PA2 Report

Topic : Context Encoder

Name : Minseok Kim

Student ID : 20215058

1. Test class

1.1 Please attach plots for 0.png through 5.png located in the test folder in the report file. [10 points]

1.2 Please describe the testing/training methods of the context encoder in the report file. [10 points]

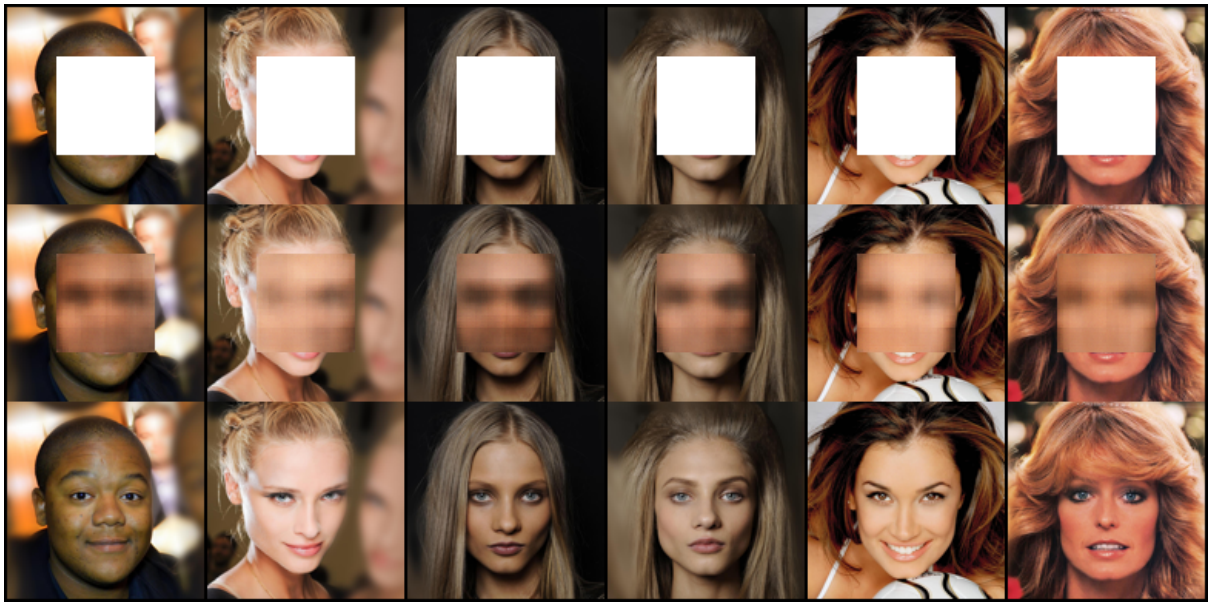
- Our goal is to inpain the mask of the image using the difference from ground truth.
- During the test time, we apply the center mask to the test image and inpaint the empty part.
- During the training time, we apply a random mask to the training image and inpaint the empty part.
- Network consist the generator and discriminator.

- Loss consist L1 loss and adversarial loss, that's why we can expect the realistic image.

2. Train class

2.1 Please execute the initial training code and attach plots for 0.png through 5.png located in the `test' folder in the report file using weight files obtained at 3 epoch and at 50 epoch, respectively. [10 points]

- Results of 3 epoch



- Results of 50 epoch

2.2 The initial code implements the context encoder training only with pixel-wise loss (torch.nn.L1Loss()). Please extend the implementation by adding the adversarial loss in the `Trainer' class to train your network using both pixel-wise and adversarial losses. (Please properly tune your learning rate, batchSize and # of epochs etc. to properly train your network.) [35 points]

```
# Generate a batch of images
gen_parts = self.gnet(masked_imgs)

# (1) Update D network : maximize log(D(x)) + log(1-D(G(z)))
# train with real
self.optimizer_D.zero_grad()
output = self.dnet(masked_parts))
errD_real = self.binary_cross_entropy(output, valid)
errD_real.backward()

# train with fake
gen_parts = self.gnet(masked_imgs)
output = self.dnet(gen_parts.detach())
errD_fake = self.binary_cross_entropy(output, fake)
errD_fake.backward()

self.optimizer_D.step()

# (2) Update G network : maximize log(D(G(z)))
self.optimizer_G.zero_grad()
gen_parts = self.gnet(masked_imgs)
output = self.dnet(gen_parts)
errG = self.binary_cross_entropy(output, valid)
errG.backward()

self.optimizer_G.step()
```

2.3 Please visualize the generated images with and without the adversarial loss and discuss the quality of the results. [10 points]

- with adversarial loss

- without adversarial loss

- discussion
  - As a result of learning without adversarial loss, the color was similar, but the image became blurred.
  - As a result of learning with adversarial loss, the image became realistic and not much blurred.

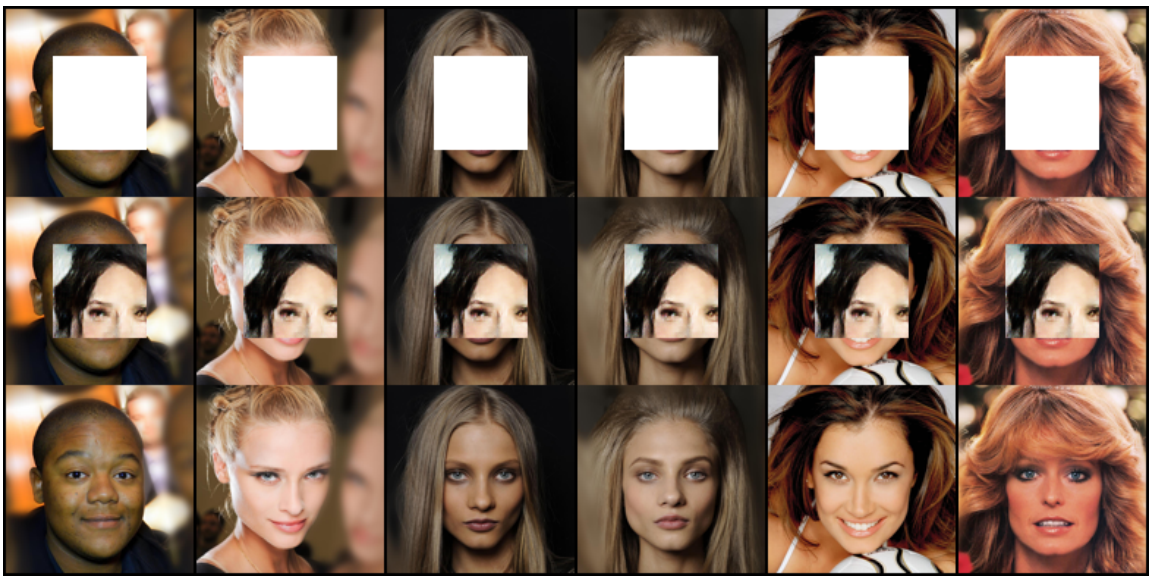
2.4 When implementing additional adversarial loss, one may need to combine two losses with the

constant lambda, as follows:

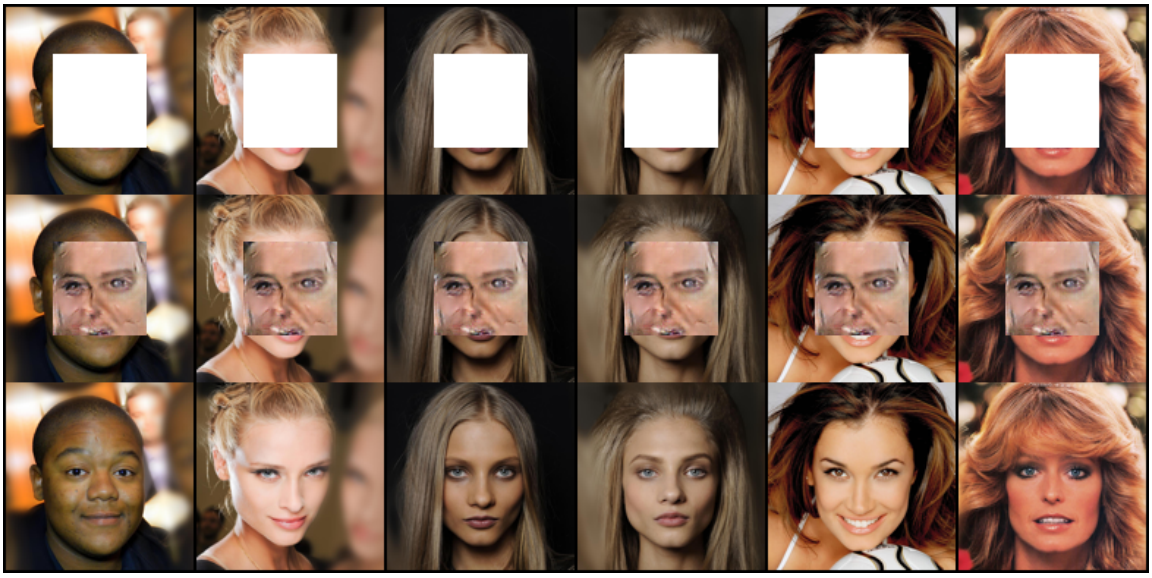
adv_loss * lambda + pixel_loss

Please visualize the generated images at epoch 50 by changing the value of `lambda' and please discuss the possible reasons for that. [20 points]

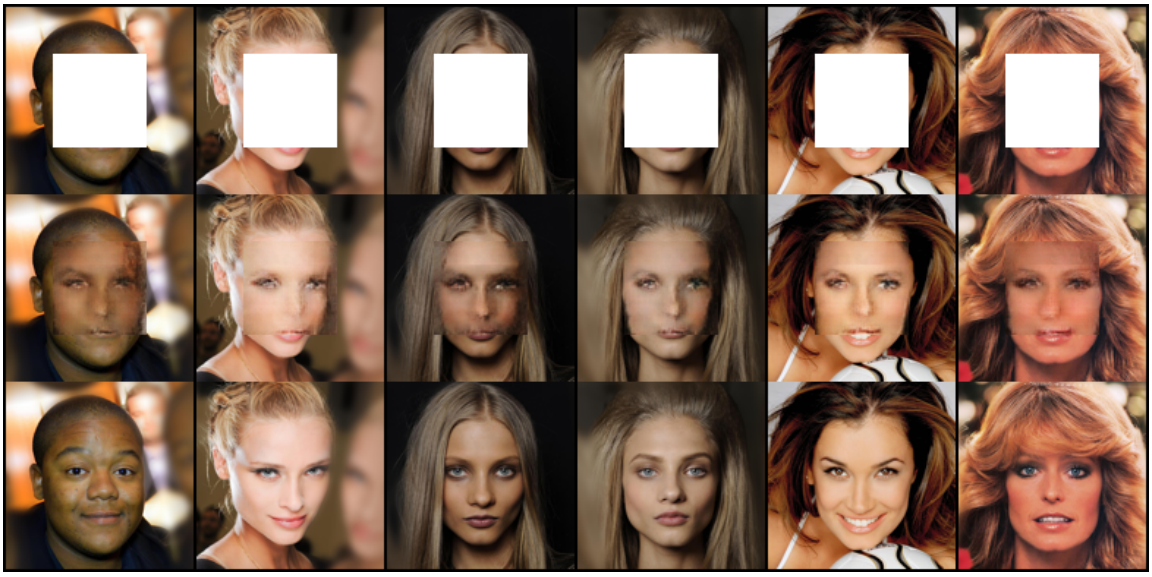
- g_lamda = 1, d_lambda = 1, (it means the lambda's value is 1)



- g_lamda = 0.001, d_lamda = 0.001, (it means the lambda's value is 0.001)



- g_lambda = 0.001, d_lambda = 1



- discussion
  - While tuning this hyperparameter, I felt that the performance was greatly changed by lambda in training the generation model.
  - When lambda is set to 1 or 0.001, bad results were obtained.
  - However, when I decomposed and tuned lambda into g_lambda and d_lambda, very good results were obtained when g_lambda was set to 0.001 and d_lambda was set to 1.

- When g_lambda was set to be smaller than d_lambda, the discriminator was well learned.

Option. How to execute my code?

1. Copy the main.py to google colab(ipynb).
2. Upload datasets in google drive.
3. Upload .pth files in google drive.
4. Upload .py files in google colab environment.
5. Check the folder path, it is conformed by original source code's location!
6. When you execute train mode, free the comment of 197,198 lines of *main.py*