Weekly Work Report

Hongzhi Liu

VISION@OUC

March 31, 2019

1 Research problem

As an important subdiscipline of imformation hiding, steganography has become one of important research fields in recent years with rapid development of deep learning. Image steganography has the advantage of embedding capacity, but the security and stability should be improved in further research.

During image-to-image translation research task, we find that CycleGAN [2] is a successful approach to learn a transformation between two image distributions. In our experiment for transformation between face and flower, it is easy to observe that CycleGAN model learns to hide imformation about a source image into the images it generates in the target domain.

Therefore, we want to learn how CycleGAN can hide the imformation and where the information is placed. Besides, the steganographic capacity of this model need to be qualitative and quantitative analyzed, then we should demonstrate that this method is able to use not only in image-to-image translation but also in steganography task. In a word, steganography based on CycleGAN model will be the study target of our paper for NeurIPS 2019.

2 Research approach

In this research task, reproducing code about CycleGAN is the first thing to do. I pay close attention to network architecture and its code implementation which works by training two transformation in parallel.

Visualization tool is the indispensable part of the research, especially study about convolutional neural networks. During the process of generating fake images, feature maps need to be visualized when we want to know how source images can translated into fake images in the generator which has both encoder and decoder. Moreover, the effect of gaussian white noise on generating and recovering process is observable and measurable with this useful tool.

The basic principle and common methods of digital image processing is widely used in this research. Source images and reference images need to be concated as samples for training. The generated fake images need to be cropped and normalized in order to recover for next step.

Furthermore, I need to read several paper relevant to this research task, such as Augmented CycleGAN [2] and SteganoGAN [1], learing the methods they choose to improve the capacity of model to work in steganographic task.

3 Research progress

During this period time, I have a clearer understanding of the model. Our method has great potential in steganography task. Through qualitative and quantitative analysis, we know about sensitivity of generator. But there is still a crucial component working as as expected. Then I will list details about weekly work in Tab. 1 below.

Table 1: Weekly work progress.

EncripGAN	Finish modifying code for six channels training. Finish evaluating the sensitivity of model by adding noise into the generated images. Finish visualizations of the last layer of resblock.

4 Progress in this week

This week I continue to check and modify the code in the training and testing process. First, I modify code to use six channels of input images which includes source and reference image into the generator. Then through several experiment, the sensitivity of model has been evaluated. And I read paper about cycleGAN and its improved algorithm.

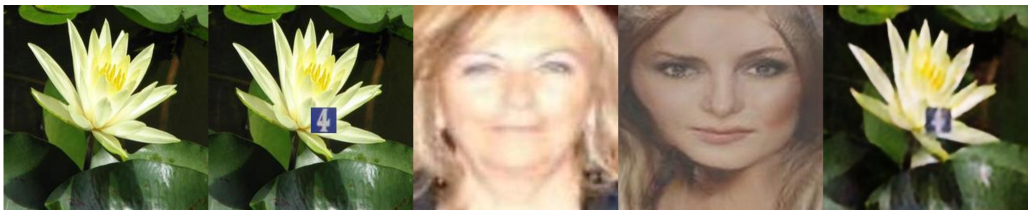


Figure 1: Images of test results generated by the method. The leftmost is base image. The second left is maskimage. The middle one is the reference image. The second one on the right is fake image generated by the B2A generator. The rightmost is the recovered image by the A2B generator.

Step 1 Finish modifying code for six channels training.

Step 2 Finish evaluating the sensitivity of model by adding noise into the generated images.

Step 3 Finish visualizations of the last layer of resblock.

Step 4 Learning about PCA(Principal Component Analysis) to analyze about feature map of generator.

4.1 Hidden Information

Image-to-image translation is the task of taking an image from one class of images and rendering it in the style of another class. CycleGAN is one technique of translation task which requires only unpaired examples from two image domains X and Y. CycleGAN works by training two transformations $F : X \rightarrow Y$and $G : Y \rightarrow X$ in parallel, with the goal of satisfying the following two conditions:

$$Fx \sim p(y) \quad for \quad x \sim p(x), \quad and \quad Gy \sim p(x) \quad for \quad y \sim p(y) \tag{1}$$
$$GFx = x \quad for \quad all \quad x \in X, \quad and \quad FGy = y \quad for \quad all \quad y \in Y. \tag{2}$$

where $p(x)$ and $p(y)$ describe the distributions of two domains of images X and Y. The first condition ensures that the generated images appear to come from the desired domains and is enforced by training two discriminators on X and Y respectively as Eq. 1. The second condition as Eq. 2 ensures that the information about a source image is encoded in the generated image and is enforced by a cyclic consistency loss of the form $\parallel GFx - x \parallel + \parallel FGy - y \parallel$. The hope is that the information about the source image x is encoded semantically into elements of the generated image Fx.

The experiment begins with a curious observation, illustrated in Fig. 1. We first take an flower photograph x with a number picture in the middle that was unseen by the network at training time. Since the network was trained to minimize the cyclic consistency loss, one would expect that $x \approx GFx$, and indeed the two images (the rightmost and the second on the left) turn out to be nearly identical.

It is obvious that CycleGAN is learning an encoding scheme in which it hides number information about the flower photograph x within the generated face image Fx. This strategy is not as surprising as it seems at first glance, since it is impossible for a CycleGAN model to learn a perfect one-to-one correspondence between flower photographs and face images.

In light of the discovery, we focus research on how CycleGAN can hide flower information into a face photograph as shown in Fig. 3. From Fig. 3(a) to Fig. 3(c), the generator encodes the flower photograph with number through the convolution network. In other word, the model learns a steganographic strategy in order to reconstruct the image during this traing process. We can see that CycleGAN may be able to encode this hidden information which incldues both flower and number, using feature map from Generator c1 to c3. The heatmap corresponding to the feature map as shown in Fig. 4. And the code implement can be seen in Algorithm 4.1.

Then it's natural for us to focus on the reconstruction process as shown in Fig. 5. The fake face image becomes as an input image or source image into the generator , which is generated in the process above, and the recover image is almost the same with the reference image.

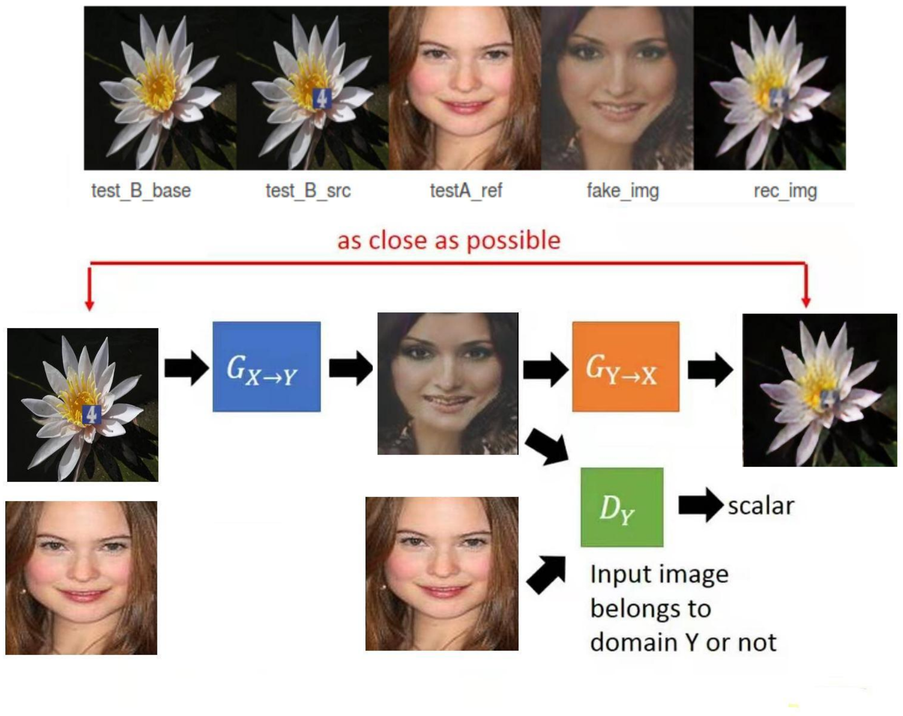
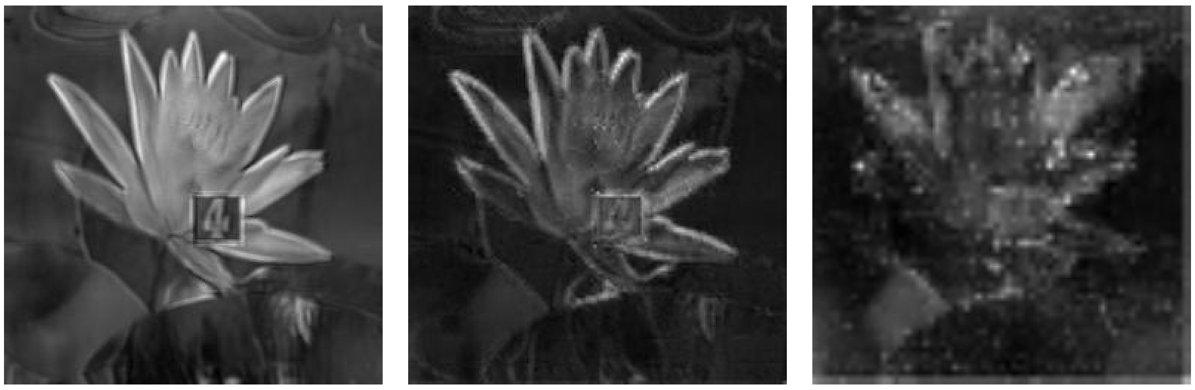
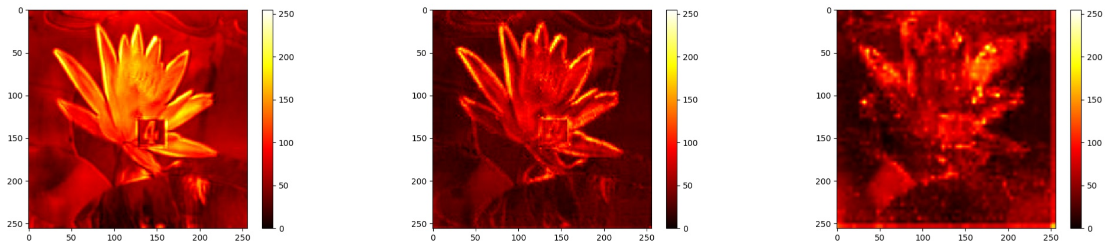


Figure 2: Overall implementation process of method.



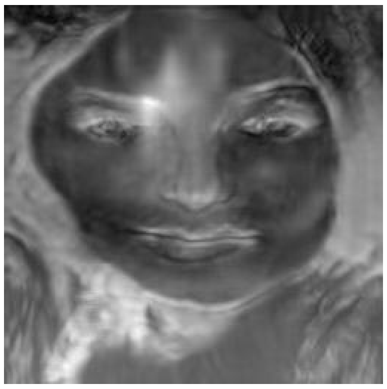
(a) (b) (c)

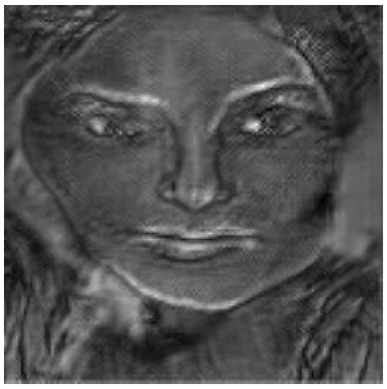
Figure 3: Visual resutls of the former three layers of encoder in process of generating.



(a) (b) (c)

Figure 4: Visual heatmap resutls of the former three layers of encoder in process of generating.


(a)


(b)

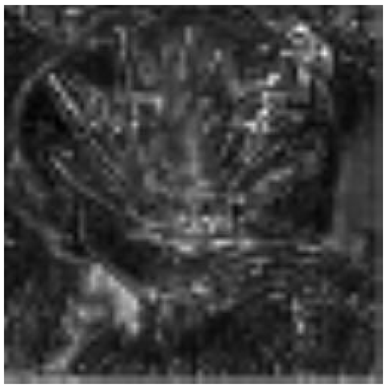
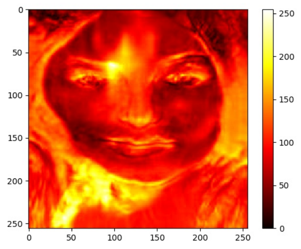
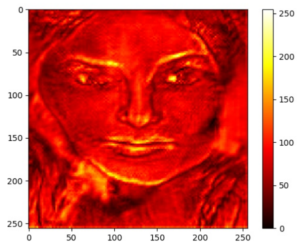

(c)

Figure 5: Visual resutls of the former three layers of encoder in process of recovering.


(a)


(b)

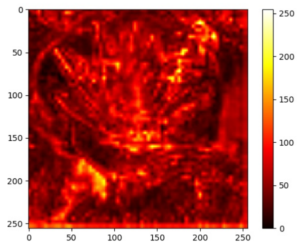

(c)

Figure 6: Visual heatmap resutls of the former three layers of encoder in process of recovering.

From Fig. 5(a) to Fig. 5(c), the generator reconstructs or recovers the flower photograph with fake face image through the convolution network. We can see that CycleGAN may be able to reconstruct information from face, using feature map from Generator c1 to c3. The heatmap corresponding to the feature map as shown in Fig. 6. In a word, CycleGAN hide information which it needs to reconstruct in the generated image, fake face image in our experiment, in order to recover details in the reconstructed images.

```
def save_seg_img_cla(base_src ,vgg ,name, direction ):

data2 = 0
for item in ["e4", "e5", "e6"]:
data1 = vgg[0][item]
data = data2 + data1
data2 = data

m, n = np.shape(data1)[1] ,np.shape(data1)[2]
data_slice = np.reshape(data1 , [m, n])
data_norm = Normalize(data_slice)
mul = np.ones((m,n))
mul*=255.0
data_norm=np.multiply(data_norm ,mul)
save_data = np.asarray(data_norm , np.int8)
processed_r = Image.fromarray(save_data , "L")

save_name_slice = name+"_"+item+"_"+direction + "_classif"
```

```
        + ".jpg"
        processed_r.save(save_name_slice)

        data_slice = mpimg.imread(save_name_slice)
        save_name_slice = name+"_"+item+"_"+direction+"_heatmap_"
        + "classif" + ".jpg"

        filename_txt = name+"_"+item+"_"+direction+"_heatmap_"
        + "classif" + ".txt"
        np.savetxt(filename_txt ,data_slice)

        plt.imshow(data_slice , cmap=plt.cm.hot , vmin=0, vmax=255)
        plt.colorbar()
        #plt.imshow()

        plt.savefig(save_name_slice)
        #plt.show()
        plt.close()


        data_all = data
        m, n = np.shape(data_all)[1], np.shape(data_all)[2]
        data_all = np.reshape(data_all , [m, n])
        data_norm = Normalize(data_all)
        mul = np.ones((m, n))
        mul *= 255.0
        data_norm = np.multiply(data_norm , mul)

        save_data = np.asarray(data_norm , np.int8)
        processed_r = Image.fromarray(save_data , "L")

        save_name = name + "_" + direction + "_classif_all" + ".jpg"
        processed_r.save(save_name)

        data_all = mpimg.imread(save_name)
        save_name_1 = name + "_" + direction + "_heatmap_"
        + "classif_all" + ".jpg"

        filename_txt = name + "_" + direction + "_heatmap_"
        + "classif_all" + ".txt"
        np.savetxt(filename_txt , data_all)

        plt.imshow(data_all , cmap=plt.cm.hot , vmin=0, vmax=255)
        plt.colorbar()
        # plt.imshow()

        plt.savefig(save_name_1)
        # plt.show()
        plt.close()
```

4.2 Analysis of Generator Sensitivity

In a steganographic task, it is important to correctly recover from encrypted image by relying on unique key pairs. Then we design an experiment to eval how sensitive the generator is to noise as shown in Fig. 7.

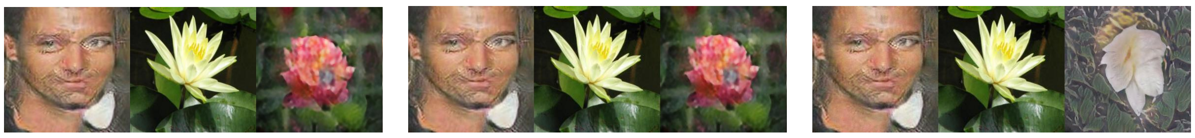

(a) (b) (c)

Figure 7: The results of influences when add noises in the fake images. The leftmost with no noise. The middle image with noise(mean = 0, std= 0.01). And the rightmost image with noise(mean = 0, std= 0.02). And we can see the bad result in the rightmost image which has obscure recovered image.

We add different levels of gaussian white noise into the fake image which means the face image in orger to get a flower photograph with number. The fake face image in Fig. 7(a) is added no noise, in the middle it has noise with std 0.01, and in the rightmost one it has noise with std 0.02. Comparing with these three images, we can realize that the generator is sensitive to noise only three levels when the image is quantized by 8-bit integers.

However, the results wo got from this experiment is far from we want. The most crucial issue is that reference image does not work in the recover process just as Fig. 7(c), which should only recover from a red flower key but the yellow one also work during the recovering. This is the question we need to further study.

```
sample_files_testB = glob('./datasets/{}/*.*'.format(
self.dataset_dir + '/testB'))
for flower_test in sample_files_testB:
count += 1

ref_image_2 = cv2.imread(flower_test)[:, :, ::-1]
ref_image_2 = scipy.misc.imresize(ref_image_2, [256, 256])
ref_image_2 = ref_image_2[np.newaxis, :, :, :]

fake_img_2 = cv2.normalize(fake_img_2, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
fake_img_2 = np.squeeze(fake_img_2)
fake_img_noise = addGaussianNoise(fake_img_2, mean=0,

var=0.0004)
fake_img_noise = fake_img_noise[np.newaxis, :, :, :]

sample_image_an = np.concatenate([fake_img_noise, fake_img_noise,
ref_image_2], axis=2)
sample_image_an = sample_image_an/127.5 -1
sample_image_an = np.array(sample_image_an).astype(np.float32)
image_addnoise_path = args.test_dir+ "/" + name +"/" +
str(count) + "/"
if not os.path.exists(image_addnoise_path):
os.makedirs(image_addnoise_path)
image_addnoise_savepath = os.path.join(image_addnoise_path,
'{0}_{1}_{2}'.format(args.which_direction, 'sample_addnoise',
os.path.basename(sample_file)))
save_images(sample_image_an, [1, 1], image_addnoise_savepath)

fake_img_B, rec_img_A, A2B_visual_A, B2A_visual_recover_B =
self.sess.run(
[out_var_addnoise, rec_var_addnoise, self.A2B_visual,
self.B2A_recover_visual], feed_dict={self.test_A: sample_image_an})
fake_img_addnoise = np.concatenate([sample_image_an, fake_img_B,
rec_img_A], axis=2)
fake_img_addnoise_path = os.path.join(image_addnoise_path,
'{0}_{1}_{2}'.format(args.which_direction, 'addnoise_recover',
os.path.basename(sample_file)))
save_images(fake_img_addnoise, [1, 1], fake_img_addnoise_path)
save_seg_img(A2B_visual_A, image_addnoise_path + name + str(count),
"A2B_addnoise_visual_")
save_seg_img(B2A_visual_recover_B, image_addnoise_path + name +
str(count), "B2A_addniose_visual_recover")
```

5 Plan

Objective:	Finish thesis for NeurIPS 2019.
Deadline:	2019.5.23

2019.04.01—2019.04.07	Trying to analyze the influence of reference image and enhance its effects.
2018.04.08—2018.04.14	Finish quantitative analysis of encryption effect compared with NISP2017 paper.
2018.04.15—2018.04.21	Finish introduction and experimental part in paper.
2018.04.22—2018.04.28	Finish draft.
2018.04.29—2018.05.05	Finish adding quantitative index.
2018.05.06—2018.05.12	Finish model refinement and detailed figures and tables.
2018.05.13—2018.05.19	Finish checking references and grammar.
2018.05.20—2018.05.23	Finish final draft for NeurIPS 2019 paper.

References

[1] K. A. Zhang, A. Cuesta-Infante, and K. Veeramachaneni. SteganoGAN: Pushing the limits of image steganography. *arXiv preprint arXiv:1901.03892*, 2019. 1

[2] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017. 1