HW2

0856703 黃威竣

1. Github link

This is my homework2 for github: https://github.com/f51980280/DLCV_hw2

2. Introduction

I used keras to build my DCGAN model

And I used helper.py for get_batches, grid_images, get_images….. some useful function. This function make me convenience to used celeba image set for scaling and output images.

Below is my code:

I use 64 x 64 pixel for training

```
import os
import helper
import matplotlib.pyplot as plt
from glob import glob
import numpy as np

data_dir = './data'
helper.download_extract(data_dir)
celeba_dataset = helper.Dataset('celeba', glob(os.path.join(data_dir, 'img_align_celeba/*.jpg')))
celeba_dataset.shape
```

```
Found celeba Data
```

```
(202599, 64, 64, 3)
```

Class init

```
from __future__ import print_function, division

from keras.datasets import mnist
from keras.layers import Input, Dense, Reshape, Flatten, Dropout
from keras.layers import BatchNormalization, Activation, ZeroPadding2D
from keras.layers.advanced_activations import LeakyReLU
from keras.layers.convolutional import UpSampling2D, Conv2D ,Conv2DTranspose
from keras.models import Sequential, Model
from keras.optimizers import Adam
import keras.backend as K

import matplotlib.pyplot as plt

import sys

import numpy as np

class DCGAN():

    def __init__(self):
        self.img_rows = 64
        self.img_cols = 64
        self.channels = 3
        self.img_shape = (self.img_rows, self.img_cols, self.channels)
        self.latent_dim = 100

        optimizer = Adam(0.0002, 0.5)

        self.discriminator = self.build_discriminator()
        self.discriminator.compile(loss='binary_crossentropy',
            optimizer=optimizer,
            metrics=['accuracy'])


        self.generator = self.build_generator()

        z = Input(shape=(self.latent_dim,))
        img = self.generator(z)

        self.discriminator.trainable = False

        valid = self.discriminator(img)

        self.combined = Model(z, valid)
        self.combined.compile(loss='binary_crossentropy', optimizer=optimizer)
```

Generator

```
def build_generator(self):

    model = Sequential()

    model.add(Dense(1024 * 4 * 4, activation="relu", input_dim=self.latent_dim))
    model.add(Reshape((4, 4, 1024)))


    model.add(Conv2DTranspose(512, kernel_size=5, strides=2, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Activation("relu"))
    model.add(Dropout(0.25))


    model.add(Conv2DTranspose(256, kernel_size=5, strides=2, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Activation("relu"))
    model.add(Dropout(0.25))

    model.add(Conv2DTranspose(128, kernel_size=5, strides=2, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Activation("relu"))
    model.add(Dropout(0.25))

    model.add(Conv2DTranspose(self.channels, kernel_size=4, strides=2, padding="same"))
    model.add(Activation("tanh"))

    model.summary()

    noise = Input(shape=(self.latent_dim,))
    img = model(noise)

    return Model(noise, img)
```

Discriminator

```
def build_discriminator(self):

    model = Sequential()


    model.add(Conv2D(128, kernel_size=5, strides=2, input_shape=self.img_shape, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(LeakyReLU(alpha=0.02))
    model.add(Dropout(0.25))

    model.add(Conv2D(256, kernel_size=5, strides=2, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(LeakyReLU(alpha=0.02))
    model.add(Dropout(0.25))

    model.add(Conv2D(512, kernel_size=5, strides=2, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(LeakyReLU(alpha=0.02))
    model.add(Dropout(0.25))

    model.add(Conv2D(1024, kernel_size=5, strides=2, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(LeakyReLU(alpha=0.02))
    model.add(Dropout(0.25))


    model.add(Flatten())
    model.add(Dense(1, activation='sigmoid'))

    model.summary()

    img = Input(shape=self.img_shape)
    validity = model(img)

    return Model(img, validity)
```

For train

```
def train(self, epochs, batch_size):

    steps = 0
    noise_fix = np.random.normal(0, 1, (9, self.latent_dim))

    valid = np.ones((batch_size, 1))
    fake = np.zeros((batch_size, 1))

    for epoch in range(epochs):
        for batch_images in celeba_dataset.get_batches(batch_size):


            batch_images*=2


            noise = np.random.normal(0, 1, (batch_size, self.latent_dim))
            gen_imgs = self.generator.predict(noise)
            out_noise = np.random.normal(0, 1, (9, self.latent_dim))
            out_imgs = self.generator.predict(out_noise)

            d_loss_real = self.discriminator.train_on_batch(batch_images, valid)
            d_loss_fake = self.discriminator.train_on_batch(gen_imgs, fake)
            d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

            if steps % 100 == 0:
                    fixed_img = self.generator.predict(noise_fix)
                    plt.imshow(helper.images_square_grid(out_imgs))
                    plt.show()
                    plt.imshow(helper.images_square_grid(fixed_img))
                    plt.show()

            g_loss = self.combined.train_on_batch(noise, valid)

            print ("%d [D loss: %f, acc.: %.2f%%] [G loss: %f]" % (steps, d_loss[0], 100*d_loss[1], g_loss))
            steps+=1
            print("epoch = " ,epoch)

        epoch+=1
```

Use for GPU

```
from distutils.version import LooseVersion
import warnings
import tensorflow as tf

# Check TensorFlow Version
assert LooseVersion(tf.__version__) >= LooseVersion('1.0'), 'Please use TensorFlow version 1.0 or newer.  You are using {}'.form
print('TensorFlow Version: {}'.format(tf.__version__))

# Check for a GPU
if not tf.test.gpu_device_name():
    warnings.warn('No GPU found. Please use a GPU to train your neural network.')
else:
    print('Default GPU Device: {}'.format(tf.test.gpu_device_name()))


gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.8)

sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))

import keras as K
K.backend.clear_session()
```

Main function

```
if __name__ == '__main__':
    dcgan = DCGAN()
    dcgan.train(epochs=30, batch_size=128)
```

Save for image png

```
for i in range(500):
    noise = np.random.normal(0, 1, (9, 100))
    plt.imshow(helper.images_square_grid(bgan.generator.predict(noise)))
    plt.axis("off")
    plt.savefig("./data/face_images_2/ %d_image.png" % i,bbox_inches='tight')
    plt.show()
```

3. Summary and Finding

I build DCGAN with Adam optimizer (learning rate = 0.0002, beta1 = 0.5)
I tried use some learning rate like 0.0001 or 0.00015…. and beta1 tried to use 0.9、0.8、0.7. but when training the model, the images looks no good, and I also tried to add some extra layer, but model is easy to unbalance, like discriminator become strong than generator. Obvious, the control of parameter or some model topology is very important, so I think the GAN model is difficult to build. At last, I only modified some DCGAN code.
I modified the Conv2D with Unpooling to Conv2DTranspose in generator, and also add Dropout(0.25) to discriminator and generator, I tried to add more conv layer or reduce conv layer, but result is seems bad. So finally my model is

Discriminator model:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 128)	9728
batch_normalization_1 (Batch	(None, 32, 32, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 32, 32, 128)	0
dropout_1 (Dropout)	(None, 32, 32, 128)	0
conv2d_2 (Conv2D)	(None, 16, 16, 256)	819456
batch_normalization_2 (Batch	(None, 16, 16, 256)	1024
leaky_re_lu_2 (LeakyReLU)	(None, 16, 16, 256)	0
dropout_2 (Dropout)	(None, 16, 16, 256)	0
conv2d_3 (Conv2D)	(None, 8, 8, 512)	3277312
batch_normalization_3 (Batch	(None, 8, 8, 512)	2048
leaky_re_lu_3 (LeakyReLU)	(None, 8, 8, 512)	0
dropout_3 (Dropout)	(None, 8, 8, 512)	0
conv2d_4 (Conv2D)	(None, 4, 4, 1024)	13108224
batch_normalization_4 (Batch	(None, 4, 4, 1024)	4096
leaky_re_lu_4 (LeakyReLU)	(None, 4, 4, 1024)	0
dropout_4 (Dropout)	(None, 4, 4, 1024)	0

```
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 16385
=================================================================
Total params: 17,238,785
Trainable params: 17,234,945
Non-trainable params: 3,840
```

Generator model:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 16384)             1654784
_________________________________________________________________
reshape_1 (Reshape)          (None, 4, 4, 1024)        0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 8, 8, 512)         13107712
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 512)         2048
_________________________________________________________________
activation_1 (Activation)    (None, 8, 8, 512)         0
_________________________________________________________________
dropout_5 (Dropout)          (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 16, 16, 256)       3277056

_________________________________________________________________
batch_normalization_6 (Batch (None, 16, 16, 256)       1024
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 256)       0
_________________________________________________________________
dropout_6 (Dropout)          (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 32, 32, 128)       819328
_________________________________________________________________
batch_normalization_7 (Batch (None, 32, 32, 128)       512
_________________________________________________________________
activation_3 (Activation)    (None, 32, 32, 128)       0
_________________________________________________________________
dropout_7 (Dropout)          (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 64, 64, 3)         6147

_________________________________________________________________
activation_4 (Activation)    (None, 64, 64, 3)         0
=================================================================
Total params: 18,868,611
Trainable params: 18,866,819
Non-trainable params: 1,792
_________________________________________________________________
```