PA 5: Generative Adversarial Networks

20222421 AIGS SimJaeYoon

0. Overview

This assignment is about Generative Adversarial Networks(GAN). The goal of this assignment is generating images with concepts of GAN. First problem is generating images from latent vector, the second problem is image-to-image translation, and the third problem is generating images from the caption.

1. [Problem #1] Training and Testing DC-GAN and Vector Arithmetic

Problem #1 is about training DC-GAN model. We can read and understand the DC-GAN using generator and discriminator network. And we can implement the DC-GAN model from github repository.

https://github.com/pytorch/examples/tree/main/dcgan

GitHub - Annusha/dcgan: train/test dcgan + arithmetic

man-woman-detection | Kaggle

Face Mask Detection ~12K Images Dataset | Kaggle

1.1. Try the efforts to improve the performance on your network. For example, your hyper-parameter setting or collecting dataset or your network improvements that are not provided by the basic codes.

First, I implemented main.py based on the reference code in github. This main.py file will be used for training. Subsequently, code was additionally implemented to perform arithmetic. I implemented generator.py, arg_parse.py, and arthmetic.py based on another github reference code.

In the case of dataset, I downloaded it from Kaggle and used it. I downloaded the face mask dataset using the reference site. After that, the hyper-parameter was set and used for learning. It was set to batchSize=64, beta1=0.5, imageSize=64, lr=0.0002, ndf=64, and ngf=64. And in the case of the most important number of iteration, the test was conducted by adjusting it from 200 to 1000.

The entire training process started with main.py and used generator.py and arg_parse.py. From this, new data was created and noise value was stored along with it. And I tried to choose a clean image for arithmetic. It seemed good to choose a clean image and noise. So, I did arithmetic on the selected image and noise value using arithmetic.py.

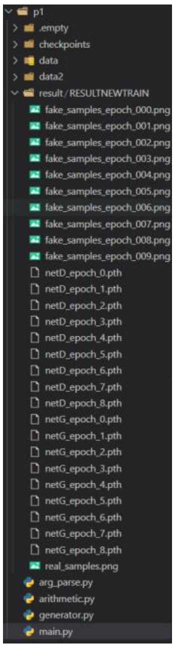
To proceed with the training, I used main.py as follows. Before that, the necessary package written in the requirement.txt file was installed (ex.lmdb). After that, learning was conducted after entering the parsing variable as follows.

```
(hhh) user@7ffe62bf4ffe:/home/DL/assn5/p1$ python main.py --dataset folder --dataroot '/home/DL/assn5/p1/data' --outf '/home/DL/assn5/p1/result/RESULTNEWTRAIN/' --niter 1000 --cuda &
```

Here, the “dataset” may specify the format of the dataset I used the data in the designated path using folder. “dataroot” is the path of the learning image, and “outf” is the path of output. "niter" determines the number of learning epochs, and "cuda" determines the GPU used for learning. Then, it can be seen that learning proceeds as follows.

```
[2/1000][65/185] Loss_D: 0.8497 Loss_G: 5.0124 D(x): 0.7856 D(G(z)): 0.3928 / 0.0087
[2/1000][66/185] Loss_D: 1.0320 Loss_G: 1.6276 D(x): 0.5100 D(G(z)): 0.0745 / 0.2393
[2/1000][67/185] Loss_D: 1.3882 Loss_G: 6.8734 D(x): 0.9058 D(G(z)): 0.6706 / 0.0015
[2/1000][68/185] Loss_D: 1.4597 Loss_G: 2.8750 D(x): 0.3440 D(G(z)): 0.0140 / 0.0912
[2/1000][69/185] Loss_D: 0.7883 Loss_G: 3.4513 D(x): 0.8019 D(G(z)): 0.3575 / 0.0497
[2/1000][70/185] Loss_D: 0.8254 Loss_G: 3.6001 D(x): 0.7197 D(G(z)): 0.3042 / 0.0386
[2/1000][71/185] Loss_D: 1.0279 Loss_G: 3.5542 D(x): 0.6596 D(G(z)): 0.3196 / 0.0414
[2/1000][72/185] Loss_D: 0.9845 Loss_G: 3.4177 D(x): 0.6694 D(G(z)): 0.3308 / 0.0408
[2/1000][73/185] Loss_D: 0.6320 Loss_G: 4.7668 D(x): 0.8077 D(G(z)): 0.2969 / 0.0110
[2/1000][74/185] Loss_D: 0.9262 Loss_G: 1.7259 D(x): 0.5581 D(G(z)): 0.1425 / 0.2064
[2/1000][75/185] Loss_D: 1.5435 Loss_G: 7.0029 D(x): 0.7984 D(G(z)): 0.6882 / 0.0020
[2/1000][76/185] Loss_D: 2.5382 Loss_G: 1.8423 D(x): 0.1665 D(G(z)): 0.0189 / 0.2051
[2/1000][77/185] Loss_D: 1.4200 Loss_G: 4.5892 D(x): 0.8491 D(G(z)): 0.6490 / 0.0264
[2/1000][78/185] Loss_D: 1.0054 Loss_G: 2.6445 D(x): 0.5122 D(G(z)): 0.1211 / 0.1125
[2/1000][79/185] Loss_D: 1.4497 Loss_G: 5.3118 D(x): 0.8313 D(G(z)): 0.6631 / 0.0098
[2/1000][80/185] Loss_D: 1.2405 Loss_G: 3.0656 D(x): 0.5014 D(G(z)): 0.1463 / 0.0617
[2/1000][81/185] Loss_D: 1.0945 Loss_G: 3.5263 D(x): 0.6938 D(G(z)): 0.3937 / 0.0501
[2/1000][82/185] Loss_D: 1.1838 Loss_G: 3.1157 D(x): 0.6168 D(G(z)): 0.3144 / 0.0556
```

Then, it can be seen that the weight and image according to epoch are stored in the result folder made in advance as follows.



When learning is completed with main.py, new data is created based on the learned model. For this purpose, generator.py and arg_parse.py codes were implemented. The following is the arg_parse.py code, which sets variables that are used jointly by the generator and subsequent arithmetic.

```python
parser = argparse.ArgumentParser()
# parser.add_argument('--dataset', required=True, help='cifar10 | lsun | imagenet | folder | lfw | fake')
parser.add_argument('--dataset', default='lsun', help='cifar10 | lsun | imagenet | folder | lfw | fake')
parser.add_argument('--dataroot', default='./data/', help='path to dataset')
parser.add_argument('--workers', type=int, help='number of data loading workers', default=4)
parser.add_argument('--batchSize', type=int, default=128, help='input batch size')
parser.add_argument('--imageSize', type=int, default=64, help='the height / width of the input image to network')
parser.add_argument('--nz', type=int, default=100, help='size of the latent z vector')
parser.add_argument('--ngf', type=int, default=64)
parser.add_argument('--ndf', type=int, default=64)
parser.add_argument('--niter', type=int, default=1000, help='number of epochs to train for')
parser.add_argument('--lr', type=float, default=0.0002, help='learning rate, default=0.0002')
parser.add_argument('--beta1', type=float, default=0.5, help='beta1 for adam. default=0.5')
parser.add_argument('--netG', default='/home/DL/assn5/p1/result/RESULTNEWTRAIN/netG_epoch_999.pth', help="path to netG (to continue training)")
parser.add_argument('--netD', default='/home/DL/assn5/p1/result/RESULTNEWTRAIN/netD_epoch_999.pth', help="path to netD (to continue training)")
parser.add_argument('--outf', default='/home/DL/assn5/p1/result/gimages/', help='folder to output images and model checkpoints')
parser.add_argument('--manualSeed', type=int, help='manual seed')
parser.add_argument('--train_svm', action='store_true', help='enable train svm using saved features')

opt = parser.parse_args()
print(opt)

try:
    os.makedirs(opt.outf)
except OSError:
    pass

if opt.manualSeed is None:
    opt.manualSeed = random.randint(1, 10000)
print("Random Seed: ", opt.manualSeed)
random.seed(opt.manualSeed)
torch.manual_seed(opt.manualSeed)
# if opt.cuda:
torch.cuda.manual_seed_all(opt.manualSeed)

cudnn.benchmark = True

# define inner parameters of network which depend on imagesize
kernels = []
strides = []
pads = []
if opt.imageSize == 64:
    kernels = [4, 4, 4, 4, 4]
    strides = [1, 2, 2, 2, 2]
    pads =    [0, 1, 1, 1, 1]

if opt.imageSize == 32:
    # first structure
    # kernels = [4, 4, 2, 4, 4]
    # strides = [1, 2, 2, 1, 2]
    # pads =    [0, 1, 2, 0, 1]

    # second structure
    kernels = [2, 4, 4, 4, 4]
    strides = [1, 2, 2, 2, 2]
    pads =    [0, 1, 1, 1, 1]
```

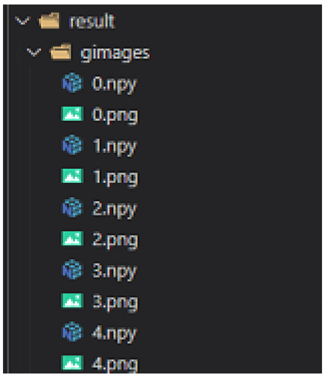
generator.py will randomly generate noise. Thus, an image may be generated through a generator. The following is code for generator network and noise generation and image generation and storage.

```python
nz = int(arg_parse.opt.nz)
ngf = int(arg_parse.opt.ngf)
ndf = int(arg_parse.opt.ndf)
nc = 3

class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.main = nn.Sequential(
            # input is Z, going into a convolution
            nn.ConvTranspose2d(     nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(True),
            # state size. (ngf*8) x 4 x 4
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),
            # state size. (ngf*4) x 8 x 8
            nn.ConvTranspose2d(ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),
            # state size. (ngf*2) x 16 x 16
            nn.ConvTranspose2d(ngf * 2,     ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            # state size. (ngf) x 32 x 32
            nn.ConvTranspose2d(    ngf,      nc, 4, 2, 1, bias=False),
            nn.Tanh()
            # state size. (nc) x 64 x 64
        )

    def forward(self, input):

        output = self.main(input)

        return output
```

```python
def _create_and_save(netG):
    number = len(os.listdir(opt.outf))

    for i in range(number, number + opt.niter):

        noise = torch.FloatTensor(1, opt.nz, 1, 1).normal_(0, 1)
        #noise = torch.FloatTensor(1, opt.nz, 1, 1).uniform_(0, 1)
        noise = Variable(noise)
        noise = noise.cuda()

        noise_np = noise.cpu().numpy()
        np.save(opt.outf + '%d' % i, noise_np)

        fake = netG(noise)
        vutils.save_image(fake.data, opt.outf + '%d.png' % i, normalize=True, nrow=4)


if __name__ == '__main__':
    netG = Generator()


    if opt.dataset == 'imagenet' :
        path_root = opt.dataroot
        path_Gs = [os.path.join(path_root, i) for i in os.listdir(path_root) if 'netG' in i]

        for path_G in path_Gs:
            digit = int(re.findall(r'\d+', path_G)[0])
            if digit < 30:
                print('save from epoch %d'%digit)
                netG.load_state_dict(torch.load(path_G))
                netG.cuda()
                netG.eval()

                _create_and_save(netG)

    else:
        if opt.netG == '':
            print('load weights for generator')
            exit(-1)
        netG.load_state_dict(torch.load(opt.netG))
        print(netG)
        netG.cuda()
        netG.eval()

        _create_and_save(netG)
```

The generator can be executed using the stored weight. The normal function, which produced the best image by testing by changing the noise generation method, was used. The noise value is also stored along with the generated image, and only one image is set to be generated at a time.

```
(hhh) user@7ffe62bf4ffe:/home/DL/assn5/p1$ python generator.py
Namespace(batchSize=128, beta1=0.5, dataroot='./data/', dataset='lsun', imageSize=64, lr=0.0002, manua
es/', train_svm=False, workers=4)
Random Seed:  1485
Generator(
  (main): Sequential(
    (0): ConvTranspose2d(3, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
)
```



I would like to do arithmetical using the noise generated in this way. It is intended to create an image in a desired direction using the properties of Latent z.

```python
def preprocess_img(img_v):
    img = img_v.data.cpu().numpy()
    img = img.transpose(0, 2, 3, 1).squeeze()
    img += 1
    img /= 2
    return img

if __name__ == '__main__' :
    noise_A_np = np.load('/home/DL/assn5/p1/result/gimages/26.npy')
    noise_A = torch.from_numpy(noise_A_np).to('cuda')
    image_A = netG(noise_A)
    A_image = preprocess_img(image_A)
    plt.imshow(A_image)
    plt.savefig('A_image.png')
    plt.cla()

    noise_B_np = np.load('/home/DL/assn5/p1/result/gimages/464.npy')
    noise_B = torch.from_numpy(noise_B_np).to('cuda')
    image_B = netG(noise_B)
    B_image = preprocess_img(image_B)
    plt.imshow(B_image)
    plt.savefig('B_image.png')
    plt.cla()

    noise_C_np = np.load('/home/DL/assn5/p1/result/gimages/14.npy')
    noise_C = torch.from_numpy(noise_C_np).to('cuda')
    image_C = netG(noise_C)
    C_image = preprocess_img(image_C)
    plt.imshow(C_image)
    plt.savefig('C_image.png')
    plt.cla()

    final_noise = noise_A - noise_B + noise_C
    final_image = netG(final_noise)
    final_im = preprocess_img(final_image)

    plt.axis('off')
    plt.title('withMaskman - withoutMaskMan + Woman')
    plt.imshow(final_im)
    plt.savefig('A_B_C.png')
    plt.show()
```

1.2. Some result images including generated images using DC-GAN.

Generated Images Using DC-GAN

Example Image 1 : With Mask Man

Example Image 2 : Without Mask Man

Example Image 3 : Woman

With Mask Man – Without Mask Man + Woman = Arithmetic Result Image

1.3. What did you learn through this problem #1.

While doing Problem #1, I was able to learn about the model structure and learning method of DC-GAN. And through arithmetic, I learned how to use DC-GAN's latent z. And I learned that the role of noise is important when generating images using DC-GAN. Also, I learned that arithmetic doesn’t work well as I expected and how to make a code using source code.

1.4. Discuss about the experimental results, network architecture, and training techniques.

From the results, it was found that the arithmetical process was not easy. And it was not easy to create an image from the noise vector. Perhaps the dataset is small in size, so the result is not perfect. And if you use the loud noise value, the result seems to be better. In particular, for hyperparameter nz at first test, we thought that we could create three characteristic features if we set nz to 3, but there was a problem that the image could not be generated well. This is thought to be because DC-GAN itself cannot define the exact condition, so if the information is too condensed during encoding by reducing nz, decoding is not good.

2. [Problem #2] Training and Testing Paired Image-to-Image Translation

Problem #2 is about training Pix2pix model. We can read and understand the pix2pix using generator and discriminator network. And we can implement the pix2pix model from github repository. We should train and test the pix2pix network with “Façade Dataset”. We can download the data set from github repository, too. The following figure shows an example of Facades Dataset.

- GitHub - junyanz/pytorch-CycleGAN-and-pix2pix: Image-to-Image Translation in PyTorch

2.1. Try the efforts to improve the performance on your network. For example, your hyper-parameter setting or collecting dataset or your network improvements that are not provided by the basic codes.

To use the source code downloaded from github, we first built an environment.

```
(p2) user@7ffe62bf4ffe:/home/DL/assn5/p2$ conda env create -f environment.yml
Collecting package metadata (repodata.json): done
Solving environment: done
```

The conda environment was constructed as follows using the “environment.yml” file.

```
(p2) user@7ffe62bf4ffe:/home/DL/assn5/p2$ conda info --envs
# conda environments:
#
base                     /home/user/miniconda
assn                     /home/user/miniconda/envs/assn
hhh                      /home/user/miniconda/envs/hhh
p2                    *  /home/user/miniconda/envs/p2
pytorch-CycleGAN-and-pix2pix     /home/user/miniconda/envs/pytorch-CycleGAN-and-pix2pix

(p2) user@7ffe62bf4ffe:/home/DL/assn5/p2$ source activate pytorch-CycleGAN-and-pix2pix
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$
```

Dataset download to be used for pix2pix model was carried out using shell script as follows.

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$ ./datasets/download_pix2pix_dataset.sh facades
Specified [facades]
WARNING: timestamping does nothing in combination with -O. See the manual
for details.

--2022-05-25 04:24:34--  http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/facades.tar.gz
Resolving efrosgans.eecs.berkeley.edu (efrosgans.eecs.berkeley.edu)... 128.32.244.190
Connecting to efrosgans.eecs.berkeley.edu (efrosgans.eecs.berkeley.edu)|128.32.244.190|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30168306 (29M) [application/x-gzip]
Saving to: './datasets/facades.tar.gz'

./datasets/facades.tar.gz                                   100%[=======================================

2022-05-25 04:25:19 (649 KB/s) - './datasets/facades.tar.gz' saved [30168306/30168306]

facades/
facades/test/
facades/test/27.jpg
facades/test/5.jpg
facades/test/72.jpg
facades/test/1.jpg
facades/test/10.jpg
facades/test/100.jpg
facades/test/101.jpg
facades/test/102.jpg
facades/test/103.jpg
```

For training, the hyperparameter was set using "train_options.py" and "base_options.py" as follows.

```python
def initialize(self, parser):
    """Define the common options that are used in both training and test."""
    # basic parameters
    parser.add_argument('--dataroot', required=True, help='path to images (should have subfolders trainA, trainB, valA, valB, etc)')
    parser.add_argument('--name', type=str, default='facades_pix2pix', help='name of the experiment. It decides where to store samples and models')
    parser.add_argument('--use_wandb', action='store_true', help='use wandb')
    parser.add_argument('--gpu_ids', type=str, default='0', help='gpu ids: e.g. 0  0,1,2, 0,2. use -1 for CPU')
    parser.add_argument('--checkpoints_dir', type=str, default='./checkpoints', help='models are saved here')
    # model parameters
    parser.add_argument('--model', type=str, default='pix2pix', help='chooses which model to use. [cycle_gan | pix2pix | test | colorization]')
    parser.add_argument('--input_nc', type=int, default=3, help='# of input image channels: 3 for RGB and 1 for grayscale')
    parser.add_argument('--output_nc', type=int, default=3, help='# of output image channels: 3 for RGB and 1 for grayscale')
    parser.add_argument('--ngf', type=int, default=64, help='# of gen filters in the last conv layer')
    parser.add_argument('--ndf', type=int, default=64, help='# of discrim filters in the first conv layer')
    parser.add_argument('--netD', type=str, default='basic', help='specify discriminator architecture [basic | n_layers | pixel]. The basic model is a 70x70 PatchGAN. n_layers allows you to specify the layers in the discriminator')
    parser.add_argument('--netG', type=str, default='resnet_9blocks', help='specify generator architecture [resnet_9blocks | resnet_6blocks | unet_256 | unet_128]')
    parser.add_argument('--n_layers_D', type=int, default=3, help='only used if netD==n_layers')
    parser.add_argument('--norm', type=str, default='batch', help='instance normalization or batch normalization [instance | batch | none]')
    parser.add_argument('--init_type', type=str, default='normal', help='network initialization [normal | xavier | kaiming | orthogonal]')
    parser.add_argument('--init_gain', type=float, default=0.02, help='scaling factor for normal, xavier and orthogonal.')
    parser.add_argument('--no_dropout', action='store_true', help='no dropout for the generator')
    # dataset parameters
    parser.add_argument('--dataset_mode', type=str, default='aligned', help='chooses how datasets are loaded. [unaligned | aligned | single | colorization]')
    parser.add_argument('--direction', type=str, default='BtoA', help='AtoB or BtoA')
    parser.add_argument('--serial_batches', action='store_true', help='if true, takes images in order to make batches, otherwise takes them randomly')
    parser.add_argument('--num_threads', default=4, type=int, help='# threads for loading data')
    parser.add_argument('--batch_size', type=int, default=1, help='input batch size')
    parser.add_argument('--load_size', type=int, default=286, help='scale images to this size')
    parser.add_argument('--crop_size', type=int, default=256, help='then crop to this size')
    parser.add_argument('--max_dataset_size', type=int, default=float("inf"), help='Maximum number of samples allowed per dataset. If the dataset directory contains more than max_dataset_size, only a subset is loaded.')
    parser.add_argument('--preprocess', type=str, default='resize_and_crop', help='scaling and cropping of images at load time [resize_and_crop | crop | scale_width | scale_width_and_crop | none]')
    parser.add_argument('--no_flip', action='store_true', help='if specified, do not flip the images for data augmentation')
    parser.add_argument('--display_winsize', type=int, default=256, help='display window size for both visdom and HTML')
    # additional parameters
    parser.add_argument('--epoch', type=str, default='latest', help='which epoch to load? set to latest to use latest cached model')
    parser.add_argument('--load_iter', type=int, default='0', help='which iteration to load? if load_iter > 0, the code will load models by iter_[load_iter]; otherwise, the code will load models by [epoch]')
    parser.add_argument('--verbose', action='store_true', help='if specified, print more debugging information')
    parser.add_argument('--suffix', default='', type=str, help='customized suffix: opt.name = opt.name + suffix: e.g., {model}_{netG}_size{load_size}')
    self.initialized = True
    return parser
```

```python
def initialize(self, parser):
    parser = BaseOptions.initialize(self, parser)
    # visdom and HTML visualization parameters
    parser.add_argument('--display_freq', type=int, default=400, help='frequency of showing training results on screen')
    parser.add_argument('--display_ncols', type=int, default=4, help='if positive, display all images in a single visdom web panel with certain number of images per row.')
    parser.add_argument('--display_id', type=int, default=1, help='window id of the web display')
    parser.add_argument('--display_server', type=str, default="http://localhost", help='visdom server of the web display')
    parser.add_argument('--display_env', type=str, default='main', help='visdom display environment name (default is "main")')
    parser.add_argument('--display_port', type=int, default=8097, help='visdom port of the web display')
    parser.add_argument('--update_html_freq', type=int, default=1000, help='frequency of saving training results to html')
    parser.add_argument('--print_freq', type=int, default=100, help='frequency of showing training results on console')
    parser.add_argument('--no_html', action='store_true', help='do not save intermediate training results to [opt.checkpoints_dir]/[opt.name]/web/')
    # network saving and loading parameters
    parser.add_argument('--save_latest_freq', type=int, default=5000, help='frequency of saving the latest results')
    parser.add_argument('--save_epoch_freq', type=int, default=5, help='frequency of saving checkpoints at the end of epochs')
    parser.add_argument('--save_by_iter', action='store_true', help='whether saves model by iteration')
    parser.add_argument('--continue_train', action='store_true', help='continue training: load the latest model')
    parser.add_argument('--epoch_count', type=int, default=1, help='the starting epoch count, we save the model by <epoch_count>, <epoch_count>+<save_latest_freq>, ...')
    parser.add_argument('--phase', type=str, default='train', help='train, val, test, etc')
    # training parameters
    parser.add_argument('--n_epochs', type=int, default=100, help='number of epochs with the initial learning rate')
    parser.add_argument('--n_epochs_decay', type=int, default=100, help='number of epochs to linearly decay learning rate to zero')
    parser.add_argument('--beta1', type=float, default=0.5, help='momentum term of adam')
    parser.add_argument('--lr', type=float, default=0.0002, help='initial learning rate for adam')
    parser.add_argument('--gan_mode', type=str, default='vanilla', help='the type of GAN objective. [vanilla| lsgan | wgangp]. vanilla GAN loss is the cross-entropy objective used in the original GAN paper.')
    parser.add_argument('--pool_size', type=int, default=50, help='the size of image buffer that stores previously generated images')
    parser.add_argument('--lr_policy', type=str, default='linear', help='learning rate policy. [linear | step | plateau | cosine]')
    parser.add_argument('--lr_decay_iters', type=int, default=50, help='multiply by a gamma every lr_decay_iters iterations')

    self.isTrain = True
    return parser
```

To train, I ran the train.py file as follows.

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$ python train.py --dataroot ./datasets/facades --name facades_pix2pix --model pix2pix --direction BtoA
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: False
                crop_size: 256
                 dataroot: ./datasets/facades                  [default: None]
             dataset_mode: aligned
                direction: BtoA
              display_env: main
             display_freq: 400
               display_id: 1
            display_ncols: 4
             display_port: 8097
           display_server: http://localhost
          display_winsize: 256
                    epoch: latest
              epoch_count: 1
                 gan_mode: vanilla
                  gpu_ids: 0
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: True                                [default: None]
                lambda_L1: 100.0
                load_iter: 0                                   [default: 0]
                load_size: 286
                       lr: 0.0002
           lr_decay_iters: 50
                lr_policy: linear
         max_dataset_size: inf
                    model: pix2pix
                 n_epochs: 100
           n_epochs_decay: 100
               n_layers_D: 3
                     name: facades_pix2pix
                      ndf: 64
                     netD: basic
                     netG: unet_256
                      ngf: 64
               no_dropout: False
                  no_flip: False
                  no_html: False
                     norm: batch
              num_threads: 4
                output_nc: 3
                    phase: train
                pool_size: 0
               preprocess: resize_and_crop
               print_freq: 100
             save_by_iter: False
          save_epoch_freq: 5
         save_latest_freq: 5000
           serial_batches: False
                   suffix: 
         update_html_freq: 1000
                use_wandb: False
                  verbose: False
----------------- End -------------------
dataset [AlignedDataset] was created
The number of training images = 400
```

As learning progresses, the results are shown for each epoch as follows.

```
(epoch: 200, iters: 100, time: 0.036, data: 0.422) G_GAN: 3.706 G_L1: 17.589 D_real: 0.069 D_fake: 0.051
(epoch: 200, iters: 200, time: 0.037, data: 0.004) G_GAN: 1.446 G_L1: 20.361 D_real: 0.071 D_fake: 0.455
(epoch: 200, iters: 300, time: 0.036, data: 0.003) G_GAN: 2.974 G_L1: 14.946 D_real: 0.034 D_fake: 0.097
(epoch: 200, iters: 400, time: 0.774, data: 0.002) G_GAN: 4.774 G_L1: 19.740 D_real: 0.008 D_fake: 0.018
saving the latest model (epoch 200, total_iters 80000)
saving the model at the end of epoch 200, iters 80000
End of epoch 200 / 200   Time Taken: 17 sec
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$
```

In the meantime, weight is saved every five turns of the epoch.

```
135_net_G.pth
140_net_D.pth
140_net_G.pth
145_net_D.pth
145_net_G.pth
150_net_D.pth
150_net_G.pth
latest_net_D.pth
latest_net_G.pth
loss_log.txt
train_opt.txt
```

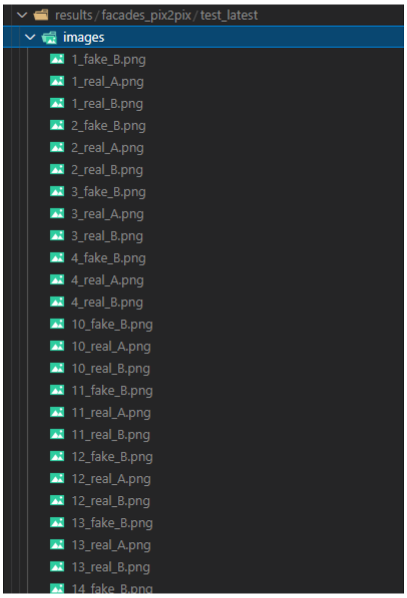
Training was also conducted in the opposite direction.

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$ python train.py --dataroo
t ./datasets/facades --name facades_pix2pix_AtoB --model pix2pix --direction AtoB --gpu_ids 1
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: False
                crop_size: 256
                 dataroot: ./datasets/facades                   [default: None]
             dataset_mode: aligned
                direction: AtoB                                 [default: BtoA]
              display_env: main
             display_freq: 400
               display_id: 1
            display_ncols: 4
             display_port: 8097
           display_server: http://localhost
          display_winsize: 256
                    epoch: latest
              epoch_count: 1
                 gan_mode: vanilla
                  gpu_ids: 1                                    [default: 0]
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: True                                 [default: None]
                lambda_L1: 100.0
                load_iter: 0                                    [default: 0]
                load_size: 286
                       lr: 0.0002
           lr_decay_iters: 50
                lr_policy: linear
         max_dataset_size: inf
                    model: pix2pix
                 n_epochs: 100
           n_epochs_decay: 100
               n_layers_D: 3
                     name: facades_pix2pix_AtoB                 [default: facades_pix2pix]
                      ndf: 64
                     netD: basic
                     netG: unet_256
                      ngf: 64
               no_dropout: False
                  no_flip: False
                  no_html: False
                     norm: batch
              num_threads: 4
                output_nc: 3
                    phase: train
                pool_size: 0
               preprocess: resize_and_crop
               print_freq: 100
             save_by_iter: False
          save_epoch_freq: 5
         save_latest_freq: 5000
           serial_batches: False
                   suffix:
         update_html_freq: 1000
                use_wandb: False
                  verbose: False
----------------- End -------------------
dataset [AlignedDataset] was created
The number of training images = 400
initialize network with normal
initialize network with normal
model [Pix2PixModel] was created
---------- Networks initialized -------------
[Network G] Total number of parameters : 54.414 M
[Network D] Total number of parameters : 2.769 M
-----------------------------------------------
```

To test, I ran the test.py file as follows.

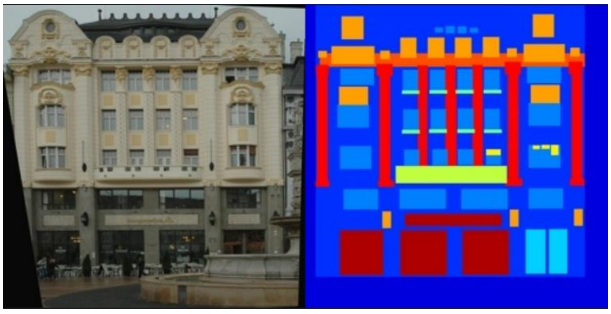
```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$ python test.py --dataroot ./datasets/facades --direction BtoA --model pix2pix --name facades_pix2pix
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
---------------- Options ---------------
             aspect_ratio: 1.0
               batch_size: 1
          checkpoints_dir: ./checkpoints
                crop_size: 256
                 dataroot: ./datasets/facades                  [default: None]
             dataset_mode: aligned
                direction: BtoA
          display_winsize: 256
                    epoch: latest
                     eval: False
                  gpu_ids: 0
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: False                               [default: None]
                load_iter: 0                                   [default: 0]
                load_size: 256
         max_dataset_size: inf
                    model: pix2pix                             [default: test]
               n_layers_D: 3
                     name: facades_pix2pix
                      ndf: 64
                     netD: basic
                     netG: unet_256
                      ngf: 64
               no_dropout: False
                  no_flip: False
                     norm: batch
                 num_test: 50
              num_threads: 4
                output_nc: 3
                    phase: test
               preprocess: resize_and_crop
              results_dir: ./results/
           serial_batches: False
                   suffix:
                use_wandb: False
                  verbose: False
----------------- End -------------------
dataset [AlignedDataset] was created
initialize network with normal
model [Pix2PixModel] was created
loading the model from ./checkpoints/facades_pix2pix/latest_net_G.pth
---------- Networks initialized -------------
[Network G] Total number of parameters : 54.414 M
-----------------------------------------------
creating web directory ./results/facades_pix2pix/test_latest
/home/user/miniconda/envs/pytorch-CycleGAN-and-pix2pix/lib/python3.6/site-packages/torchvision/transforms/transforms.py:288: UserWarning: Argument interpolation should be
  "Argument interpolation should be of type InterpolationMode instead of int. "
processing (0000)-th image... ['./datasets/facades/test/1.jpg']
processing (0005)-th image... ['./datasets/facades/test/103.jpg']
processing (0010)-th image... ['./datasets/facades/test/12.jpg']
processing (0015)-th image... ['./datasets/facades/test/17.jpg']
```

Images are then created in the result folder as follows.

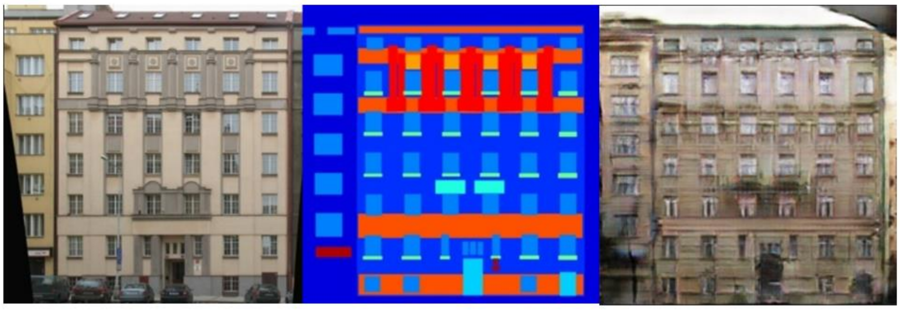


2.2. Some result images including generated images using pix2pix.

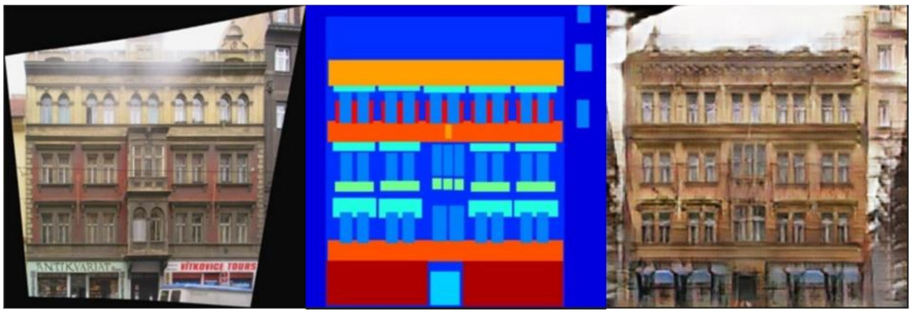
The training data set has the following image structure.



Train A **Train B**



Real A **Real B** **Fake A**



Real A **Real B** **Fake A**

Real B **Real A** **Fake B**

2.3. What did you learn through this problem #2.

I learned pix2pix structure and learning method for paired image-to-image translation. And I learned how to configure the paired image set for paired image-to-image translation learning. I learned about reflecting learning changes according to paired image characteristics.

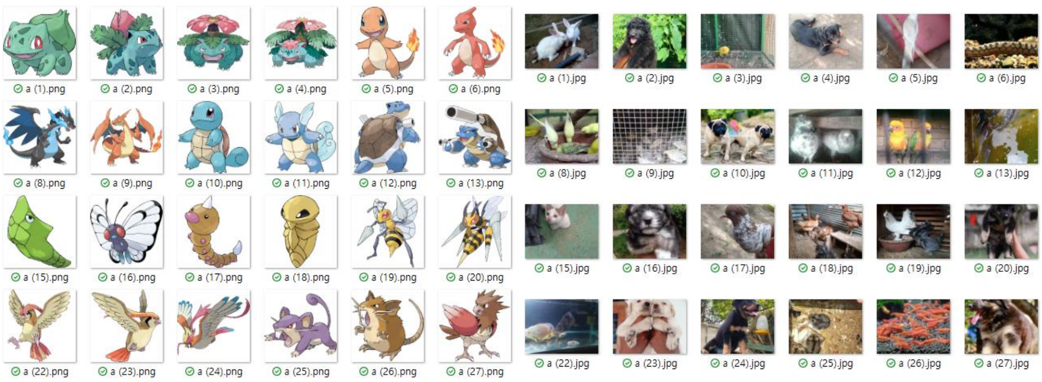
2.4. Discuss about the experimental results, network architecture, and training techniques.

In the case of facades dataset, the processing image B and A characterizing the window and door of the real image are translated into B, so it can be confirmed that the actual image A is converted to B well according to the characteristics of the window and door. However, it was confirmed that the image generated during the opposite conversion was similar to image B in terms of the number of windows, doors, and appearance structure, but the texture and color of the exterior of the building were not learned as similar to A. In this case, the characteristics of the paired images are well learned in the pix2pix learning structure. In other words, train B set is an image that structures the characteristics of windows and doors well, so if we convert it to A->B, I think I can clearly express the characteristics, but when converting to B->A, the train A data can be used as a whole. The characteristics of a building and its exterior structure, windows, doors, etc. are well learned and expressed, but the color and texture of the building. It was confirmed that it was not learned up to the characteristics that were difficult to learn

2.5. Create your own idea and show the implementation results.

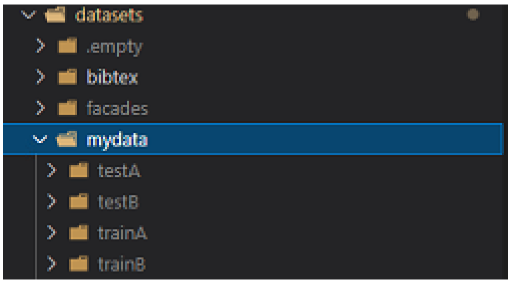
- **Collect the dataset for your idea.**

I collected data through googling to use pokemon which is from animation and animals as dataset. I prepared 500 training data and 150 test data.



- **Implement the code that realizes your idea.**

Mydata folder was created in ./datasets as follows to organize training data and test data.



In order to use pix2pix, it is necessary to combine each data into one. Therefore, the process of combining data into one using "combine_A_and_B.py" was carried out as follows.

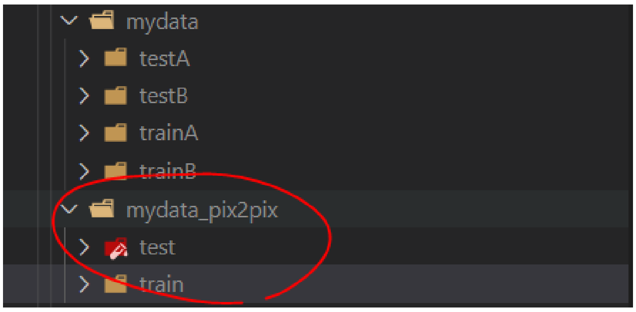
```python
parser = argparse.ArgumentParser('create image pairs')
parser.add_argument('--fold_A', dest='fold_A', help='input directory for image A', type=str, default='/home/DL/assn5/p2/datasets/mydata/trainA')
parser.add_argument('--fold_B', dest='fold_B', help='input directory for image B', type=str, default='/home/DL/assn5/p2/datasets/mydata/trainB')
parser.add_argument('--fold_AB', dest='fold_AB', help='output directory', type=str, default='/home/DL/assn5/p2/datasets/mydata_pix2pix')
parser.add_argument('--num_imgs', dest='num_imgs', help='number of images', type=int, default=100000)
parser.add_argument('--use_AB', dest='use_AB', help='if true: (0001_A, 0001_B) to (0001_AB)', action='store_true')
parser.add_argument('--no_multiprocessing', dest='no_multiprocessing', help='If used, chooses single CPU execution instead of parallel execution', action='store_true',default=True)
args = parser.parse_args()
```

```python
parser = argparse.ArgumentParser('create image pairs')
parser.add_argument('--fold_A', dest='fold_A', help='input directory for image A', type=str, default='/home/DL/assn5/p2/datasets/mydata/testA')
parser.add_argument('--fold_B', dest='fold_B', help='input directory for image B', type=str, default='/home/DL/assn5/p2/datasets/mydata/testB')
parser.add_argument('--fold_AB', dest='fold_AB', help='output directory', type=str, default='/home/DL/assn5/p2/datasets/mydata_pix2pix')
parser.add_argument('--num_imgs', dest='num_imgs', help='number of images', type=int, default=100000)
parser.add_argument('--use_AB', dest='use_AB', help='if true: (0001_A, 0001_B) to (0001_AB)', action='store_true')
parser.add_argument('--no_multiprocessing', dest='no_multiprocessing', help='If used, chooses single CPU execution instead of parallel execution', action='store_true',default=True)
args = parser.parse_args()
```

The extensions of the data I collected were different from .jpg and .png, so I needed to modify the existing code.

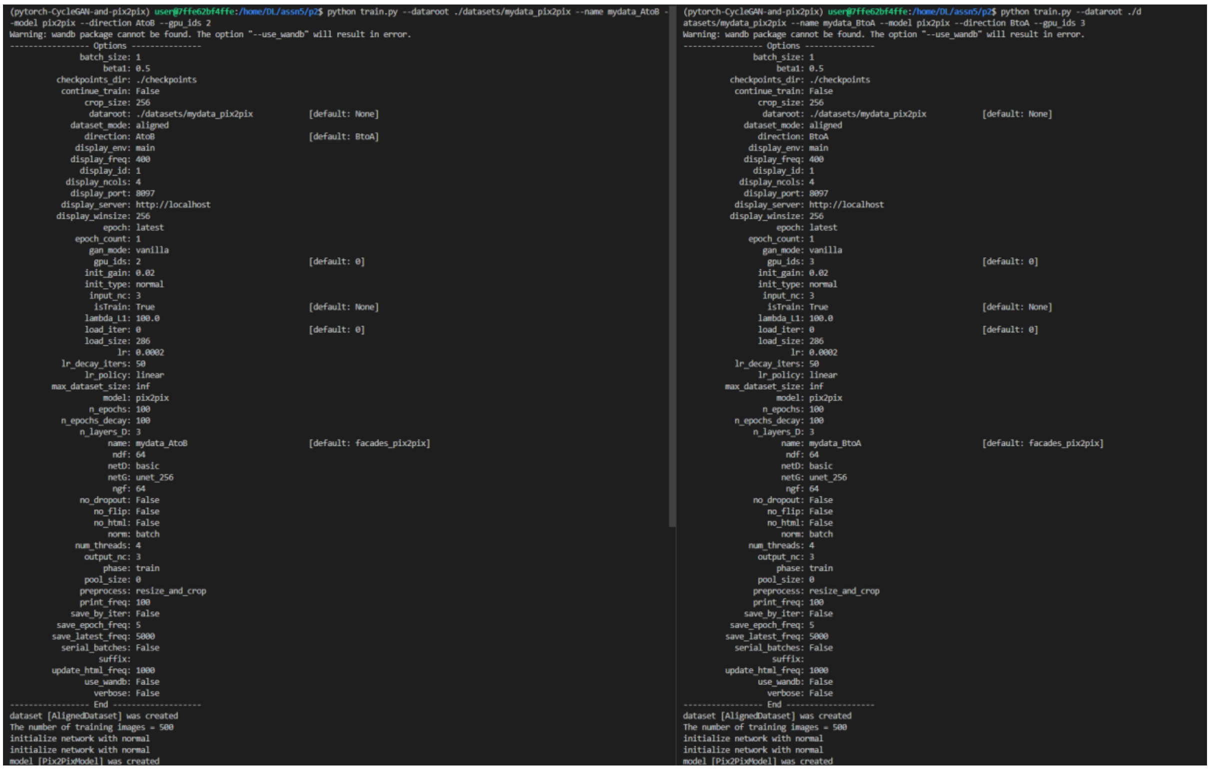
```python
for sp in splits:
    img_fold_A = os.path.join(args.fold_A, sp)
    img_fold_B = os.path.join(args.fold_B, sp)
    img_list = os.listdir(img_fold_A)
    if args.use_AB:
        img_list = [img_path for img_path in img_list if '_A.' in img_path]

    num_imgs = min(args.num_imgs, len(img_list))
    print('split = %s, use %d/%d images' % (sp, num_imgs, len(img_list)))
    img_fold_AB = os.path.join(args.fold_AB, sp)
    if not os.path.isdir(img_fold_AB):
        os.makedirs(img_fold_AB)
    print('split = %s, number of images = %d' % (sp, num_imgs))

    for n in range(num_imgs):
        name_A = img_list[n]
        path_A = os.path.join(img_fold_A, name_A)
        if args.use_AB:
            name_B = name_A.replace('_A.', '_B.')
        else:
            name_B = name_A
        name_B = name_B[:-4] + '.jpg'
        path_B = os.path.join(img_fold_B, name_B)
        if os.path.isfile(path_A) and os.path.isfile(path_B):
            name_AB = name_A
            if args.use_AB:
                name_AB = name_AB.replace('_A.', '.')  # remove _A
            path_AB = os.path.join(img_fold_AB, name_AB)
            if not args.no_multiprocessing:
                pool.apply_async(image_write, args=(path_A, path_B, path_AB))
            else:
                im_A = cv2.imread(path_A, 1) # python2: cv2.CV_LOAD_IMAGE_COLOR; python3: cv2.IMREAD_COLOR
                im_B = cv2.imread(path_B, 1) # python2: cv2.CV_LOAD_IMAGE_COLOR; python3: cv2.IMREAD_COLOR
                im_B = cv2.resize(im_B, (256, 256))
                im_AB = np.concatenate([im_A, im_B], 1)
                cv2.imwrite(path_AB, im_AB)
if not args.no_multiprocessing:
    pool.close()
    pool.join()
```

And when I run "combine_A_and_B.py", the data is combined as follows.



500 training data and 150 test data exist as a pair as follows.

And in order to train, I ran "train.py" in both directions.



```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$ python train.py --dataroot ./datasets/mydata_pix2pix --name mydata_AtoB -
-model pix2pix --direction AtoB --gpu_ids 2
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: False
                crop_size: 256
                 dataroot: ./datasets/mydata_pix2pix           [default: None]
             dataset_mode: aligned
                direction: AtoB                                [default: BtoA]
              display_env: main
             display_freq: 400
               display_id: 1
            display_ncols: 4
             display_port: 8097
           display_server: http://localhost
          display_winsize: 256
                    epoch: latest
              epoch_count: 1
                 gan_mode: vanilla
                  gpu_ids: 2                                   [default: 0]
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: True                                [default: None]
                lambda_L1: 100.0
                load_iter: 0                                   [default: 0]
                load_size: 286
                       lr: 0.0002
           lr_decay_iters: 50
                lr_policy: linear
         max_dataset_size: inf
                    model: pix2pix
                 n_epochs: 100
           n_epochs_decay: 100
               n_layers_D: 3
                     name: mydata_AtoB                         [default: facades_pix2pix]
                      ndf: 64
                     netD: basic
                     netG: unet_256
                      ngf: 64
               no_dropout: False
                  no_flip: False
                  no_html: False
                     norm: batch
              num_threads: 4
                output_nc: 3
                    phase: train
                pool_size: 0
               preprocess: resize_and_crop
               print_freq: 100
             save_by_iter: False
          save_epoch_freq: 5
         save_latest_freq: 5000
           serial_batches: False
                   suffix:
         update_html_freq: 1000
                use_wandb: False
                  verbose: False
----------------- End -------------------
dataset [AlignedDataset] was created
The number of training images = 500
initialize network with normal
initialize network with normal
model [Pix2PixModel] was created
```

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$ python train.py --dataroot ./d
atasets/mydata_pix2pix --name mydata_BtoA --model pix2pix --direction BtoA --gpu_ids 3
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: False
                crop_size: 256
                 dataroot: ./datasets/mydata_pix2pix           [default: None]
             dataset_mode: aligned
                direction: BtoA
              display_env: main
             display_freq: 400
               display_id: 1
            display_ncols: 4
             display_port: 8097
           display_server: http://localhost
          display_winsize: 256
                    epoch: latest
              epoch_count: 1
                 gan_mode: vanilla
                  gpu_ids: 3                                   [default: 0]
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: True                                [default: None]
                lambda_L1: 100.0
                load_iter: 0                                   [default: 0]
                load_size: 286
                       lr: 0.0002
           lr_decay_iters: 50
                lr_policy: linear
         max_dataset_size: inf
                    model: pix2pix
                 n_epochs: 100
           n_epochs_decay: 100
               n_layers_D: 3
                     name: mydata_BtoA                         [default: facades_pix2pix]
                      ndf: 64
                     netD: basic
                     netG: unet_256
                      ngf: 64
               no_dropout: False
                  no_flip: False
                  no_html: False
                     norm: batch
              num_threads: 4
                output_nc: 3
                    phase: train
                pool_size: 0
               preprocess: resize_and_crop
               print_freq: 100
             save_by_iter: False
          save_epoch_freq: 5
         save_latest_freq: 5000
           serial_batches: False
                   suffix:
         update_html_freq: 1000
                use_wandb: False
                  verbose: False
----------------- End -------------------
dataset [AlignedDataset] was created
The number of training images = 500
initialize network with normal
initialize network with normal
model [Pix2PixModel] was created
```

For test, I ran "test.py" in both directions.

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$ python test.py --dataroot ./datasets/mydata_pix2pix --name mydata_A
toB --model pix2pix --direction AtoB
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               aspect_ratio: 1.0
                 batch_size: 1
            checkpoints_dir: ./checkpoints
                  crop_size: 256
                   dataroot: ./datasets/mydata_pix2pix          [default: None]
               dataset_mode: aligned
                  direction: AtoB                               [default: BtoA]
            display_winsize: 256
                      epoch: latest
                       eval: False
                    gpu_ids: 0
                  init_gain: 0.02
                  init_type: normal
                   input_nc: 3
                    isTrain: False                              [default: None]
                  load_iter: 0                                  [default: 0]
                  load_size: 256
           max_dataset_size: inf
                      model: pix2pix                            [default: test]
                 n_layers_D: 3
                       name: mydata_AtoB                        [default: facades_pix2pix]
                        ndf: 64
                       netD: basic
                       netG: unet_256
                        ngf: 64
                 no_dropout: False
                    no_flip: False
                       norm: batch
                   num_test: 50
                num_threads: 4
                  output_nc: 3
                      phase: test
                 preprocess: resize_and_crop
                results_dir: ./results/
             serial_batches: False
                     suffix:
                  use_wandb: False
                    verbose: False
----------------- End -------------------
dataset [AlignedDataset] was created
initialize network with normal
model [Pix2PixModel] was created
loading the model from ./checkpoints/mydata_AtoB/latest_net_G.pth
---------- Networks initialized -------------
[Network G] Total number of parameters : 54.414 M
-----------------------------------------------
creating web directory ./results/mydata_AtoB/test_latest
/home/user/miniconda/envs/pytorch-CycleGAN-and-pix2pix/lib/python3.6/site-packages/torchvision/transforms/transforms.py:288: UserWarnin
g: Argument interpolation should be of type InterpolationMode instead of int. Please, use InterpolationMode enum.
  "Argument interpolation should be of type InterpolationMode instead of int. "
processing (0000)-th image... ['./datasets/mydata_pix2pix/test/b (1).png']
processing (0005)-th image... ['./datasets/mydata_pix2pix/test/b (103).png']
processing (0010)-th image... ['./datasets/mydata_pix2pix/test/b (108).png']
processing (0015)-th image... ['./datasets/mydata_pix2pix/test/b (112).png']
processing (0020)-th image... ['./datasets/mydata_pix2pix/test/b (117).png']
processing (0025)-th image... ['./datasets/mydata_pix2pix/test/b (121).png']
processing (0030)-th image... ['./datasets/mydata_pix2pix/test/b (126).png']
processing (0035)-th image... ['./datasets/mydata_pix2pix/test/b (130).png']
processing (0040)-th image... ['./datasets/mydata_pix2pix/test/b (135).png']
processing (0045)-th image... ['./datasets/mydata_pix2pix/test/b (14).png']
```

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p2$ python test.py --dataroot ./datasets/mydata_pix2pix --name mydata_BtoA --model pix
2pix --direction BtoA
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               aspect_ratio: 1.0
                 batch_size: 1
            checkpoints_dir: ./checkpoints
                  crop_size: 256
                   dataroot: ./datasets/mydata_pix2pix          [default: None]
               dataset_mode: aligned
                  direction: BtoA
            display_winsize: 256
                      epoch: latest
                       eval: False
                    gpu_ids: 0
                  init_gain: 0.02
                  init_type: normal
                   input_nc: 3
                    isTrain: False                              [default: None]
                  load_iter: 0                                  [default: 0]
                  load_size: 256
           max_dataset_size: inf
                      model: pix2pix                            [default: test]
                 n_layers_D: 3
                       name: mydata_BtoA                        [default: facades_pix2pix]
                        ndf: 64
                       netD: basic
                       netG: unet_256
                        ngf: 64
                 no_dropout: False
                    no_flip: False
                       norm: batch
                   num_test: 50
                num_threads: 4
                  output_nc: 3
                      phase: test
                 preprocess: resize_and_crop
                results_dir: ./results/
             serial_batches: False
                     suffix:
                  use_wandb: False
                    verbose: False
----------------- End -------------------
dataset [AlignedDataset] was created
initialize network with normal
model [Pix2PixModel] was created
loading the model from ./checkpoints/mydata_BtoA/latest_net_G.pth
---------- Networks initialized -------------
[Network G] Total number of parameters : 54.414 M
-----------------------------------------------
creating web directory ./results/mydata_BtoA/test_latest
/home/user/miniconda/envs/pytorch-CycleGAN-and-pix2pix/lib/python3.6/site-packages/torchvision/transforms/transforms.py:288: UserWarning: Argument int
erpolation should be of type InterpolationMode instead of int. Please, use InterpolationMode enum.
  "Argument interpolation should be of type InterpolationMode instead of int. "
processing (0000)-th image... ['./datasets/mydata_pix2pix/test/b (1).png']
processing (0005)-th image... ['./datasets/mydata_pix2pix/test/b (103).png']
processing (0010)-th image... ['./datasets/mydata_pix2pix/test/b (108).png']
processing (0015)-th image... ['./datasets/mydata_pix2pix/test/b (112).png']
processing (0020)-th image... ['./datasets/mydata_pix2pix/test/b (117).png']
processing (0025)-th image... ['./datasets/mydata_pix2pix/test/b (121).png']
processing (0030)-th image... ['./datasets/mydata_pix2pix/test/b (126).png']
processing (0035)-th image... ['./datasets/mydata_pix2pix/test/b (130).png']
processing (0040)-th image... ['./datasets/mydata_pix2pix/test/b (135).png']
processing (0045)-th image... ['./datasets/mydata_pix2pix/test/b (14).png']
```

- **Demonstrate the implementation results.**

Real A

Real B

Fake B

Real A

Real B

Fake A

- **Discuss about your achievement.**

As shown in the above results, in the case of my dataset, it was confirmed that the learned results were not very satisfactory. It was found that even if two pairs of data sets were tried to be well constructed, they could not form a perfectly matched pair, so they could not generate an image that reflected the characteristics of each picture, and slightly produced a general image such as DC-GAN. (Images are created similarly regardless of the input image.) In the case of pix2pix, it was confirmed that the matching pairing similar to the input image had the greatest influence on learning, except for specific parts of the input data and ground truth. In fact, it has been confirmed that many image conversion apps using pix2pix are being developed as mobile apps, and it has been confirmed that the pix2pix model has an excellent effect to apply to certain characteristic values (change to black-and-white, sketch, oil painting, cartoon, etc.).

3. [Problem #3] Training and Testing Paired Text-to-Image Synthesis

Problem #3 is about training paired text-to-image model. We can read and understand the Generative Adversarial Text-to-Image Synthesis using generator and discriminator work. We can implement the text-to-image synthesis model from github repository. Also, we can train and test the text-to-image synthesis model with Flower dataset.

- https://github.com/mirrortower/Text-to-Image-Synthesis

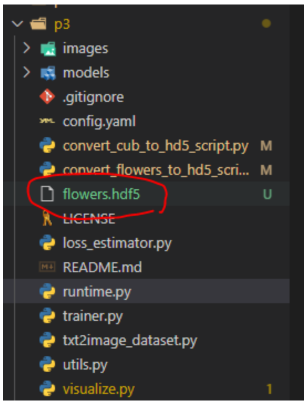
3.1. Try the efforts to improve the performance on your network. For example, your hyper-parameter setting or collecting dataset or your network improvements that are not provided by the basic codes.

First, we modified the "convert_flowers_to_hd5_script.py" file to use the flower dataset. The torch that was previously deleted from here. The "torch.utils.serialization.load_lua" module was not available, so the code was modified to use the "torchfile" module.

```
#from torch.utils.serialization import load_lua
import torchfile
```

```
for example, txt_file in zip(sorted(glob(data_path + "/*.t7")), sorted(glob(txt_path + "/*.txt"))):
    #example_data = load_lua(example)
    example_data = torchfile.load(example)
    img_path = example_data['img']
```

In order to use the flowers dataset, it was changed to hd5 format as follows. I downloaded the pre-converted hd5 data to eliminate the inconvenience of changing the image to hd5 format.



Hyper-parameter was set through github's argument explanation.

Arguments:

- `type` : GAN archiecture to use `(gan | wgan | vanilla_gan | vanilla_wgan)`. default = `gan`. Vanilla mean not conditional
- `dataset`: Dataset to use `(birds | flowers)`. default = `flowers`
- `split` : An integer indicating which split to use `(0 : train | 1: valid | 2: test)`. default = `0`
- `lr` : The learning rate. default = `0.0002`
- `diter` : Only for WGAN, number of iteration for discriminator for each iteration of the generator. default = `5`
- `vis_screen` : The visdom env name for visualization. default = `gan`
- `save_path` : Path for saving the models.
- `l1_coef` : L1 loss coefficient in the generator loss fucntion for gan and vanilla_gan. default=`50`
- `l2_coef` : Feature matching coefficient in the generator loss fucntion for gan and vanilla_gan. default=`100`
- `pre_trained_disc` : Discriminator pre-tranined model path used for intializing training.
- `pre_trained_gen` Generator pre-tranined model path used for intializing training.
- `batch_size`: Batch size. default= `64`
- `num_workers`: Number of dataloader workers used for fetching data. default = `8`
- `epochs` : Number of training epochs. default=`200`
- `cls`: Boolean flag to whether train with cls algorithms or not. default=`False`

The following are arguments in "runtime.py".

p3 > runtime.py > ...

```python
import argparse
from PIL import Image
import os

parser = argparse.ArgumentParser()
parser.add_argument("--type", default='gan')
parser.add_argument("--lr", default=0.0002, type=float)
parser.add_argument("--l1_coef", default=50, type=float)
parser.add_argument("--l2_coef", default=100, type=float)
parser.add_argument("--diter", default=5, type=int)
parser.add_argument("--cls", default=False, action='store_true')
parser.add_argument("--vis_screen", default='gan')
parser.add_argument("--save_path", default='./checkpoints/')
parser.add_argument("--inference", default=False, action='store_true')
parser.add_argument('--pre_trained_disc', default=None)
parser.add_argument('--pre_trained_gen', default=None)
parser.add_argument('--dataset', default='flowers')
parser.add_argument('--split', default=0, type=int)
parser.add_argument('--batch_size', default=64, type=int)
parser.add_argument('--num_workers', default=8, type=int)
parser.add_argument('--epochs', default=200, type=int)
args = parser.parse_args()

trainer = Trainer(type=args.type,
```

The path was modified in "config.yaml" to use the downloaded "flowers.hdf5".

```yaml
#flowers_dataset_path: '/scratch/aelnouby/text2image/flowers.hdf5'
flowers_dataset_path: './flowers.hdf5'
```

The code received from the git clone is an old code, so the updateTrace() function of visdom disappeared, so the code was modified as follows.

```
        else:
            #self.viz.updateTrace(X=np.array([x]), Y=np.array([y]), env=self.env, win=self.plots[var_name], name=split_name)
            self.viz.scatter(X=np.array([x]), Y=np.array([y]), env=self.env, win=self.plots[var_name], name=split_name, update='append')
```

Now, we can use "runtime.py" to learn.

```
(hhh) user@7ffe62bf4ffe:/home/DL/assn5/p3$ python runtime.py
/home/DL/assn5/p3/trainer.py:17: YAMLLoadWarning: calling yaml.load() without Loader=... is deprecated, as the default Loader is unsafe. Please rea
  config = yaml.load(f)
Setting up a new session...
Epoch: 0, d_loss= 1.794668, g_loss= 37.672554, D(X)= 0.650038, D(G(X))= 0.525757
Epoch: 0, d_loss= 1.942220, g_loss= 32.695656, D(X)= 0.265561, D(G(X))= 0.068352
Epoch: 0, d_loss= 2.308926, g_loss= 34.753044, D(X)= 0.218201, D(G(X))= 0.025430
Epoch: 0, d_loss= 1.155093, g_loss= 35.283421, D(X)= 0.522767, D(G(X))= 0.176589
Epoch: 0, d_loss= 1.048515, g_loss= 37.287331, D(X)= 0.687467, D(G(X))= 0.285874
Epoch: 0, d_loss= 1.356555, g_loss= 35.514931, D(X)= 0.458255, D(G(X))= 0.014313
Epoch: 0, d_loss= 1.377186, g_loss= 32.609604, D(X)= 0.809726, D(G(X))= 0.384073
```

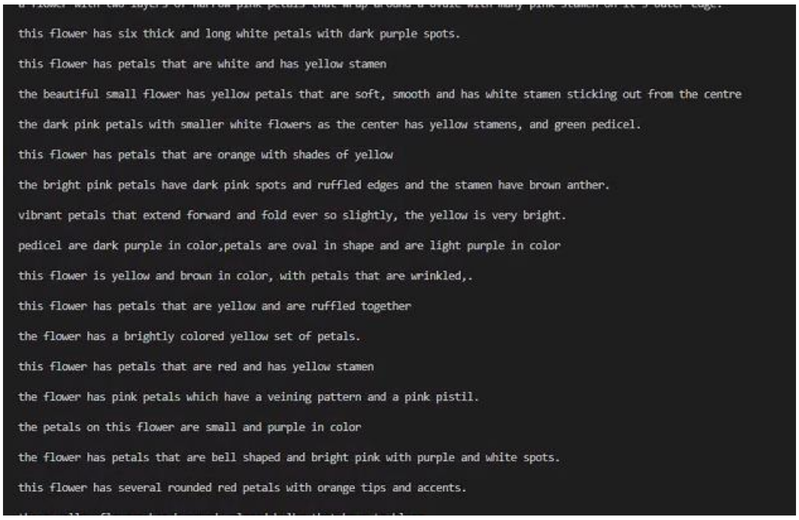
In the meantime, weights are saved every ten turns of the epoch.

```
checkpoints / checkpoints
    disc_0.pth
    disc_10.pth
    disc_20.pth
    disc_30.pth
    disc_40.pth
    disc_50.pth
    disc_60.pth
    disc_70.pth
    disc_80.pth
    disc_90.pth
    disc_100.pth
    disc_110.pth
    disc_120.pth
    disc_130.pth
    disc_140.pth
    disc_150.pth
    disc_160.pth
    disc_170.pth
    disc_180.pth
    disc_190.pth
    gen_0.pth
```

In order to test, the interference mode was changed to true and pretrained weights.

```
parser = argparse.ArgumentParser()
parser.add_argument("--type", default='gan')
parser.add_argument("--lr", default=0.0002, type=float)
parser.add_argument("--l1_coef", default=50, type=float)
parser.add_argument("--l2_coef", default=100, type=float)
parser.add_argument("--diter", default=5, type=int)
parser.add_argument("--cls", default=False, action='store_true')
parser.add_argument("--vis_screen", default='gan')
parser.add_argument("--save_path", default='./checkpoints/')
parser.add_argument("--inference", default=True, action='store_true')
parser.add_argument('--pre_trained_disc', default='./checkpoints/checkpoints/disc_190.pth')
parser.add_argument('--pre_trained_gen', default='./checkpoints/checkpoints/gen_190.pth')
parser.add_argument('--dataset', default='flowers')
parser.add_argument('--split', default=0, type=int)
parser.add_argument('--batch_size', default=64, type=int)
parser.add_argument('--num_workers', default=8, type=int)
parser.add_argument('--epochs', default=200, type=int)
args = parser.parse_args()
```

Then, the model will read the following text to generate images.



```
this flower has six thick and long white petals with dark purple spots.
this flower has petals that are white and has yellow stamen
the beautiful small flower has yellow petals that are soft, smooth and has white stamen sticking out from the centre
the dark pink petals with smaller white flowers as the center has yellow stamens, and green pedicel.
this flower has petals that are orange with shades of yellow
the bright pink petals have dark pink spots and ruffled edges and the stamen have brown anther.
vibrant petals that extend forward and fold ever so slightly, the yellow is very bright.
pedicel are dark purple in color,petals are oval in shape and are light purple in color
this flower is yellow and brown in color, with petals that are wrinkled,.
this flower has petals that are yellow and are ruffled together
the flower has a brightly colored yellow set of petals.
this flower has petals that are red and has yellow stamen
the flower has pink petals which have a veining pattern and a pink pistil.
the petals on this flower are small and purple in color
the flower has petals that are bell shaped and bright pink with purple and white spots.
this flower has several rounded red petals with orange tips and accents.
```

It can be seen that an image is created in the result folder according to each text.

results / checkpoints
{}
0outer petals are larger in size ,inner petals are smaleer n size and purple in color
2 purple, round petals, 2 pale purple petals with dark lines, and 1 yellow petal with dark lines.
5 delicate white petals that fold and turn to gold at the base.
5 purple and red pedals, with pink stamen covered in small drops of water.
5 white petals that turn yellow as they get closer to the center, in the shape of a pinwheel.
6 bright orange petals that look like a tiger stripes.
a beautiful butterfly shaped flower with large soft white petals and light orange colored stamen.
a beautiful flower with outer whorl of prominent big white sepals and the corolla contains several t.jpg
a beautiful flower with pink and fuchsia petals that seem to grow in clusters.
a beautiful flower with white and purple petals and yellow stamen
a beautiful flower with white petals and a yellow stamen
a beautiful small flower with very pretty purple and yellow petals.
a beautiful tri colored flower that is light pink and dark pink petals and a orange ovary
a beautiful white flower with fused petals forming a trumpet shape with stamen sticking out from the.jpg
a beautiful white flower with petals curved under and tan stamen and stigma sticking out from the mi.jpg
a beautiful white serrated sepaled flower has the androecium and gynoecium fused in the center
a beautiful yellow rose with a small bud in the center
a bell shaped flower that has white petals and a cream colored center
a bell shaped flower with purple spotted petals and no visible pistils.
a bell-shaped flower with red and white petals, green pedicel, and red sepals
a belly shaped flower that has tones of orange and yellow
a big flower that has white petals with pink tips, yellow big and stubby stigma, and thin yellow sta.jpg
a blood colored pistil collects together with a group of long yellow stamens around the outside whic.jpg
a blooming white flower with a lot of petals and yellow pollen tube.
a blue and white funnel shaped flower with many wrinkles.
a blue flower with a white center and a white stamen at it's core.
a blue flower with large petals, and a white center.
a blue flower with many orange and red petals at it's center.
a blue flower with pointed petals and large green sepal.
a blue flower with yellow disc floret bud and the pedicel are green
a blue flower with yellow filament and anther and the leaves are green and thin
a bowl shaped white flower with a dull yellow center.
a bowl shaped white petaled flower with yellow anther and yellow pistil.
a bright flower with large yellow petals and a wider style.
a bright lavender flower with wide slightly pointed petals, white pistil, and light yellow stamen.

3.2. Some result images including generated images using Text-to-Image synthesis model.

white petals with a yellow center

this yellow flowers have smooth petals and a bunch of stamens

this white and pale pink flower has a dark pink center

yellow petals little green leaves

violet pointed and vein showing petals with a violet and green pistil

 white and yellow ovary flower

3.3. What did you learn through this problem #3.

Through problem #3, I learned about the structure and learning method of the GAN model for text-to-image synthesis. And I learned what dataset should be used and learned for text-to-image synthesis learning. And it was very interesting to be able to create an image using text.

3.4. Discuss about the experimental results, network architecture, and training techniques.

Basically, it seems that it took a long time to learn because of the structure of creating an image from text. It seems that it took about a day to learn 200 epochs. I think I showed good results by setting the hyperparameter and learning the model by referring to the paper. In most cases, good results were produced due to good learning, but there were cases where results were somewhat inconsistent with text. I think this reason is caused by the mode collapse problem. This problem may be solved by improving cycle consistency.

4. [Problem #4] Training and Testing Unpaired Image-to-Image Translation

Problem #4 is about training CycleGAN model. We can read and understand the CycleGAN using the generator and discriminator network. We can implement the CycleGAN from github repository which is same as problem #2. We can train and test the CycleGAN network with horse-to-zebra dataset. We can download horse-to-zebra dataset from that github repository. The following shows example of horse to zebra image translation.

4.1. Try the efforts to improve the performance on your network. For example, your hyper-parameter setting or collecting dataset or your network improvements that are not provided by the basic codes.

Dataset download to be used for pix2pix model was carried out using shell script as follows.

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p4$ ./datasets/download_cyclegan_dataset.sh horse2zebra
```

Then, the horse2zebra dataset is created as follows.

```
datasets
  bibtex
  horse2zebra
    testA
    testB
    trainA
    trainB
```

To train, I ran the train.py file as follows.

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p4$ python train.py --dataroot ./datasets/horse2zebra --name horse2zebra --model cycle_gan
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: False
                crop_size: 256
                 dataroot: ./datasets/horse2zebra              [default: None]
             dataset_mode: unaligned
                direction: AtoB
              display_env: main
             display_freq: 400
               display_id: 1
            display_ncols: 4
             display_port: 8097
           display_server: http://localhost
          display_winsize: 256
                    epoch: latest
              epoch_count: 1
                 gan_mode: lsgan
                  gpu_ids: 0
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: True                                [default: None]
                 lambda_A: 10.0
                 lambda_B: 10.0
          lambda_identity: 0.5
                load_iter: 0                                   [default: 0]
                load_size: 286
                       lr: 0.0002
           lr_decay_iters: 50
                lr_policy: linear
         max_dataset_size: inf
                    model: cycle_gan
                 n_epochs: 100
           n_epochs_decay: 100
               n_layers_D: 3
                     name: horse2zebra                         [default: experiment_name]
                      ndf: 64
                     netD: basic
                     netG: resnet_9blocks
                      ngf: 64
               no_dropout: True
                  no_flip: False
                  no_html: False
                     norm: instance
              num_threads: 4
                output_nc: 3
                    phase: train
                pool_size: 50
               preprocess: resize_and_crop
               print_freq: 100
             save_by_iter: False
          save_epoch_freq: 5
         save_latest_freq: 5000
           serial_batches: False
                   suffix: 
         update_html_freq: 1000
                use_wandb: False
                  verbose: False
----------------- End -------------------
```

As learning progresses, the results are shown for each epoch as follows.

```
End of epoch 21 / 200 	 Time Taken: 185 sec
learning rate 0.0002000 -> 0.0002000
(epoch: 22, iters: 86, time: 0.158, data: 0.003) D_A: 0.183 G_A: 0.296 cycle_A: 1.032 idt_A: 0.533 D_B: 0.206 G_B: 0.288 cycle_B: 1.556 idt_B: 0.440
(epoch: 22, iters: 186, time: 0.129, data: 0.002) D_A: 0.051 G_A: 0.668 cycle_A: 1.046 idt_A: 0.372 D_B: 0.100 G_B: 0.455 cycle_B: 1.206 idt_B: 0.384
(epoch: 22, iters: 286, time: 0.125, data: 0.002) D_A: 0.322 G_A: 0.488 cycle_A: 1.336 idt_A: 0.391 D_B: 0.175 G_B: 0.105 cycle_B: 1.294 idt_B: 0.476
(epoch: 22, iters: 386, time: 0.568, data: 0.002) D_A: 0.231 G_A: 0.341 cycle_A: 1.107 idt_A: 0.721 D_B: 0.357 G_B: 0.375 cycle_B: 1.197 idt_B: 0.571
(epoch: 22, iters: 486, time: 0.129, data: 0.003) D_A: 0.229 G_A: 0.198 cycle_A: 1.306 idt_A: 0.377 D_B: 0.191 G_B: 0.514 cycle_B: 0.964 idt_B: 0.513
(epoch: 22, iters: 586, time: 0.127, data: 0.003) D_A: 0.180 G_A: 0.258 cycle_A: 1.356 idt_A: 0.530 D_B: 0.274 G_B: 0.882 cycle_B: 1.608 idt_B: 0.577
(epoch: 22, iters: 686, time: 0.133, data: 0.003) D_A: 0.238 G_A: 0.245 cycle_A: 1.591 idt_A: 0.585 D_B: 0.195 G_B: 0.335 cycle_B: 1.375 idt_B: 0.659
(epoch: 22, iters: 786, time: 0.310, data: 0.002) D_A: 0.124 G_A: 0.419 cycle_A: 2.366 idt_A: 0.484 D_B: 0.065 G_B: 0.536 cycle_B: 0.925 idt_B: 1.160
(epoch: 22, iters: 886, time: 0.127, data: 0.002) D_A: 0.159 G_A: 0.170 cycle_A: 1.174 idt_A: 0.524 D_B: 0.516 G_B: 0.734 cycle_B: 1.370 idt_B: 0.483
(epoch: 22, iters: 986, time: 0.131, data: 0.003) D_A: 0.281 G_A: 0.343 cycle_A: 1.386 idt_A: 0.539 D_B: 0.291 G_B: 0.134 cycle_B: 1.242 idt_B: 0.553
(epoch: 22, iters: 1086, time: 0.129, data: 0.002) D_A: 0.080 G_A: 0.455 cycle_A: 1.203 idt_A: 0.566 D_B: 0.214 G_B: 0.196 cycle_B: 1.462 idt_B: 0.562
(epoch: 22, iters: 1186, time: 0.316, data: 0.002) D_A: 0.055 G_A: 1.220 cycle_A: 1.502 idt_A: 0.362 D_B: 0.135 G_B: 0.441 cycle_B: 0.947 idt_B: 0.564
(epoch: 22, iters: 1286, time: 0.155, data: 0.002) D_A: 0.097 G_A: 0.504 cycle_A: 1.398 idt_A: 0.780 D_B: 0.244 G_B: 0.652 cycle_B: 2.095 idt_B: 0.622
End of epoch 22 / 200 	 Time Taken: 176 sec
learning rate 0.0002000 -> 0.0002000
(epoch: 23, iters: 52, time: 0.139, data: 0.004) D_A: 0.095 G_A: 0.601 cycle_A: 1.326 idt_A: 0.489 D_B: 0.371 G_B: 0.936 cycle_B: 1.042 idt_B: 0.457
(epoch: 23, iters: 152, time: 0.132, data: 0.002) D_A: 0.048 G_A: 0.220 cycle_A: 1.484 idt_A: 0.544 D_B: 0.144 G_B: 0.523 cycle_B: 1.401 idt_B: 0.878
(epoch: 23, iters: 252, time: 0.673, data: 0.002) D_A: 0.121 G_A: 0.960 cycle_A: 1.300 idt_A: 0.557 D_B: 0.298 G_B: 0.402 cycle_B: 1.187 idt_B: 0.533
(epoch: 23, iters: 352, time: 0.161, data: 0.002) D_A: 0.225 G_A: 0.193 cycle_A: 1.362 idt_A: 0.577 D_B: 0.088 G_B: 0.906 cycle_B: 1.271 idt_B: 0.654
(epoch: 23, iters: 452, time: 0.124, data: 0.002) D_A: 0.134 G_A: 0.600 cycle_A: 1.143 idt_A: 0.447 D_B: 0.131 G_B: 0.467 cycle_B: 0.976 idt_B: 0.488
(epoch: 23, iters: 552, time: 0.131, data: 0.003) D_A: 0.032 G_A: 0.671 cycle_A: 1.440 idt_A: 0.512 D_B: 0.144 G_B: 0.732 cycle_B: 1.558 idt_B: 0.568
(epoch: 23, iters: 652, time: 0.568, data: 0.004) D_A: 0.185 G_A: 0.323 cycle_A: 0.877 idt_A: 0.638 D_B: 0.080 G_B: 0.254 cycle_B: 1.315 idt_B: 0.401
saving the latest model (epoch 23, total_iters 30000)
(epoch: 23, iters: 752, time: 0.128, data: 0.002) D_A: 0.116 G_A: 0.727 cycle_A: 1.046 idt_A: 0.430 D_B: 0.215 G_B: 0.318 cycle_B: 1.328 idt_B: 0.403
(epoch: 23, iters: 852, time: 0.125, data: 0.003) D_A: 0.088 G_A: 0.467 cycle_A: 1.235 idt_A: 0.396 D_B: 0.077 G_B: 0.472 cycle_B: 1.119 idt_B: 0.623
```

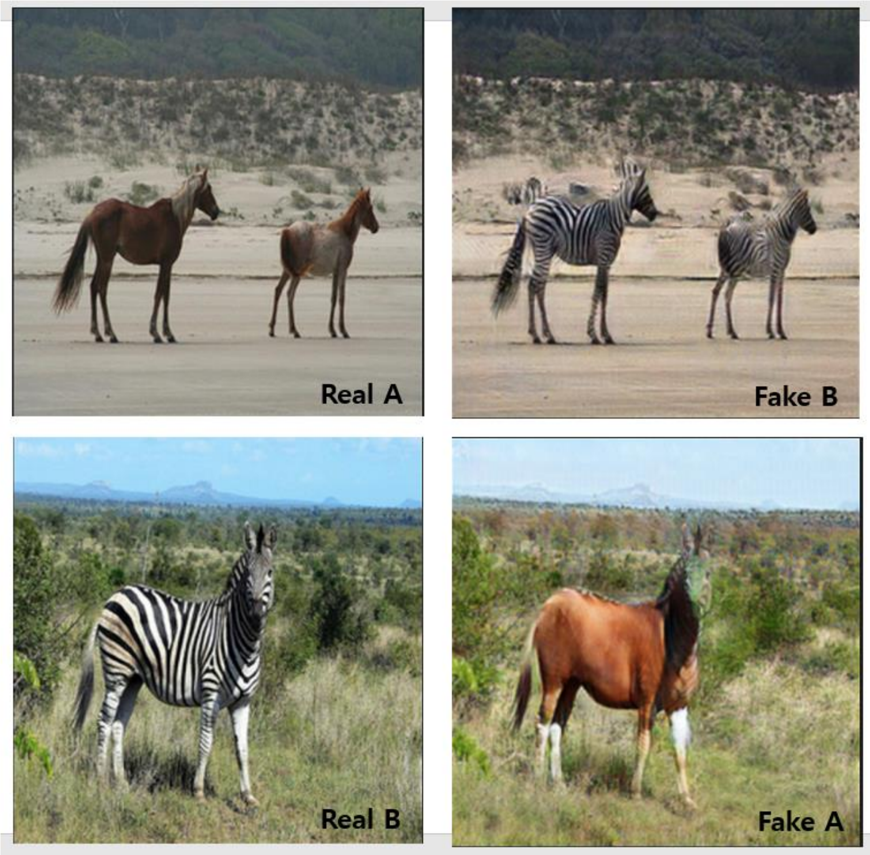
Training was also conducted in the opposite direction.

```
x2pix) user@7ffe62bf4ffe:/home/DL/assn5/p4$ python train.py --dataroot ./datasets/horse2zebra --name horse2zebra_BtoA --model cycle_gan --direction BtoA --gpu_ids 2
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: False
                crop_size: 256
                 dataroot: ./datasets/horse2zebra          [default: None]
             dataset_mode: unaligned
                direction: BtoA                            [default: AtoB]
              display_env: main
             display_freq: 400
               display_id: 1
            display_ncols: 4
             display_port: 8097
           display_server: http://localhost
          display_winsize: 256
                    epoch: latest
              epoch_count: 1
                 gan_mode: lsgan
                  gpu_ids: 2                               [default: 0]
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: True                            [default: None]
                 lambda_A: 10.0
                 lambda_B: 10.0
          lambda_identity: 0.5
                load_iter: 0                               [default: 0]
                load_size: 286
                       lr: 0.0002
           lr_decay_iters: 50
                lr_policy: linear
         max_dataset_size: inf
                    model: cycle_gan
                 n_epochs: 100
           n_epochs_decay: 100
               n_layers_D: 3
                     name: horse2zebra_BtoA                [default: experiment_name]
                      ndf: 64
                     netD: basic
                     netG: resnet_9blocks
                      ngf: 64
               no_dropout: True
                  no_flip: False
                  no_html: False
                     norm: instance
              num_threads: 4
                output_nc: 3
                    phase: train
                pool_size: 50
               preprocess: resize_and_crop
               print_freq: 100
             save_by_iter: False
          save_epoch_freq: 5
         save_latest_freq: 5000
           serial_batches: False
                   suffix: 
         update_html_freq: 1000
                use_wandb: False
                  verbose: False
----------------- End -------------------
/home/user/miniconda/envs/pytorch-CycleGAN-and-pix2pix/lib/python3.6/site-packages/torchvision/transforms/transforms.py:288: UserWarning: Argument interpolation should be o
ad of int. Please, use InterpolationMode enum.
  "Argument interpolation should be of type InterpolationMode instead of int. "
dataset [UnalignedDataset] was created
The number of training images = 1334
initialize network with normal
initialize network with normal
initialize network with normal
initialize network with normal
model [CycleGANModel] was created
---------- Networks initialized -------------
[Network G_A] Total number of parameters : 11.378 M
[Network G_B] Total number of parameters : 11.378 M
```

To test, I ran the test.py file as follows.

```
(pytorch-CycleGAN-and-pix2pix) user@7ffe62bf4ffe:/home/DL/assn5/p4$ python test.py --dataroot ./datasets/horse2zebra --name horse2zebra_AtoB --model cycle_gan --no_dropout --direction AtoB
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
Warning: wandb package cannot be found. The option "--use_wandb" will result in error.
----------------- Options ---------------
               aspect_ratio: 1.0
                 batch_size: 1
            checkpoints_dir: ./checkpoints
                  crop_size: 256
                   dataroot: ./datasets/horse2zebra             [default: None]
               dataset_mode: unaligned
                  direction: AtoB
            display_winsize: 256
                      epoch: latest
                       eval: False
                    gpu_ids: 0
                  init_gain: 0.02
                  init_type: normal
                   input_nc: 3
                    isTrain: False                              [default: None]
                  load_iter: 0                                  [default: 0]
                  load_size: 256
           max_dataset_size: inf
                      model: cycle_gan                          [default: test]
                 n_layers_D: 3
                       name: horse2zebra_AtoB                   [default: experiment_name]
                        ndf: 64
                       netD: basic
                       netG: resnet_9blocks
                        ngf: 64
                 no_dropout: True
                    no_flip: False
                       norm: instance
                   num_test: 50
                num_threads: 4
                  output_nc: 3
                      phase: test
                 preprocess: resize_and_crop
                results_dir: ./results/
             serial_batches: False
                     suffix:
                  use_wandb: False
                    verbose: False
----------------- End -------------------
/home/user/miniconda/envs/pytorch-CycleGAN-and-pix2pix/lib/python3.6/site-packages/torchvision/transforms/transforms.py:288: UserWarning: Argument interpolation should be of type Interpolati
  "Argument interpolation should be of type InterpolationMode instead of int. "
dataset [UnalignedDataset] was created
initialize network with normal
initialize network with normal
model [CycleGANModel] was created
loading the model from ./checkpoints/horse2zebra_AtoB/latest_net_G_A.pth
loading the model from ./checkpoints/horse2zebra_AtoB/latest_net_G_B.pth
---------- Networks initialized -------------
[Network G_A] Total number of parameters : 11.378 M
[Network G_B] Total number of parameters : 11.378 M
-----------------------------------------------
creating web directory ./results/horse2zebra_AtoB/test_latest
processing (0000)-th image... ['./datasets/horse2zebra/testA/n02381460_1000.jpg']
```

4.2. Some result images including generated images using CycleGAN.


Real A
Fake B
Real B
Fake A


Real A
Fake B
Real B
Fake A

4.3. What did you learn through this problem #4.

I learned about CycleGAN structure and learning method for unpaired image to image translation. And I learned how to configure the unpaired image set for unpaired image to image translation learning. Also I learned reflecting of learning change according to unpaired image characteristics.

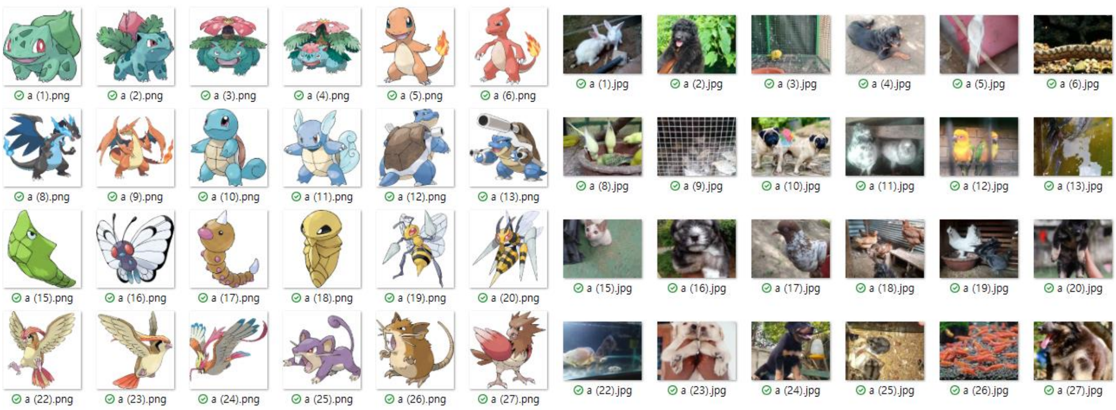
4.4. Discuss about the experimental results, network architecture, and training techniques.

Unlike pix2pix, which is a paired image to image translation, the CycleGAN uses unpaired data as a learning set, so it was much more free to configure data for training, and it was confirmed that image generation was good regardless of the characteristics of the two paired data during the test. In particular, it is thought that this algorithm, which is much more useful, can be used as it can save considerable time compared to pix2pix in the part of creating the learning data set. However, while pix2pix takes very little time to learn, CycleGAN has a disadvantage that it takes a lot of time to learn If speed is not important, cycle rather than pix2fix. I think it is more useful to use GAN. When learning the same dataset (about 5000 with 200 epochs), pix2pix takes about 3 hours, whereas Cycle GAN takes about 12 hours.

4.5. Create your own idea and show the implementation results.

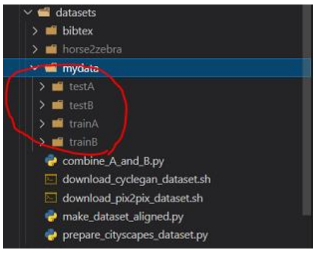
- **Collect the dataset for your idea.**

I collected data through googling to use pokemon which is from animation and animals as dataset. I prepared 500 training data and 150 test data. The same data were prepared to compare the results of Problem #2 with the performance of CycleGAN.

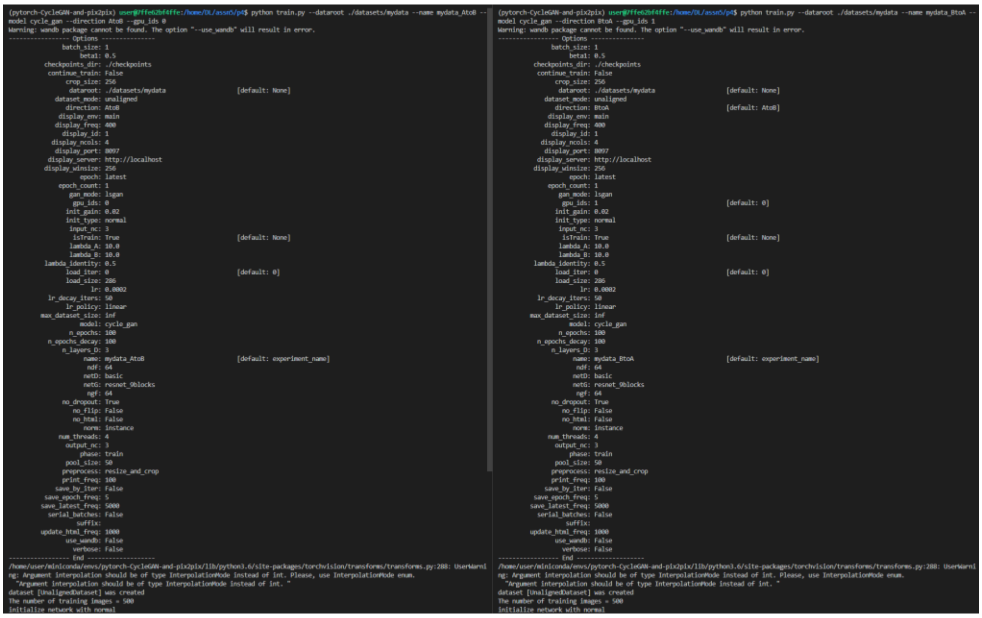


- **Implement the code that realizes your idea.**

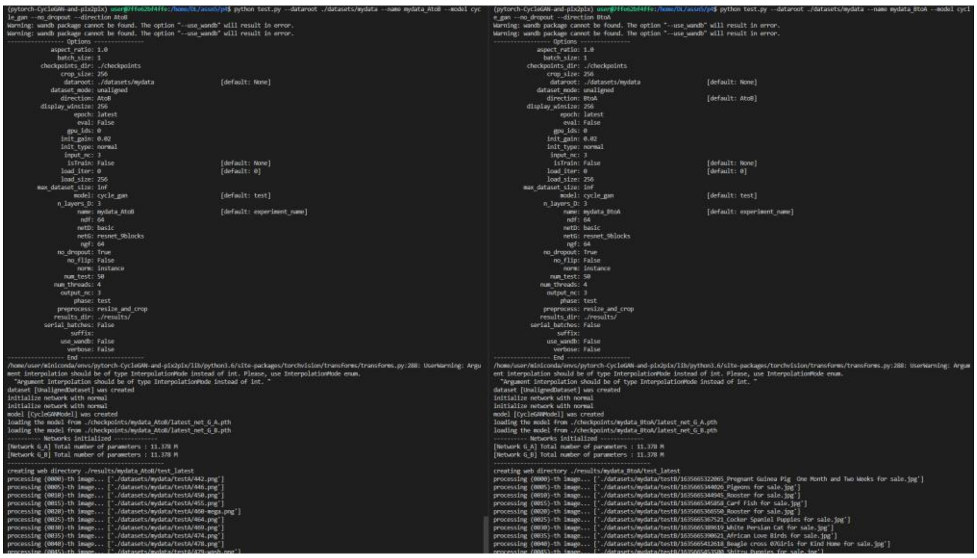
However, here, I don't have to pair it up, and I can proceed with training by dividing it into folders as follows.



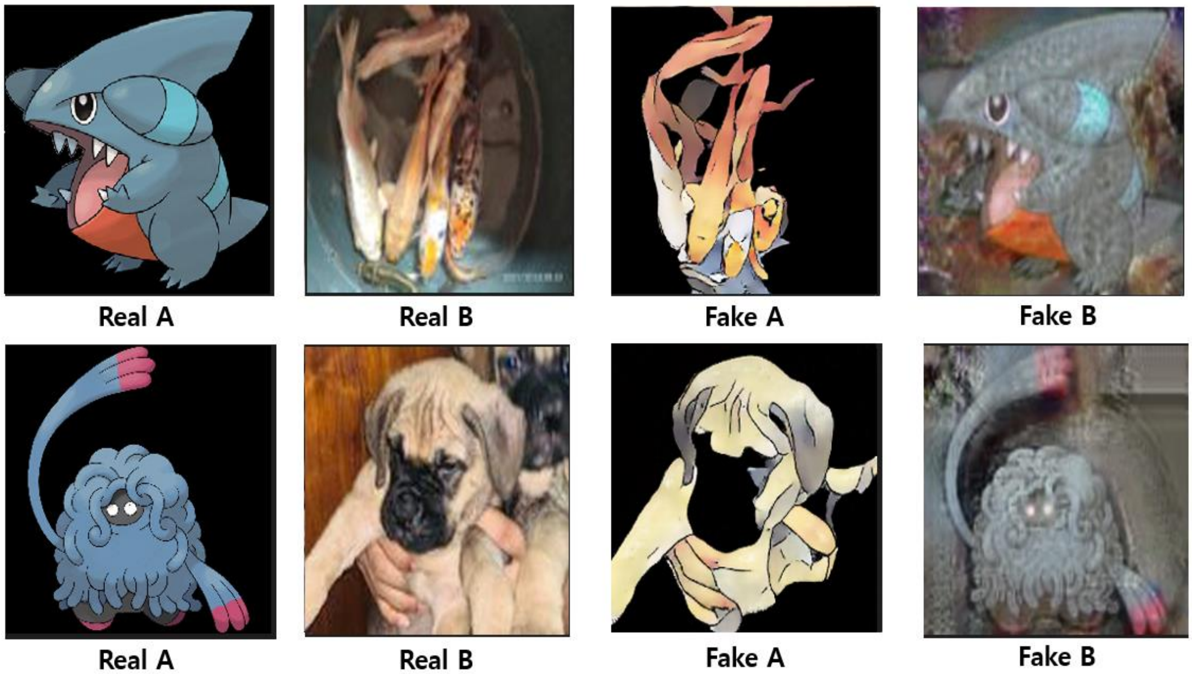
Again, training was conducted in both directions using "train.py".



And test was conducted in both directions using “test.py”



- **Demonstrate the implementation results.**



- **Discuss about your achievement.**

The paired image-to-image translation, pix2pix, is the input image for the unpaired image dataset. Unlike the fact that the characteristics of the input image were not reflected at all when the ground truth was reflected, CycleGAN verified that an image whose characteristics were well reflected by reflecting the characteristics of the input image was generated and that the performance was excellent. In fact, it can be difficult to say that it went completely well because there is a feeling that learning data was made too forcibly. However, I think it was a sufficiently valuable experiment in that it showed better results than the existing pix2pix.