e., first 12 images) as secret images and the last half of the data set (i.e., last 12 images) as cover images. Then, for the i_{th} secret image, I cover it in the i_{th} cover images. For example, for the first secret image (i.e., image kodim01.png), I cover it in the first cover image (i.e., image kodim13.png). Figure 1 shows the result of covering image kodim01.png in image kodim13.png.

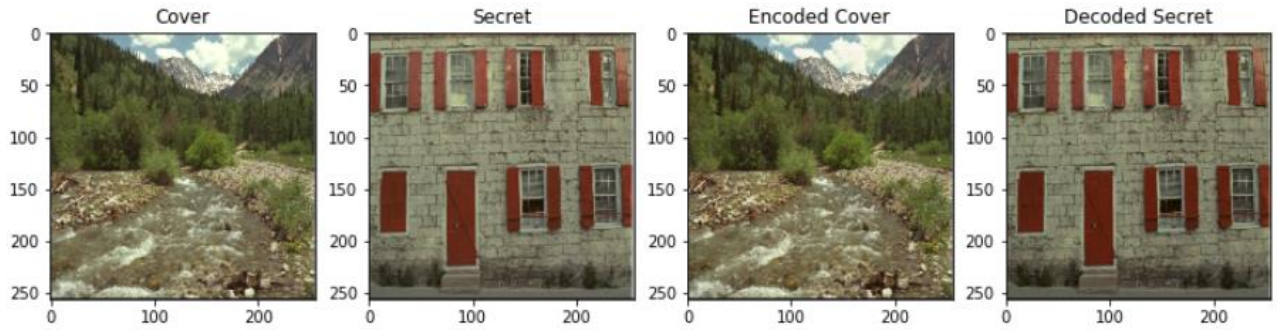

Figure 1

- In the last cell in file Steganography_task1.ipynb, I outputted the results for each image pair. The loss of the LBS in encoding the images and decoding the images is shown in Table 1. There are three types of losses: encoding losses, decoding losses, and total losses. The encoding loss is the **root mean squared error (RMSE)** between the cover image and the encoded cover image. The decoding loss is the RMSE between the secret image and the decoded secret image. The total loss is the sum of the decoding less and encoding loss.

Table 1

Secret image	Cover image	Encoding loss	Decoding loss	Total loss
kodim01.png	`kodim13.png`	0.03	0.03	0.06
kodim02.png	`kodim14.png`	0.03	0.03	0.06
kodim03.png	`kodim15.png`	0.03	0.03	0.06

kodim04.png	kodim16.png	0.03	0.03	0.06
kodim05.png	kodim17.png	0.03	0.03	0.06
kodim06.png	kodim18.png	0.02	0.04	0.06
kodim07.png	Kodim19.png	0.03	0.03	0.06
kodim08.png	Kodim20.png	0.03	0.04	0.07
kodim09.png	Kodim21.png	0.02	0.03	0.05
Kodim10.png	Kodim22.png	0.02	0.05	0.05
Kodim11.png	Kodim23.png	0.02	0.04	0.06
Kodim12.png	Kodim24.png	0.02	0.03	0.05

The total loss is among 0.05 to 0.07.

Task 1. Implement Neural Network

Work I did

I implemented a convolutional neural network based on the paper, i.e., "**Hiding Images in Plain Sight: Deep Steganography**". The neural network has three parts, the pre-processing network, the encoder, and the decoder. There are 2 layers of 65 filters in the pre-processing network, 3 layers of 65 filters in the encoder, and 3 layers of 65 filters in the decoder. I used 65 filters because 65 is used in most of the implementation GitHub repositories given in the project description. I also added noise to the encoder output before the decoder to ensure that the networks do not simply encode the secret image in the LSB. In total, there are 488,661 parameters in the neural network.

Training data set:

80% of the images from https://www.kaggle.com/gaz3ll3/optimization-ii-project-3

Validation data set:

20% of the images from https://www.kaggle.com/gaz3ll3/optimization-ii-project-3

Testing data set:

the data set: http://r0k.us/graphics/kodak/

Issues that I faced

The most difficult issue that I faced is that I couldn't not train the neural network on my PC because the neural network is too big and my pc does not have enough memory. Thus, I have to ask my friends to run the training script (seen the deep_steg.py file in the repository) on their server. For their privacy, I cannot share their information. Their sever is using SLURM to schedule all the jobs. In the fl.sh file in the repository, I included the script to create virtual python environment, specify the GPU, CPU, and memory resources, and submit the neural network training script.

- After training the network on the server, I downloaded the trained weights of the network, the training losses, and the validation losses from the server. There are three types of losses: encoding losses, decoding losses, and total losses. The encoding loss is the RMSE between the cover image and

the encoded cover image. The decoding loss is the RMSE between the secret image and the decoded secret image. The total loss is the sum of the decoding less and encoding loss.

The downloaded losses can be found in the models/*.txt files. Figures 2 and 3 show the training losses and validation losses. Because on the time limitation, I only trained the neural network for 20 epochs. I think the losses will be reduced if I increase the number of epochs.

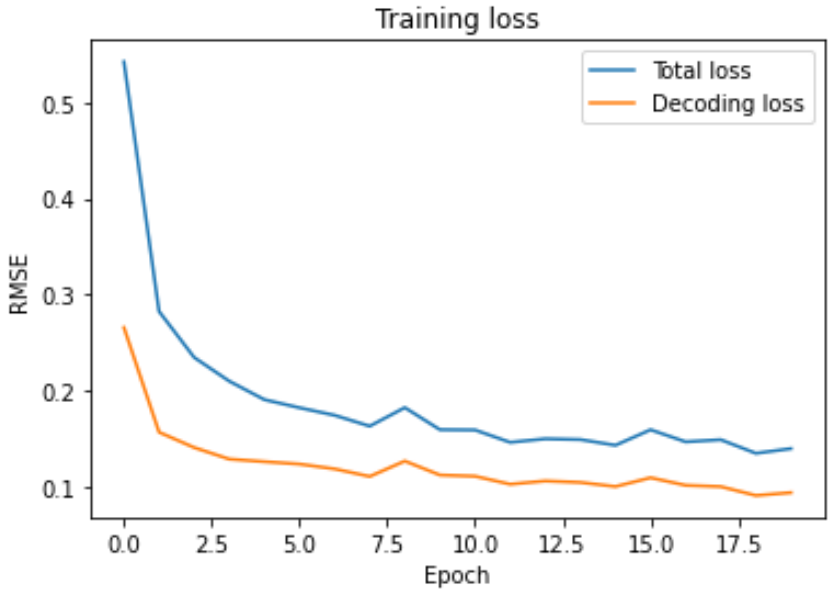

Figure 2

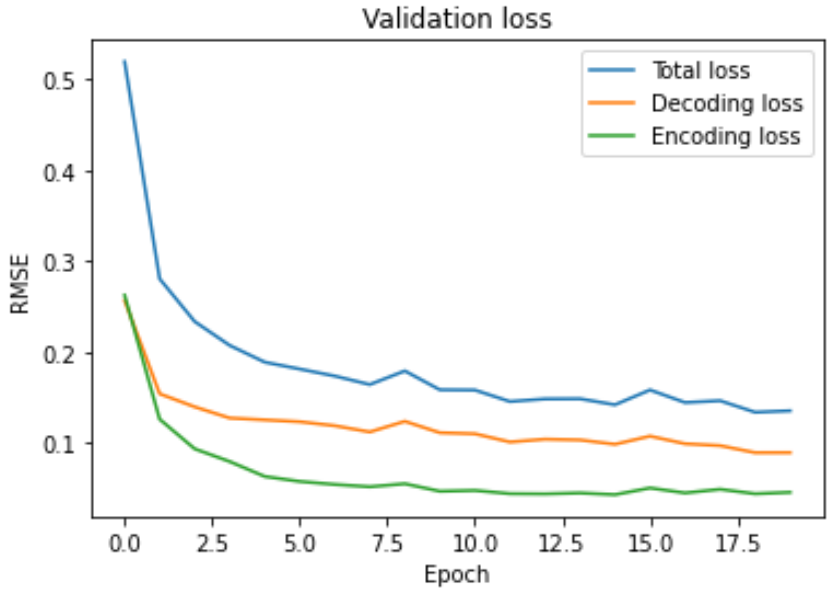

Figure 3

Performance

Based on the trained weights of the neural network downloaded from server (weights can be found in the models/weights.hdf5 file), I can load the neural network on my PC and test the neural network on the testing data set.

As explain above, I tested the neural network on the data set http://r0k.us/graphics/kodak/, I used the first half of the data set (i.e., first 12 images) as secret images and the last half of the data set (i.e., last 12 images) as cover images. Then, for the i_{th} secret image, I cover it in the i_{th} cover images. Figure 4 shows the result of covering image kodim01.png in image kodim13.png.

Figure 4

- In the last cell in file Steganography_task2.ipynb, I outputted the results for each image pair. The loss of the neural network is shown in Table 2.

Table 2

Secret image	Cover image	Encoding loss	Decoding loss	Total loss
kodim01.png	kodim13.png	0.04	0.09	0.13
kodim02.png	kodim14.png	0.05	0.15	0.20
kodim03.png	kodim15.png	0.06	0.11	0.17
kodim04.png	kodim16.png	0.03	0.11	0.14
kodim05.png	kodim17.png	0.03	0.09	0.12
kodim06.png	kodim18.png	0.04	0.08	0.12
kodim07.png	Kodim19.png	0.03	0.08	0.11
kodim08.png	Kodim20.png	0.04	0.10	0.14
kodim09.png	Kodim21.png	0.03	0.07	0.10
Kodim10.png	Kodim22.png	0.04	0.06	0.10
Kodim11.png	Kodim23.png	0.07	0.08	0.15
Kodim12.png	Kodim24.png	0.04	0.08	0.12

Analyze the performance

Based on the losses and the decoded secret and encoded cover images, I can tell that the neural network is performing worse than the simple LBS algorithm. I think there are several reasons behind the poor performance of the neural network:

- Not enough training data. We have only 2,150 images in the training data set. But there are 488,661 parameters in the neural network need to be trained. The images are just not enough to train these half a million parameters.
- Didn't train the neural network well enough. I only trained the neural network for 20 epochs. I believe if I train the neural network for more epochs, the neural network could have better performance.