**导入各种模块**

import os

import scipy.misc

import numpy as np

import json

from model import DCGAN

from utils import pp, visualize, show\_all\_variables, expand\_path, timestamp

import tensorflow as tf

**这里需要注意的是 flags = tf.app.flags 用于tensorflow构建命令行参数, flags.DEFINE\_xxx(param,default,description) 用于定义命令行参数及其取值,第一个参数param是具体参数值,第二个参数default是参数默认取值,第三个参数description是参数描述字符串。**

**执行main函数之前首先进行flags的解析，TensorFlow底层使用了python-gflags项目，然后封装成tf.app.flags接口，也就是说TensorFlow通过设置flags来传递tf.app.run()所需要的参数，我们可以直接在程序运行前初始化flags，也可以在运行程序的时候设置命令行参数来达到传参的目的。**

flags = tf.app.flags

flags.DEFINE\_integer("epoch", 25, "Epoch to train [25]")

flags.DEFINE\_float("learning\_rate", 0.0002, "Learning rate of for adam [0.0002]")

flags.DEFINE\_float("beta1", 0.5, "Momentum term of adam [0.5]")

flags.DEFINE\_float("train\_size", np.inf, "The size of train images [np.inf]")

flags.DEFINE\_integer("batch\_size", 64, "The size of batch images [64]")

flags.DEFINE\_integer("input\_height", 108, "The size of image to use (will be center cropped). [108]")

flags.DEFINE\_integer("input\_width", None, "The size of image to use (will be center cropped). If None, same value as input\_height [None]")

flags.DEFINE\_integer("output\_height", 64, "The size of the output images to produce [64]")

flags.DEFINE\_integer("output\_width", None, "The size of the output images to produce. If None, same value as output\_height [None]")

flags.DEFINE\_string("dataset", "celebA", "The name of dataset [celebA, mnist, lsun]")

flags.DEFINE\_string("input\_fname\_pattern", "\*.jpg", "Glob pattern of filename of input images [\*]")

flags.DEFINE\_string("data\_dir", "./data", "path to datasets [e.g. $HOME/data]")

flags.DEFINE\_string("out\_dir", "./out", "Root directory for outputs [e.g. $HOME/out]")

flags.DEFINE\_string("out\_name", "", "Folder (under out\_root\_dir) for all outputs. Generated automatically if left blank []")

flags.DEFINE\_string("checkpoint\_dir", "checkpoint", "Folder (under out\_root\_dir/out\_name) to save checkpoints [checkpoint]")

flags.DEFINE\_string("sample\_dir", "samples", "Folder (under out\_root\_dir/out\_name) to save samples [samples]")

flags.DEFINE\_boolean("train", False, "True for training, False for testing [False]")

flags.DEFINE\_boolean("crop", False, "True for training, False for testing [False]")

flags.DEFINE\_boolean("visualize", True, "True for visualizing, False for nothing [False]")

flags.DEFINE\_boolean("export", False, "True for exporting with new batch size")

flags.DEFINE\_boolean("freeze", False, "True for exporting with new batch size")

flags.DEFINE\_integer("max\_to\_keep", 1, "maximum number of checkpoints to keep")

flags.DEFINE\_integer("sample\_freq", 50, "sample every this many iterations")#simple图像导出

flags.DEFINE\_integer("ckpt\_freq", 100, "save checkpoint every this many iterations")

flags.DEFINE\_integer("z\_dim", 100, "dimensions of z")

flags.DEFINE\_string("z\_dist", "uniform\_signed", "'normal01' or 'uniform\_unsigned' or uniform\_signed")

flags.DEFINE\_boolean("G\_img\_sum", True, "Save generator image summaries in log")

#flags.DEFINE\_integer("generate\_test\_images", 100, "Number of images to generate during test. [100]")

FLAGS = flags.FLAGS

**首先是打印参数数据，然后判断输入图像的输出图像的宽是否指定，如果没有指定，则等于其图像的高。**

def main(\_):

pp.pprint(flags.FLAGS.\_\_flags)

**# 展开用户名和环境变量**

FLAGS.data\_dir = expand\_path(FLAGS.data\_dir)

FLAGS.out\_dir = expand\_path(FLAGS.out\_dir)

FLAGS.out\_name = expand\_path(FLAGS.out\_name)

FLAGS.checkpoint\_dir = expand\_path(FLAGS.checkpoint\_dir)

FLAGS.sample\_dir = expand\_path(FLAGS.sample\_dir)

if FLAGS.output\_height is None: FLAGS.output\_height = FLAGS.input\_height

if FLAGS.input\_width is None: FLAGS.input\_width = FLAGS.input\_height

if FLAGS.output\_width is None: FLAGS.output\_width = FLAGS.output\_height

**然后判断checkpoint和sample的文件是否存在，不存在则创建。**

if FLAGS.out\_name == "":

FLAGS.out\_name = '{} - {} - {}'.format(timestamp(), FLAGS.data\_dir.split('/')[-1], FLAGS.dataset) # penultimate folder of path

if FLAGS.train:

FLAGS.out\_name += ' - x{}.z{}.{}.y{}.b{}'.format(FLAGS.input\_width, FLAGS.z\_dim, FLAGS.z\_dist, FLAGS.output\_width, FLAGS.batch\_size)

FLAGS.out\_dir = os.path.join(FLAGS.out\_dir, FLAGS.out\_name)

FLAGS.checkpoint\_dir = os.path.join(FLAGS.out\_dir, FLAGS.checkpoint\_dir)

FLAGS.sample\_dir = os.path.join(FLAGS.out\_dir, FLAGS.sample\_dir)

if not os.path.exists(FLAGS.checkpoint\_dir): os.makedirs(FLAGS.checkpoint\_dir)

if not os.path.exists(FLAGS.sample\_dir): os.makedirs(FLAGS.sample\_dir)

**GPU训练选项使用allow\_growth option，刚一开始分配少量的GPU容量，然后按需慢慢的增加，由于不会释放内存，所以会导致碎片**

**【碎片化可选选项】**

#gpu\_options = tf.GPUOptions(per\_process\_gpu\_memory\_fraction=0.333)

run\_config = tf.compat.v1.ConfigProto()

run\_config.gpu\_options.allow\_growth=True

**然后是设置session参数。tf.ConfigProto一般用在创建session的时候，用来对session进行参数配置【session可选选项】**

with tf.compat.v1.Session(config=run\_config) as sess:

dcgan = DCGAN(

sess,

input\_width=FLAGS.input\_width,

input\_height=FLAGS.input\_height,

output\_width=FLAGS.output\_width,

output\_height=FLAGS.output\_height,

batch\_size=FLAGS.batch\_size,

sample\_num=FLAGS.batch\_size,

z\_dim=FLAGS.z\_dim,

dataset\_name=FLAGS.dataset,

input\_fname\_pattern=FLAGS.input\_fname\_pattern,

crop=FLAGS.crop,

checkpoint\_dir=FLAGS.checkpoint\_dir,

sample\_dir=FLAGS.sample\_dir,

data\_dir=FLAGS.data\_dir,

out\_dir=FLAGS.out\_dir,

max\_to\_keep=FLAGS.max\_to\_keep)

**show所有与训练相关的变量**

show\_all\_variables()

**接下来如果是训练状态( FLAGS.train == True ),则进行模型训练( dcgan.train(FLAGS) ;否则进行测试,即加载之前训练时候保存的checkpoint文件，然后调用 visualize 函数进行test(该函数可以生成image,可视化展示训练的效果)。**

**判断是训练还是测试，如果是训练，则进行训练；如果不是，判断是否有训练好的model，然后进行测试，如果没有先训练，则会提示“[!] Train a model first, then run test mode”**

if FLAGS.train:

dcgan.train(FLAGS)

else:

load\_success, load\_counter = dcgan.load(FLAGS.checkpoint\_dir)

if not load\_success:

raise Exception("Checkpoint not found in " + FLAGS.checkpoint\_dir)

**# 检查点保存**

if FLAGS.export:

export\_dir = os.path.join(FLAGS.checkpoint\_dir, 'export\_b'+str(FLAGS.batch\_size))

dcgan.save(export\_dir, load\_counter, ckpt=True, frozen=False)

if FLAGS.freeze:

export\_dir = os.path.join(FLAGS.checkpoint\_dir, 'frozen\_b'+str(FLAGS.batch\_size))

dcgan.save(export\_dir, load\_counter, ckpt=False, frozen=True)

**最后进行可视化，**

visualize(sess, dcgan, FLAGS, OPTION)

if FLAGS.visualize:

OPTION = 1

visualize(sess, dcgan, FLAGS, OPTION, FLAGS.sample\_dir)

if \_\_name\_\_ == '\_\_main\_\_':

tf.compat.v1.app.run()