Final project: Steganography
Group member: Yanxin Wang, Qingcheng Zhang, Zhaoyue Wang

Task1: In this task,we use the code which given on github:
https://github.com/kelvins/steganography

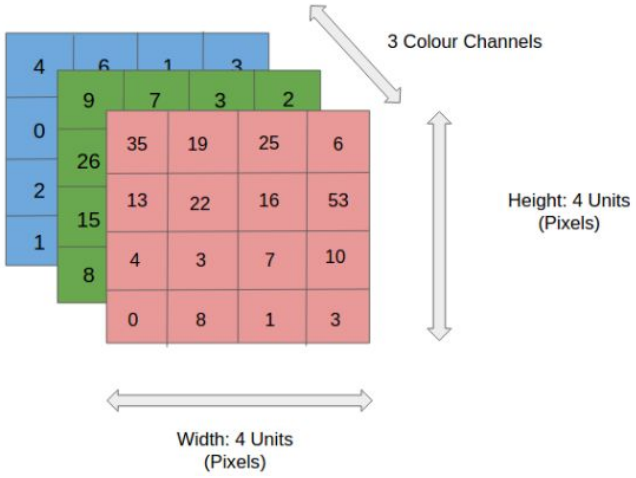
In this task, we understand how to use python to implement LSB(LSB is called least significant bits, which means you will replace the least significant bits of the original image with the secret image's most significant bits), and make the images (original and secret) are of the same sizes. First of all, we understand what the pixel(the smallest element is an image) is and We split the image into three channels: red, green, and blue. So, each pixel has three channels, and each channel is 8-bits sequence. For 8-bits calculation, the last 4 digits do not influence too much.

	128	64	32	16	8	4	2	1
8 bit binary digit	1	0	1	1	0	0	0	1
	128 + 32 + 16 + 1 = 177							

So, we change the original image's last four digits into the hidden image's four digits to complete the requirements of steganography.

Task2: In this task, we use the code which given on github:
https://github.com/fpingham/DeepSteg/blob/master/DeepSteganography.ipynb

There are 5 layers in the method of CNN, the Convolutional Layer, Pooling Layer, Normalization Layer, Fully-Connected Layer and Converting Fully-Connected Layers to Convolutional Layer.
In this case l, we take an RGB image which has been separated by its three color planes — Red, Green, and Blue, as an input.



The goal of the CNN is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.
The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting.
Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. This is useful for extracting dominant features which are rotational and positional invariant, while maintaining the process of effectively training of the model.
The Fully-Connected layer is learning a possibly non-linear function in that space.

In this task, the biggest problem for us is to run the code and find the paraments. For this code, we use GPU computing. It is significantly faster than CPU. If we use CPU, 5 data set only have 870 batch, it will use 18 hours to finish running(only for training part). But if we use GPU, it will shorten time 30 times, it will only take 30 mins for 5 data set.

Because of the big data set, we want to minimize the data set, we want to minimize the data set. But the influence is very small. Total number of iterations(batch size is 2) does not have big difference with smaller dataset.

1.Number of epochs:
If only have one epoch, the loss will be big. But if we have 3 or 5 or bigger epoches, the performances are the same. Even though we run 50 epochs, loss is the same as 3 epochs.

```
Training: Batch 321/341. Loss of 0.7577, cover loss of 0.0150, secret loss of 0.7427
Training: Batch 322/341. Loss of 1.5407, cover loss of 0.0099, secret loss of 1.5309
Training: Batch 323/341. Loss of 1.1648, cover loss of 0.0175, secret loss of 1.1473
Training: Batch 324/341. Loss of 2.2294, cover loss of 0.0074, secret loss of 2.2221
Training: Batch 325/341. Loss of 1.2714, cover loss of 0.0169, secret loss of 1.2545
Training: Batch 326/341. Loss of 1.0754, cover loss of 0.0199, secret loss of 1.0554
Training: Batch 327/341. Loss of 1.2851, cover loss of 0.0265, secret loss of 1.2585
Training: Batch 328/341. Loss of 1.0854, cover loss of 0.0078, secret loss of 1.0776
Training: Batch 329/341. Loss of 1.7912, cover loss of 0.0294, secret loss of 1.7618
Training: Batch 330/341. Loss of 0.7136, cover loss of 0.0239, secret loss of 0.6897
Training: Batch 331/341. Loss of 1.0583, cover loss of 0.0146, secret loss of 1.0437
Training: Batch 332/341. Loss of 1.7779, cover loss of 0.0172, secret loss of 1.7607
Training: Batch 333/341. Loss of 0.8513, cover loss of 0.0060, secret loss of 0.8453
Training: Batch 334/341. Loss of 1.4013, cover loss of 0.0102, secret loss of 1.3911
Training: Batch 335/341. Loss of 1.0252, cover loss of 0.0143, secret loss of 1.0109
Training: Batch 336/341. Loss of 0.7235, cover loss of 0.0169, secret loss of 0.7066
Training: Batch 337/341. Loss of 1.2230, cover loss of 0.0158, secret loss of 1.2072
Training: Batch 338/341. Loss of 2.3297, cover loss of 0.0251, secret loss of 2.3046
Training: Batch 339/341. Loss of 0.9144, cover loss of 0.0148, secret loss of 0.8996
Training: Batch 340/341. Loss of 1.4749, cover loss of 0.0380, secret loss of 1.4369
Training: Batch 341/341. Loss of 1.5751, cover loss of 0.0148, secret loss of 1.5603
Epoch [3/3], Average_loss: 1.3278
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 0.68
Loss on secret: 0.65
Loss on cover: 0.03
Total loss: 1.24
Loss on secret: 1.23
Loss on cover: 0.01
Total loss: 2.48
Loss on secret: 2.44
Loss on cover: 0.04
Total loss: 1.49
Loss on secret: 1.48
Loss on cover: 0.01
Average loss on test set: 1.30
```

The above picture shows that decrease the total number of batches smaller do not have influence on the performance.

2.Learning rate:
0.0001 and 0.000146 have the same result. 0.0005's loss is a little bigger than the previous two choices. These 3 learning rates are the popular choices of CNN.

3.Beta:

$$\mathcal{L}(c, c', s, s') = ||\mathbf{c} - \mathbf{c}'|| + \beta||\mathbf{s} - \mathbf{s}'||$$

The smaller the beta, the loss is smaller.

4.Activation Function:
Since sigmoid tangent h are not popular choice, so we tried activated function related with rectified linear unit. But dying relu problem is a problem. "Unfortunately, ReLU units can be

fragile during training and can ‘die’. For example, a large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any datapoint again. If this happens, then the gradient flowing through the unit will forever be zero from that point on. That is, the ReLU units can irreversibly die during training since they can get knocked off the data manifold. For example, you may find that as much as 40% of your network can be "dead" (i.e. neurons that never activate across the entire training dataset) if the learning rate is set too high. With a proper setting of the learning rate this is less frequently an issue."(http://cs231n.github.io/neural-networks-1/#actfun)
So we also choose leaky relu(Leaky ReLUs are one attempt to fix the “dying ReLU” problem. Instead of the function being zero when x < 0, a leaky ReLU will instead have a small negative slope (of 0.01, or so). That is, the function computes $f(x)=1(x<0)(\alpha x)+1(x>=0)(x)f(x)=1(x<0)(\alpha x)+1(x>=0)(x)$ where α is a small constant(http://cs231n.github.io/neural-networks-1/)) and randomize relu(Randomized ReLU (RReLU) is a randomized version of Leaky ReLU, where the α is a random number. In RReLU, the slopes of negative parts are randomized in a given range in the training, and then fixed in the testing. It is reported that RReLU could reduce overfitting due to its randomized nature in the Kaggle National Data Science Bowl (NDSB) competition(https://isaacchanghau.github.io/post/activation_functions/)).
For the author’s code(https://github.com/fpingham/DeepSteg/blob/master/DeepSteganography.ipynb), the tset loss is 1.37, and the training loss is 1.33.

Epoch = 1, testing lass 1.4, training loss is 1.4399

```
Training: Batch 860/870. Loss of 1.0608, cover loss of 0.0711, secret loss of 0.9898
Training: Batch 861/870. Loss of 0.7820, cover loss of 0.0724, secret loss of 0.7096
Training: Batch 862/870. Loss of 0.8232, cover loss of 0.0204, secret loss of 0.8028
Training: Batch 863/870. Loss of 1.0961, cover loss of 0.0161, secret loss of 1.0801
Training: Batch 864/870. Loss of 1.0948, cover loss of 0.0380, secret loss of 1.0568
Training: Batch 865/870. Loss of 1.3199, cover loss of 0.0458, secret loss of 1.2742
Training: Batch 866/870. Loss of 1.3823, cover loss of 0.0247, secret loss of 1.3576
Training: Batch 867/870. Loss of 2.3242, cover loss of 0.0226, secret loss of 2.3016
Training: Batch 868/870. Loss of 0.7793, cover loss of 0.0290, secret loss of 0.7503
Training: Batch 869/870. Loss of 1.4547, cover loss of 0.0289, secret loss of 1.4257
Training: Batch 870/870. Loss of 1.0139, cover loss of 0.0132, secret loss of 1.0007
Epoch [1/1], Average_loss: 1.4399
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 1.29
Loss on secret: 1.13
Loss on cover: 0.16
Total loss: 1.67
Loss on secret: 1.59
Loss on cover: 0.08
Total loss: 2.36
Loss on secret: 2.33
Loss on cover: 0.03
Total loss: 1.38
Loss on secret: 1.35
Loss on cover: 0.03
Average loss on test set: 1.40
```

Epoch =3, testing lass 1.29, training loss is 1.31

```
Training: Batch 850/870. Loss of 0.6433, cover loss of 0.0043, secret loss of 0.6390
Training: Batch 851/870. Loss of 1.4826, cover loss of 0.0031, secret loss of 1.4795
Training: Batch 852/870. Loss of 1.1072, cover loss of 0.0042, secret loss of 1.1030
Training: Batch 853/870. Loss of 0.6779, cover loss of 0.0040, secret loss of 0.6739
Training: Batch 854/870. Loss of 2.2714, cover loss of 0.0042, secret loss of 2.2671
Training: Batch 855/870. Loss of 1.2985, cover loss of 0.0084, secret loss of 1.2901
Training: Batch 856/870. Loss of 1.1800, cover loss of 0.0127, secret loss of 1.1672
Training: Batch 857/870. Loss of 1.3025, cover loss of 0.0039, secret loss of 1.2986
Training: Batch 858/870. Loss of 1.6549, cover loss of 0.0039, secret loss of 1.6510
Training: Batch 859/870. Loss of 1.0353, cover loss of 0.0041, secret loss of 1.0312
Training: Batch 860/870. Loss of 1.0060, cover loss of 0.0023, secret loss of 1.0037
Training: Batch 861/870. Loss of 1.0824, cover loss of 0.0017, secret loss of 1.0808
Training: Batch 862/870. Loss of 0.6250, cover loss of 0.0026, secret loss of 0.6223
Training: Batch 863/870. Loss of 1.0465, cover loss of 0.0020, secret loss of 1.0445
Training: Batch 864/870. Loss of 1.4767, cover loss of 0.0172, secret loss of 1.4595
Training: Batch 865/870. Loss of 1.2822, cover loss of 0.0045, secret loss of 1.2778
Training: Batch 866/870. Loss of 2.0761, cover loss of 0.0026, secret loss of 2.0734
Training: Batch 867/870. Loss of 1.3348, cover loss of 0.0035, secret loss of 1.3313
Training: Batch 868/870. Loss of 0.7506, cover loss of 0.0079, secret loss of 0.7426
Training: Batch 869/870. Loss of 1.5056, cover loss of 0.0041, secret loss of 1.5015
Training: Batch 870/870. Loss of 0.8331, cover loss of 0.0031, secret loss of 0.8301
Epoch [3/3], Average_loss: 1.3146
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 1.69
Loss on secret: 1.67
Loss on cover: 0.01
Total loss: 1.68
Loss on secret: 1.68
Loss on cover: 0.00
Total loss: 1.28
Loss on secret: 1.26
Loss on cover: 0.02
Total loss: 1.79
Loss on secret: 1.78
Loss on cover: 0.00
Average loss on test set: 1.29
```

After epoch = 10, the training loss is the same, and the testing loss is slightly better.

```
Training: Batch 326/341. Loss of 1.9794, cover loss of 0.0006, secret loss of 1.9787
Training: Batch 327/341. Loss of 0.9884, cover loss of 0.0005, secret loss of 0.9879
Training: Batch 328/341. Loss of 1.2985, cover loss of 0.0004, secret loss of 1.2981
Training: Batch 329/341. Loss of 1.2826, cover loss of 0.0003, secret loss of 1.2822
Training: Batch 330/341. Loss of 1.1812, cover loss of 0.0003, secret loss of 1.1809
Training: Batch 331/341. Loss of 1.5190, cover loss of 0.0004, secret loss of 1.5186
Training: Batch 332/341. Loss of 0.7433, cover loss of 0.0003, secret loss of 0.7430
Training: Batch 333/341. Loss of 1.0871, cover loss of 0.0005, secret loss of 1.0867
Training: Batch 334/341. Loss of 1.3973, cover loss of 0.0003, secret loss of 1.3970
Training: Batch 335/341. Loss of 0.7229, cover loss of 0.0004, secret loss of 0.7224
Training: Batch 336/341. Loss of 0.9016, cover loss of 0.0002, secret loss of 0.9014
Training: Batch 337/341. Loss of 1.4415, cover loss of 0.0005, secret loss of 1.4410
Training: Batch 338/341. Loss of 1.1583, cover loss of 0.0004, secret loss of 1.1579
Training: Batch 339/341. Loss of 2.1787, cover loss of 0.0004, secret loss of 2.1782
Training: Batch 340/341. Loss of 0.8286, cover loss of 0.0004, secret loss of 0.8282
Training: Batch 341/341. Loss of 0.9900, cover loss of 0.0003, secret loss of 0.9898
Epoch [50/50], Average_loss: 1.3069
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 1.74
Loss on secret: 1.74
Loss on cover: 0.00
Total loss: 0.31
Loss on secret: 0.31
Loss on cover: 0.00
Total loss: 0.43
Loss on secret: 0.43
Loss on cover: 0.00
Total loss: 0.43
Loss on secret: 0.43
Loss on cover: 0.00
Average loss on test set: 1.23
```

Leaky Relu's performance is slightly better than relu.

```
Training: Batch 850/870. Loss of 0.6433, cover loss of 0.0043, secret loss of 0.6390
Training: Batch 851/870. Loss of 1.4826, cover loss of 0.0031, secret loss of 1.4795
Training: Batch 852/870. Loss of 1.1072, cover loss of 0.0042, secret loss of 1.1030
Training: Batch 853/870. Loss of 0.6779, cover loss of 0.0040, secret loss of 0.6739
Training: Batch 854/870. Loss of 2.2714, cover loss of 0.0042, secret loss of 2.2671
Training: Batch 855/870. Loss of 1.2985, cover loss of 0.0084, secret loss of 1.2901
Training: Batch 856/870. Loss of 1.1800, cover loss of 0.0127, secret loss of 1.1672
Training: Batch 857/870. Loss of 1.3025, cover loss of 0.0039, secret loss of 1.2986
Training: Batch 858/870. Loss of 1.6549, cover loss of 0.0039, secret loss of 1.6510
Training: Batch 859/870. Loss of 1.0353, cover loss of 0.0041, secret loss of 1.0312
Training: Batch 860/870. Loss of 1.0060, cover loss of 0.0023, secret loss of 1.0037
Training: Batch 861/870. Loss of 1.0824, cover loss of 0.0017, secret loss of 1.0808
Training: Batch 862/870. Loss of 0.6250, cover loss of 0.0026, secret loss of 0.6223
Training: Batch 863/870. Loss of 1.0465, cover loss of 0.0020, secret loss of 1.0445
Training: Batch 864/870. Loss of 1.4767, cover loss of 0.0172, secret loss of 1.4595
Training: Batch 865/870. Loss of 1.2822, cover loss of 0.0045, secret loss of 1.2778
Training: Batch 866/870. Loss of 2.0761, cover loss of 0.0026, secret loss of 2.0734
Training: Batch 867/870. Loss of 1.3348, cover loss of 0.0035, secret loss of 1.3313
Training: Batch 868/870. Loss of 0.7506, cover loss of 0.0079, secret loss of 0.7426
Training: Batch 869/870. Loss of 1.5056, cover loss of 0.0041, secret loss of 1.5015
Training: Batch 870/870. Loss of 0.8331, cover loss of 0.0031, secret loss of 0.8301
Epoch [3/3], Average_loss: 1.3146
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 1.69
Loss on secret: 1.67
Loss on cover: 0.01
Total loss: 1.68
Loss on secret: 1.68
Loss on cover: 0.00
Total loss: 1.28
Loss on secret: 1.26
Loss on cover: 0.02
Total loss: 1.79
Loss on secret: 1.78
Loss on cover: 0.00
Average loss on test set: 1.29
```

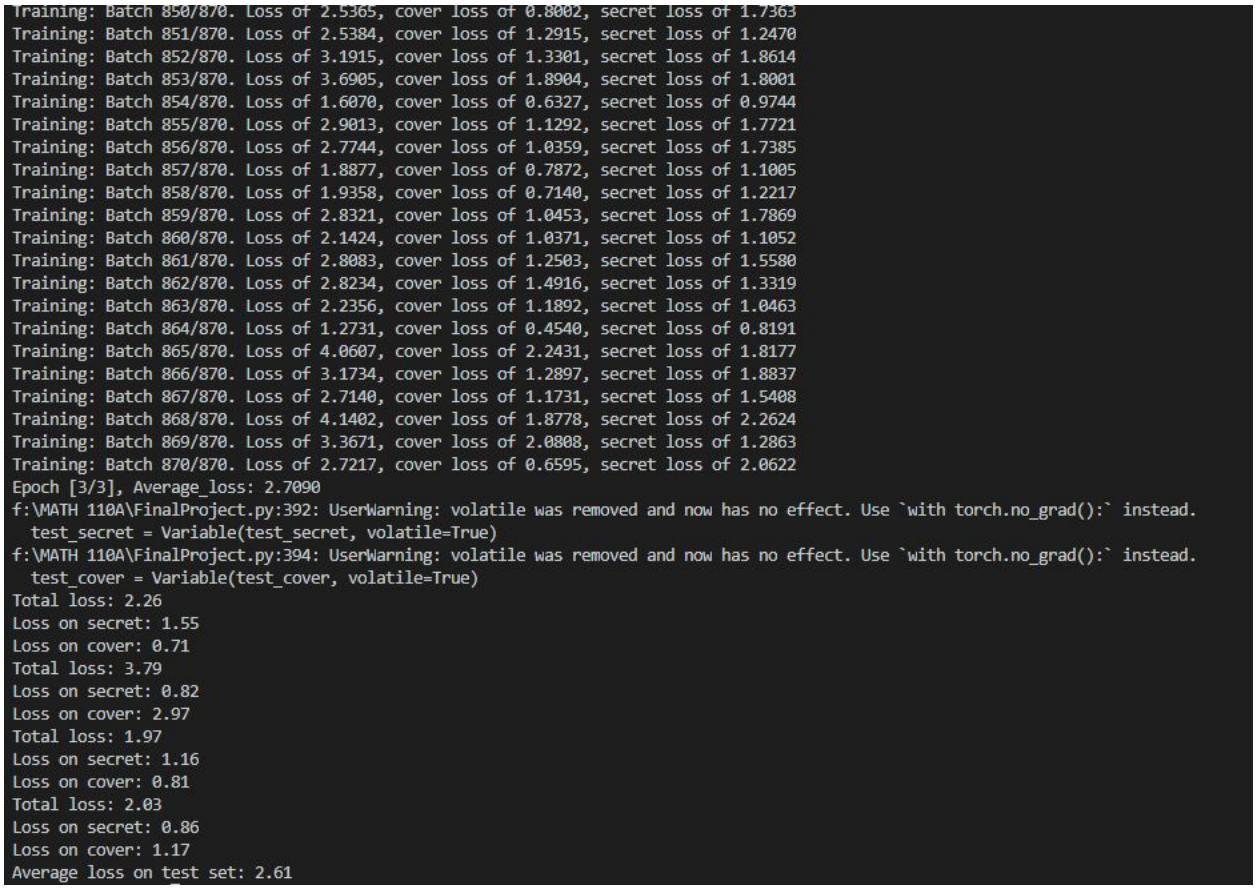
In our code, we use both leaky relu and relu to avoid the dying relu problem.

```
Training: Batch 331/341. Loss of 2.1895, cover loss of 0.0113, secret loss of 2.1782
Training: Batch 332/341. Loss of 1.2069, cover loss of 0.0111, secret loss of 1.1958
Training: Batch 333/341. Loss of 1.0616, cover loss of 0.0127, secret loss of 1.0489
Training: Batch 334/341. Loss of 1.9334, cover loss of 0.0167, secret loss of 1.9168
Training: Batch 335/341. Loss of 1.4376, cover loss of 0.0048, secret loss of 1.4327
Training: Batch 336/341. Loss of 1.3655, cover loss of 0.0169, secret loss of 1.3487
Training: Batch 337/341. Loss of 1.3151, cover loss of 0.0121, secret loss of 1.3030
Training: Batch 338/341. Loss of 1.1835, cover loss of 0.0038, secret loss of 1.1797
Training: Batch 339/341. Loss of 0.8800, cover loss of 0.0038, secret loss of 0.8762
Training: Batch 340/341. Loss of 0.7741, cover loss of 0.0125, secret loss of 0.7617
Training: Batch 341/341. Loss of 1.1066, cover loss of 0.0081, secret loss of 1.0985
Epoch [5/5], Average_loss: 1.3158
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 0.98
Loss on secret: 0.97
Loss on cover: 0.01
Total loss: 1.58
Loss on secret: 1.57
Loss on cover: 0.01
Total loss: 1.57
Loss on secret: 1.57
Loss on cover: 0.00
Total loss: 1.63
Loss on secret: 1.63
Loss on cover: 0.00
Average loss on test set: 1.22
```

Training loss is the same, test loss is better than relu. The loss is reduced by 6 to 8 percent.

4.Optimizer:

SGD: By using it, the loss is large.



Adam: (which the author uses) it the best.

```
Training: Batch 850/870. Loss of 0.6433, cover loss of 0.0043, secret loss of 0.6390
Training: Batch 851/870. Loss of 1.4826, cover loss of 0.0031, secret loss of 1.4795
Training: Batch 852/870. Loss of 1.1072, cover loss of 0.0042, secret loss of 1.1030
Training: Batch 853/870. Loss of 0.6779, cover loss of 0.0040, secret loss of 0.6739
Training: Batch 854/870. Loss of 2.2714, cover loss of 0.0042, secret loss of 2.2671
Training: Batch 855/870. Loss of 1.2985, cover loss of 0.0084, secret loss of 1.2901
Training: Batch 856/870. Loss of 1.1800, cover loss of 0.0127, secret loss of 1.1672
Training: Batch 857/870. Loss of 1.3025, cover loss of 0.0039, secret loss of 1.2986
Training: Batch 858/870. Loss of 1.6549, cover loss of 0.0039, secret loss of 1.6510
Training: Batch 859/870. Loss of 1.0353, cover loss of 0.0041, secret loss of 1.0312
Training: Batch 860/870. Loss of 1.0060, cover loss of 0.0023, secret loss of 1.0037
Training: Batch 861/870. Loss of 1.0824, cover loss of 0.0017, secret loss of 1.0808
Training: Batch 862/870. Loss of 0.6250, cover loss of 0.0026, secret loss of 0.6223
Training: Batch 863/870. Loss of 1.0465, cover loss of 0.0020, secret loss of 1.0445
Training: Batch 864/870. Loss of 1.4767, cover loss of 0.0172, secret loss of 1.4595
Training: Batch 865/870. Loss of 1.2822, cover loss of 0.0045, secret loss of 1.2778
Training: Batch 866/870. Loss of 2.0761, cover loss of 0.0026, secret loss of 2.0734
Training: Batch 867/870. Loss of 1.3348, cover loss of 0.0035, secret loss of 1.3313
Training: Batch 868/870. Loss of 0.7506, cover loss of 0.0079, secret loss of 0.7426
Training: Batch 869/870. Loss of 1.5056, cover loss of 0.0041, secret loss of 1.5015
Training: Batch 870/870. Loss of 0.8331, cover loss of 0.0031, secret loss of 0.8301
Epoch [3/3], Average_loss: 1.3146
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 1.69
Loss on secret: 1.67
Loss on cover: 0.01
Total loss: 1.68
Loss on secret: 1.68
Loss on cover: 0.00
Total loss: 1.28
Loss on secret: 1.26
Loss on cover: 0.02
Total loss: 1.79
Loss on secret: 1.78
Loss on cover: 0.00
Average loss on test set: 1.29
```

RMSPROP:

```
Training: Batch 326/341. Loss of 1.7779, cover loss of 0.0656, secret loss of 1.7123
Training: Batch 327/341. Loss of 0.9205, cover loss of 0.1915, secret loss of 0.7290
Training: Batch 328/341. Loss of 1.6046, cover loss of 0.0729, secret loss of 1.5317
Training: Batch 329/341. Loss of 0.9872, cover loss of 0.0265, secret loss of 0.9608
Training: Batch 330/341. Loss of 0.9590, cover loss of 0.0178, secret loss of 0.9412
Training: Batch 331/341. Loss of 1.3388, cover loss of 0.0966, secret loss of 1.2422
Training: Batch 332/341. Loss of 1.3994, cover loss of 0.0362, secret loss of 1.3632
Training: Batch 333/341. Loss of 0.9453, cover loss of 0.0396, secret loss of 0.9058
Training: Batch 334/341. Loss of 1.1558, cover loss of 0.0154, secret loss of 1.1404
Training: Batch 335/341. Loss of 0.9366, cover loss of 0.0147, secret loss of 0.9219
Training: Batch 336/341. Loss of 2.3314, cover loss of 0.0325, secret loss of 2.2989
Training: Batch 337/341. Loss of 3.0707, cover loss of 0.0144, secret loss of 3.0563
Training: Batch 338/341. Loss of 1.3052, cover loss of 0.0251, secret loss of 1.2800
Training: Batch 339/341. Loss of 0.8446, cover loss of 0.0587, secret loss of 0.7858
Training: Batch 340/341. Loss of 1.3741, cover loss of 0.0271, secret loss of 1.3470
Training: Batch 341/341. Loss of 1.7827, cover loss of 0.0290, secret loss of 1.7537
Epoch [3/3], Average_loss: 1.3536
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 1.15
Loss on secret: 1.14
Loss on cover: 0.01
Total loss: 0.71
Loss on secret: 0.68
Loss on cover: 0.02
Total loss: 1.28
Loss on secret: 1.27
Loss on cover: 0.02
Total loss: 0.73
Loss on secret: 0.70
Loss on cover: 0.02
Average loss on test set: 1.27
```

5.Increase module and decrease module:

Increasing or decreasing modules does not influence too much. Increasing may influence loss fluctuation larger.

6.Result:

```
Training: Batch 294/300. Loss of 1.0892, cover loss of 0.0882, secret loss of 1.0010
Training: Batch 295/300. Loss of 1.3108, cover loss of 0.0360, secret loss of 1.2748
Training: Batch 296/300. Loss of 1.4874, cover loss of 0.0273, secret loss of 1.4601
Training: Batch 297/300. Loss of 0.6985, cover loss of 0.0218, secret loss of 0.6767
Training: Batch 298/300. Loss of 1.2844, cover loss of 0.0373, secret loss of 1.2472
Training: Batch 299/300. Loss of 1.1444, cover loss of 0.0505, secret loss of 1.0939
Training: Batch 300/300. Loss of 0.7013, cover loss of 0.0463, secret loss of 0.6550
Epoch [3/3], Average_loss: 1.2957
f:\MATH 110A\FinalProject.py:392: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_secret = Variable(test_secret, volatile=True)
f:\MATH 110A\FinalProject.py:394: UserWarning: volatile was removed and now has no effect. Use `with torch.no_grad():` instead.
  test_cover = Variable(test_cover, volatile=True)
Total loss: 1.23
Loss on secret: 1.21
Loss on cover: 0.02
Total loss: 1.20
Loss on secret: 1.15
Loss on cover: 0.05
Total loss: 0.83
Loss on secret: 0.82
Loss on cover: 0.02
Total loss: 1.99
Loss on secret: 1.98
Loss on cover: 0.01
Average loss on test set: 1.28
```

This is the best attemption we had, both the training and testing loss is reduced by 3 percent than the author's loss.

Reference:
https://ieeexplore.ieee.org/document/7915002