COMPUTER VISION AND ITS APPLICATIONS

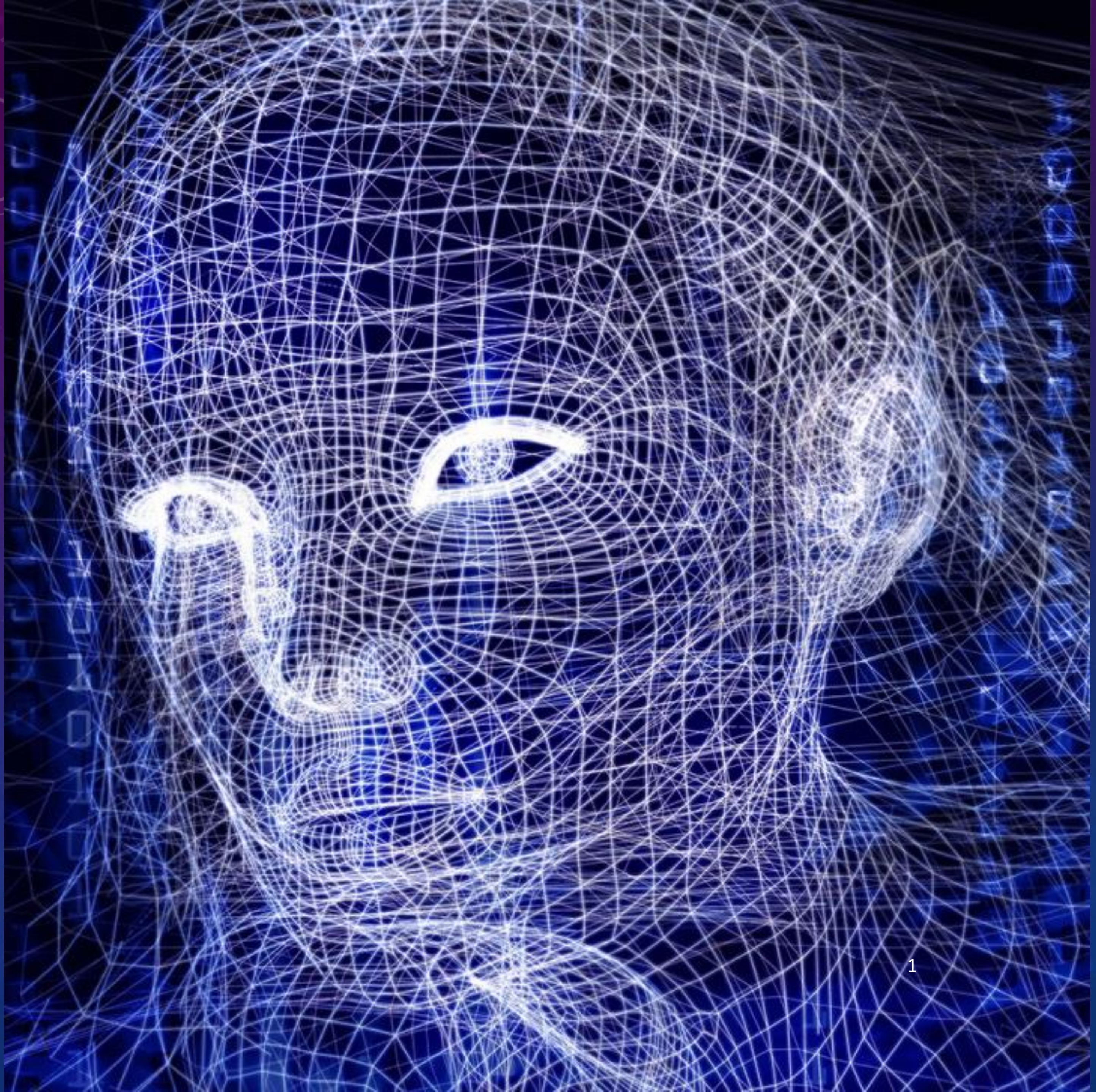
CH 5

EXAMPLE AND EXERCISE

徐繼聖

Gee-Sern Jison Hsu

National Taiwan University of Science and Technology



Example 5.1 - CGAN

- Please download the "exercise_5.1_CGAN.ipynb" from Moodle.
- Upload the Notebook to GoogleColab.
- In the following pages, we will introduce the CGAN structure, show you how to train the CGAN model, and visualize the results with the specified labels.

Example 5.1 - CGAN

```
# training parameters
batch_size = 128
lr = 0.0002
train_epoch = 10
```

Define the hyperparameters

```
# data_loader
img_size = 32
transform = transforms.Compose([
        transforms.Resize(img_size),
        transforms.ToTensor(),
        transforms.Normalize([0.5], [0.5])
        ])
train_loader = torch.utils.data.DataLoader(
    datasets.MNIST('data', train=True, download=True, transform=transform),
    batch_size=batch_size, shuffle=True)
```

Download the Mnist dataset to the folder './data'

Example 5.1 - CGAN

```
G = generator(128)
D = discriminator(128)
```

Build the instance of our classes generator and discriminator model

```
G.weight_init(mean=0.0, std=0.02)
D.weight_init(mean=0.0, std=0.02)
G.cuda()
D.cuda()
```

Weight initialization

```
BCE_loss = nn.BCELoss()
```

“Binary cross-entropy” as loss function

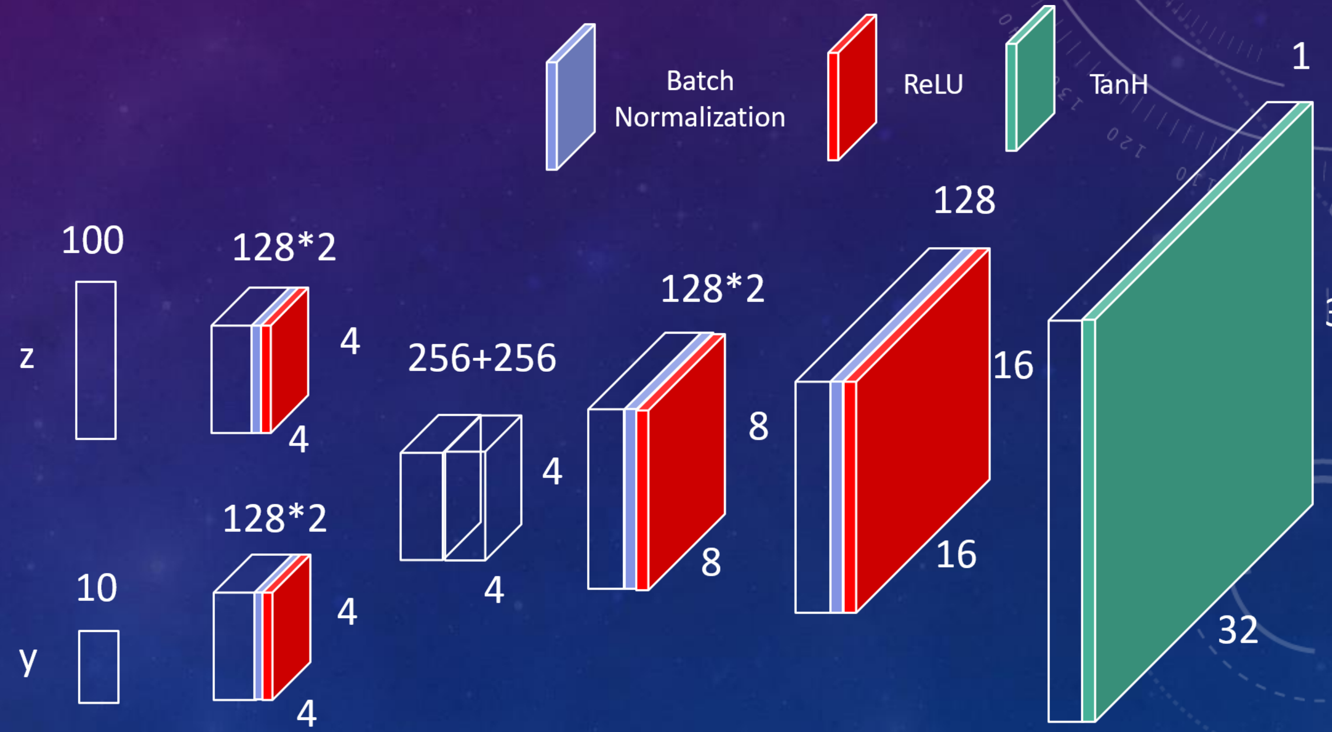
```
G_optimizer = optim.Adam(G.parameters(), lr=lr, betas=(0.5, 0.999))
D_optimizer = optim.Adam(D.parameters(), lr=lr, betas=(0.5, 0.999))
```

Define the “Adam” as a optimizer

Example 5.1 - Generator in the CGAN

Input Information	
Input Noise (z)	Dimension = 100
Input Label (y)	Dimension = 10
Output Information	
Output	Dimension = 32×32 (FashionMNIST)

The network structure of the generator in Example 5.1



Example 5.1 - CGAN

```
class generator(nn.Module):
    def __init__(self, d=128):
        super(generator, self).__init__()
        self.deconv1_1 = nn.ConvTranspose2d(100, d*2, 4, 1, 0)
        self.deconv1_1_bn = nn.BatchNorm2d(d*2)
        self.deconv1_2 = nn.ConvTranspose2d(10, d*2, 4, 1, 0)
        self.deconv1_2_bn = nn.BatchNorm2d(d*2)
        self.deconv2 = nn.ConvTranspose2d(d*4, d*2, 4, 2, 1)
        self.deconv2_bn = nn.BatchNorm2d(d*2)
        self.deconv3 = nn.ConvTranspose2d(d*2, d, 4, 2, 1)
        self.deconv3_bn = nn.BatchNorm2d(d)
        self.deconv4 = nn.ConvTranspose2d(d, 1, 4, 2, 1)
```

Define the architecture

```
    def weight_init(self, mean, std):
        for m in self._modules:
            normal_init(self._modules[m], mean, std)
```

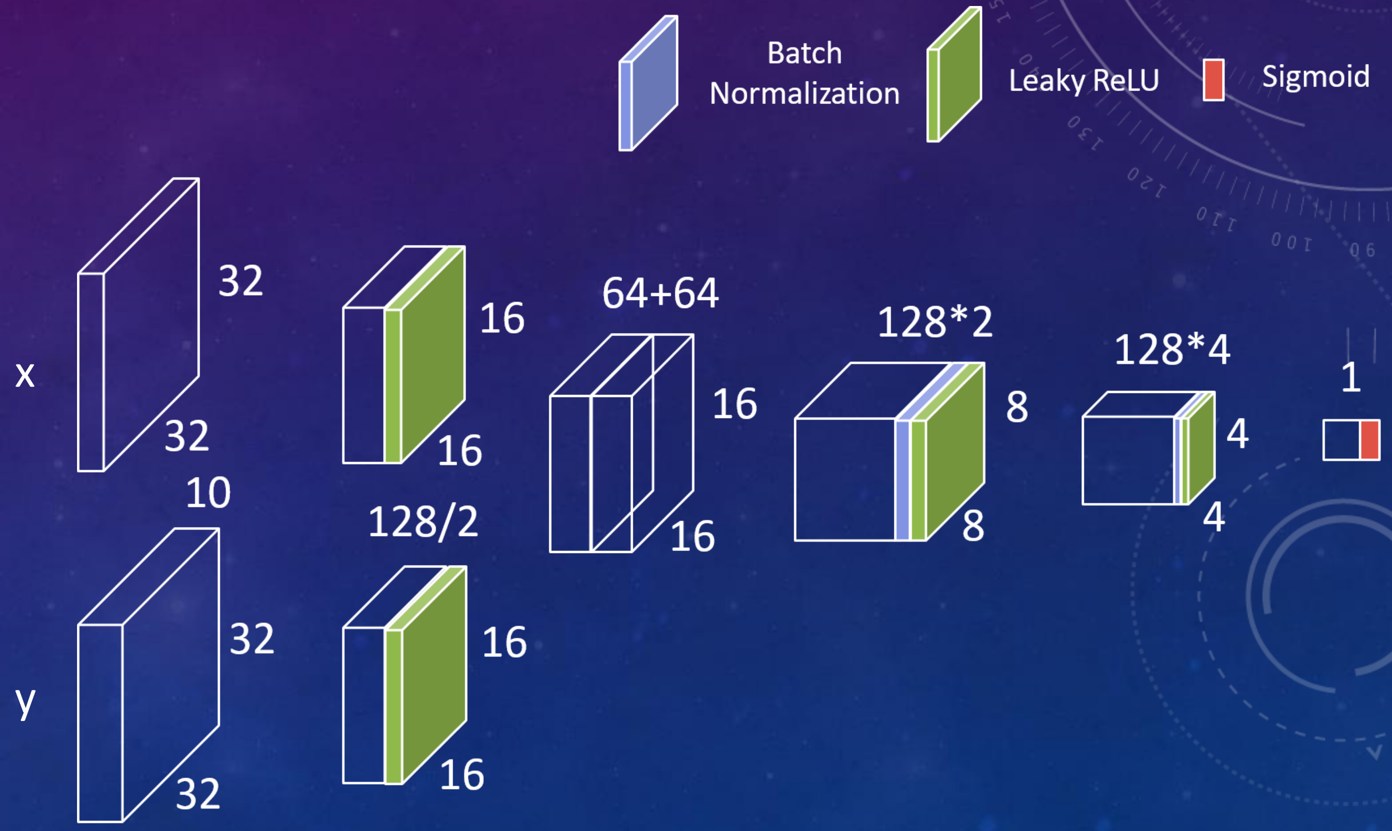
```
def forward(self, input, label):
    x  =  F.relu(self.deconv1_1_bn(self.deconv1_1(input)))
    y  =  F.relu(self.deconv1_2_bn(self.deconv1_2(label)))
    x = torch.cat([x, y], 1)
    x  =  F.relu(self.deconv2_bn(self.deconv2(x)))
    x  =  F.relu(self.deconv3_bn(self.deconv3(x)))
    x = F.tanh(self.deconv4(x))

    return x
```

Forward pass

Example 5.1 - Discriminator in the CGAN

Input Information	
Input Image (x)	Dimension = 32×32
Input Label (y)	Dimension = 32×32×10
Output Information	
Output	Dimension = 1

The network structure of the Discriminator in Example 5.1



Example 5.1 - CGAN

```
class discriminator(nn.Module):
    def __init__(self, d=128):
        super(discriminator, self).__init__()
        self.conv1_1 = nn.Conv2d(1, d//2, 4, 2, 1)
        self.conv1_2 = nn.Conv2d(10, d//2, 4, 2, 1)
        self.conv2 = nn.Conv2d(d, d*2, 4, 2, 1)
        self.conv2_bn = nn.BatchNorm2d(d*2)
        self.conv3 = nn.Conv2d(d*2, d*4, 4, 2, 1)
        self.conv3_bn = nn.BatchNorm2d(d*4)
        self.conv4 = nn.Conv2d(d * 4, 1, 4, 1, 0)
```

Define the architecture

```
    def weight_init(self, mean, std):
        for m in self._modules:
            normal_init(self._modules[m], mean, std)
```

```
def forward(self, input, label):
    x = F.leaky_relu(self.conv1_1(input), 0.2)
    y = F.leaky_relu(self.conv1_2(label), 0.2)
    x = torch.cat([x, y], 1)
    x = F.leaky_relu(self.conv2_bn(self.conv2(x)), 0.2)
    x = F.leaky_relu(self.conv3_bn(self.conv3(x)), 0.2)
    x = F.sigmoid(self.conv4(x))

    return x
```

Forward pass

Example 5.1 - CGAN

```
# fixed noise & label
fixed _z_2 = torch.randn(10, 100)
fixed_z_2 = fixed_z_2.view(-1, 100, 1, 1)
```

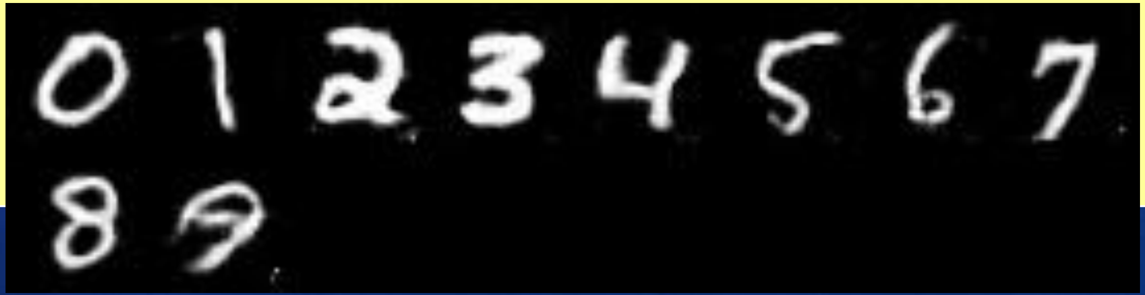
Define the fixed noise

```
one_hot = [[1,0,0,0,0,0,0,0,0,0],
           [0,1,0,0,0,0,0,0,0,0],
           [0,0,1,0,0,0,0,0,0,0],
           [0,0,0,1,0,0,0,0,0,0],
           [0,0,0,0,1,0,0,0,0,0],
           [0,0,0,0,0,1,0,0,0,0],
           [0,0,0,0,0,0,1,0,0,0],
           [0,0,0,0,0,0,0,1,0,0],
           [0,0,0,0,0,0,0,0,1,0],
           [0,0,0,0,0,0,0,0,0,1]]
```

Define the label from 0 to 9

```
one_hot_arr = np.array(one_hot)
fixed_y_label_2 = torch.from_numpy(one_hot_arr)

fixed_y_label_2 = fixed_y_label_2.view(-1, 10, 1, 1)

fixed_y_label_2 = fixed_y_label_2.float()
fixed_z_2 = fixed_z_2.float()

fixed_z_2, fixed_y_label_2 = Variable(fixed_z_2.cuda()),
Variable(fixed_y_label_2.cuda())
```

Result :



Exercise 5.1 - CGAN

- Please download the “exercise_5.1_CGAN.ipynb” from Moodle.
- Upload the Notebook to GoogleColab.
- Train the CGAN and compare the images reconstructed from different numbers of epochs.
- Define the 5 different one-hot vector to generate the images and compare the difference.
- Please copy your results and code and paste to a MS Word, then upload to Moodle.

Example :

[1,0,0,0,0,0,0,0,0,0] ⇨ [0.5,0.5,0,0,0,0,0,0,0,0]

Example 5.2

[1] For this exercise we are using the github from the authors which is available from the link https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

To clone it into your colab drive we use git clone followed by the github link

```
!git clone https://github.co
```

```
[2] import os
    os.chdir('pytorch-CycleGAN-and-pix2pix/')
```

```
[3] !pip install -r requirements.txt
```

[2] Os.chdir changes the cwd

[3] we install all packages necessary for the github which are saved in the requirements.txt file

Datasets

Download one of the official datasets with:

- `bash ./datasets/download_pix2pix_datas`

Or use your own dataset by creating the appropriate folders and adding in the images. Follow the instructions here.

```
[4] !bash ./datasets/download_pix2pix_dataset.sh facades
```

Pretrained models

Download one of the official pretrained models with:

- `bash ./scripts/download_pix2pix_model.sh [edges2shoes, sat2map, map2sat, facades_label2photo, and day2night]`

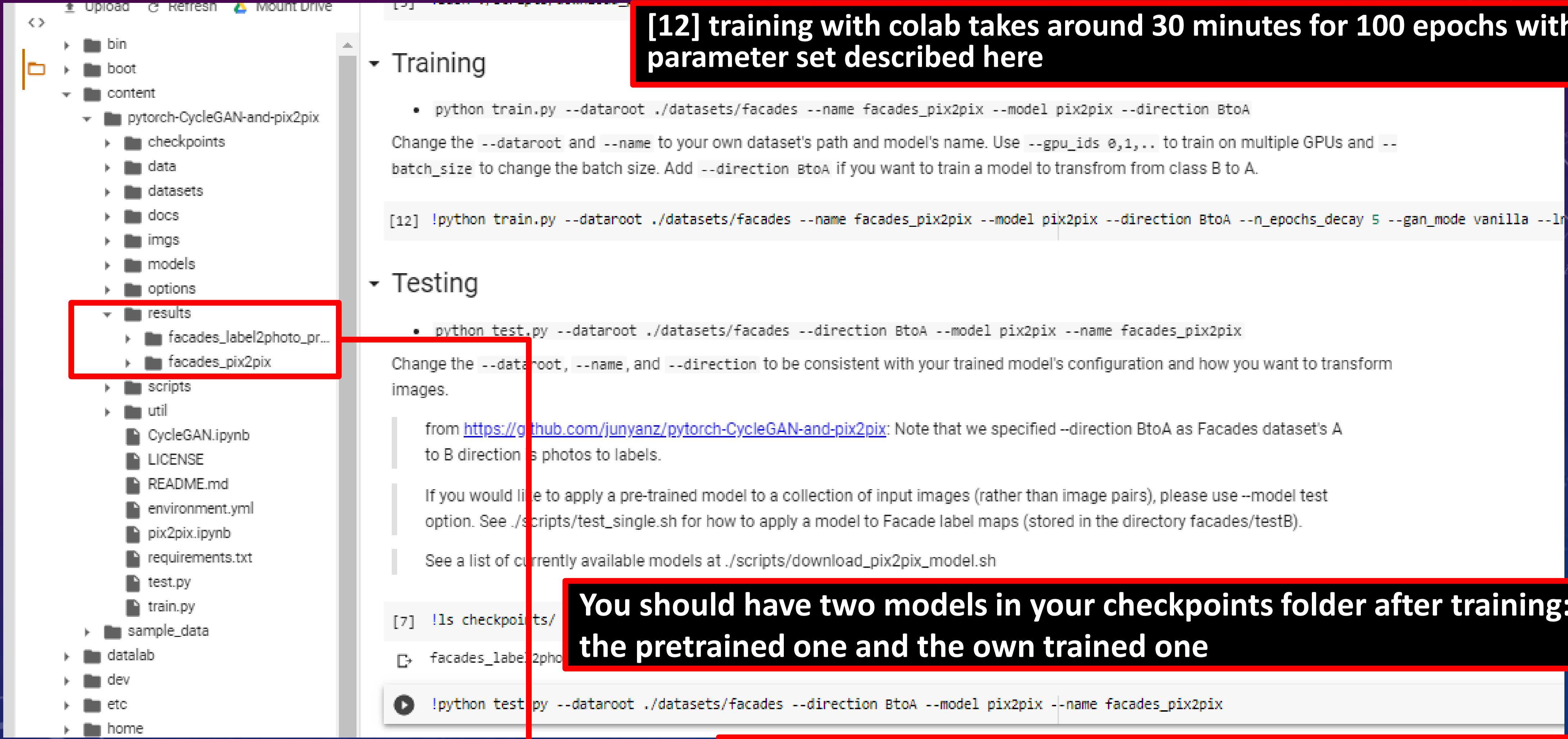
Or add your own pretrained model to `./checkpoints/{NAME}_pretrained/latest_net_G.pt`

```
!bash ./scripts/download_pix2pix_model.sh facades_label2photo
```

[4,5] Bash command executes the commands in .sh files

Here we download the dataset and the pretrained weights

Example 5.2



[12] training with colab takes around 30 minutes for 100 epochs with the parameter set described here

You should have two models in your checkpoints folder after training: the pretrained one and the own trained one

The test.py will output the translated images to the results folder. You can change the weights with the –name argument

Example 5.2 – Training overview

You can train the model by the following commands：

From domain B to domain A

```
!python train.py --dataroot ./datasets/facades --name facades_pix2pix --model pix2pix --direction BtoA --n_epochs_decay 5 --continue_train --gan_mode vanilla --lr_policy step
```

(train.py) You can change the loss function in the following command：

```
parser.add_argument('--gan_mode', type=str, default='lsgan', help='the type of GAN objective. [vanilla| lsgan | wgangp]
```

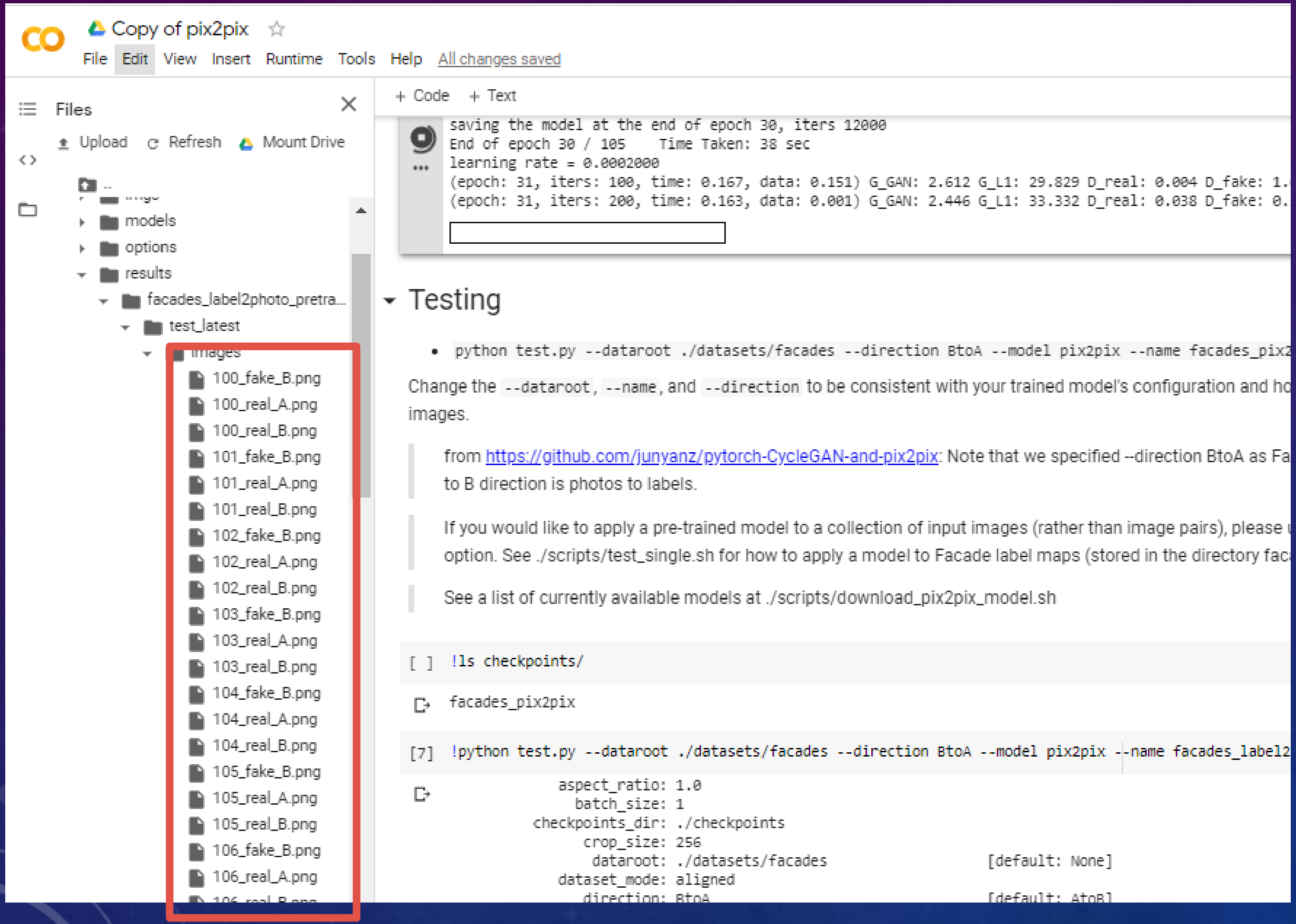
Overview of training

```
----------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: True                          	[default: False]
                crop_size: 256
                 dataroot: ./datasets/facades            	[default: None]
             dataset_mode: aligned
                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: vanilla
                  gpu_ids: 0
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: True                          	[default: None]
                lambda_L1: 100.0
```

Observing the training

```
Setting up a new session...
create web directory ./checkpoints/facades_pix2pix/web...
(epoch: 1, iters: 100, time: 0.165, data: 0.293) G_GAN: 2.553 G_L1: 47.217 D_real: 0.009 D_fake: 0.177
(epoch: 1, iters: 200, time: 0.165, data: 0.001) G_GAN: 3.394 G_L1: 40.468 D_real: 0.030 D_fake: 0.135
(epoch: 1, iters: 300, time: 0.165, data: 0.002) G_GAN: 3.118 G_L1: 40.225 D_real: 0.003 D_fake: 1.685
(epoch: 1, iters: 400, time: 0.339, data: 0.001) G_GAN: 1.706 G_L1: 35.663 D_real: 0.118 D_fake: 0.926
End of epoch 1 / 105 	 Time Taken: 39 sec
learning rate = 0.0002000
(epoch: 2, iters: 100, time: 0.163, data: 0.108) G_GAN: 2.846 G_L1: 30.274 D_real: 0.021 D_fake: 0.664
(epoch: 2, iters: 200, time: 0.165, data: 0.002) G_GAN: 1.453 G_L1: 27.789 D_real: 1.481 D_fake: 0.160
(epoch: 2, iters: 300, time: 0.166, data: 0.001) G_GAN: 2.581 G_L1: 39.521 D_real: 0.017 D_fake: 0.320
(epoch: 2, iters: 400, time: 0.312, data: 0.001) G_GAN: 2.497 G_L1: 27.504 D_real: 0.568 D_fake: 0.039
End of epoch 2 / 105 	 Time Taken: 37 sec
learning rate = 0.0002000
```

Example 5.2 - Results



The red textbox shows the folder with all your results. You can later have a look by adjusting the names of the files in the next slide, downloading or just clicking them one by one.

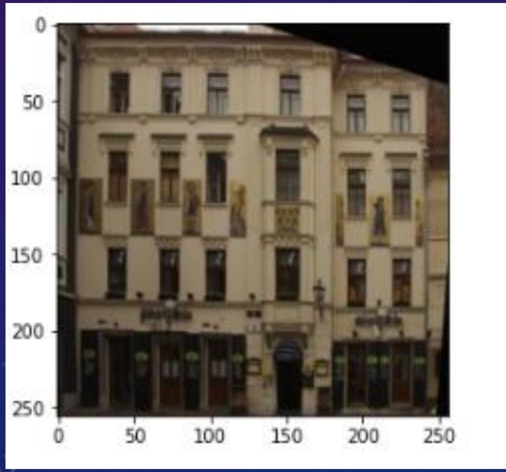
Example 5.2 - Results

The command below will show the images. Remember you can change the output of the imageplot by changing the file name. left is the Fake image on the left side on Domain A,

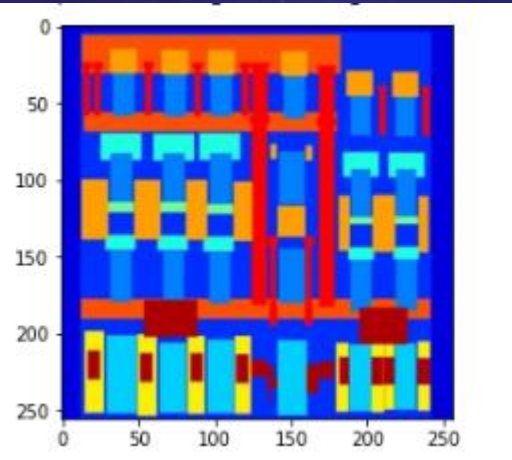
```
img = plt.imread('./results/facades_label2photo_pretrained/test_latest/images/100_fake_B.png')
plt.imshow(img)
```

```
img = plt.imread('./results/facades_label2photo_pretrained/test_latest/images/104_fake_B.png')
plt.imshow(img)
```

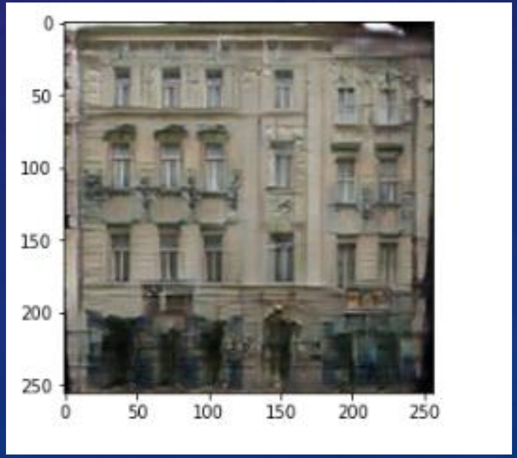
Real Image (Training Set)



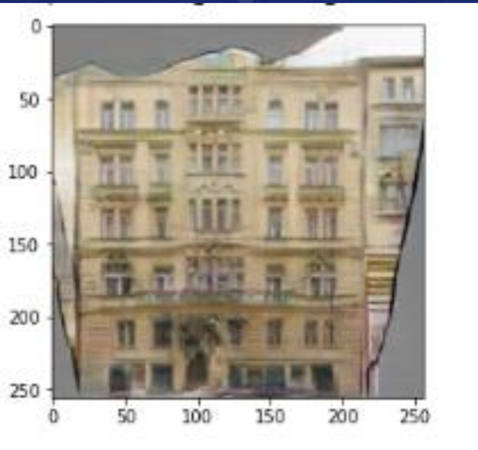
Labels of real image (Domain B)



Fake Image from Labels of Domain B (Domain A)



Another fake image



Exercise 5.2 - Pix2Pix

- Download “Pix2Pix_Exercise_5_2.ipynb” from Moodle and upload it to colab
- For training: adjust the dataset you are using, the gan_mode, the lr_policy and the number of epochs
- Use the same pre-trained model as you are training to make a comparison in the end
- Compare the results of the pre-trained and your own trained model
- Lastly, change the direction of your training from B to A to A to B
- Note down your observations and upload both your notebook and observations to Moodle.