"""**PPT里写上引用**

Some codes from https://github.com/Newmu/dcgan\_code

"""

**这份代码主要是定义了各种对图像处理的函数，相当于其他3个文件的头文件。**

**glob库用来list 某一个文件夹下的files;os库用来操作路径和文件夹等;pprint用于美观打印;gtime和strftime有用格式化日期;scipy.misc包含了很多和图像相关的有用的函数。**

from \_\_future\_\_ import division

import math

import json

import random

import pprint

import scipy.misc

import cv2

import numpy as np

import os

import time

import datetime

from time import gmtime, strftime

from six.moves import xrange

from PIL import Image

import PIL

import imageio

import tensorflow as tf

import tensorflow.contrib.slim as slim

**首先定义了一个pp = pprint.PrettyPrinter()，以方便打印数据结构信息**

**详细：**<https://blog.csdn.net/liangyuannao/article/details/8709190>

pp = pprint.PrettyPrinter()

get\_stddev = lambda x, k\_h, k\_w: 1/math.sqrt(k\_w\*k\_h\*x.get\_shape()[-1])

**定义了get\_stddev函数，是三个参数乘积后开平方的倒数，应该是为了随机化用**

def expand\_path(path):

return os.path.expanduser(os.path.expandvars(path))

def timestamp(s='%Y%m%d.%H%M%S', ts=None):

if not ts: ts = time.time()

st = datetime.datetime.fromtimestamp(ts).strftime(s)

return st

**定义show\_all\_variables()函数。首先，tf.trainable\_variables返回的是需要训练的变量列表；然后用tensorflow.contrib.slim中的model\_analyzer.analyze\_vars打印出所有与训练相关的变量信息。调用了 slim.model\_analyzer.analyze\_vars(vars,print\_info) 函数来打印model所有variables的信息**

def show\_all\_variables():

model\_vars = tf.compat.v1.trainable\_variables()

slim.model\_analyzer.analyze\_vars(model\_vars, print\_info=True)

**定义get\_image(image\_path,input\_height,input\_width,resize\_height=64, resize\_width=64,crop=True, grayscale=False)函数。首先根据图像路径参数读取路径，根据灰度化参数选择是否进行灰度化。然后对图像参照输入的参数进行裁剪**

def get\_image(image\_path, input\_height, input\_width,

resize\_height=64, resize\_width=64,

crop=True, grayscale=False):

image = imread(image\_path, grayscale)

return transform(image, input\_height, input\_width,

resize\_height, resize\_width, crop)

**定义save\_images(images,size,image\_path)函数。调用imsave(inverse\_transform(images), size, image\_path)函数并返回新图像**

def save\_images(images, size, image\_path):

return imsave(inverse\_transform(images), size, image\_path)

**定义imread(path, grayscale = False)函数。调用cipy.misc.imread()函数，判断grayscale参数是否进行范围灰度化，并进行类型转换为np.float**

def imread(path, grayscale = False):

if (grayscale):

return scipy.misc.imread(path, flatten = True).astype(np.float)

else:

**# Reference: @ DNXie**

img\_bgr = cv2.imread(path)

**# Reference: https://stackoverflow.com/a/15074748/**

img\_rgb = img\_bgr[..., ::-1]

return img\_rgb.astype(np.float)

**定义merge\_images(images, size)函数。调用inverse\_transform(images)函数，并返回新图像**

def merge\_images(images, size):

return inverse\_transform(images)

**定义merge(images, size)函数。首先获取image的高和宽。然后判断image是RGB图还是灰度图，以分别进行不同的处理。如果通道数是3或4，则对每一批次（如，batch\_size=64）的所有图像，用0初始化一张原始图像放大8\*8的图像，然后循环，依次将所有图像填入大图像，并且返回这张大图像。如果通道数是1，也是一样，只不过填入图像的时候只填一个通道的信息。如果不是上述两种情况，则抛出错误提示**

**merge函数用于从一系列小图产生大图,images[0]表示小图的个数,h=images[1]表示小图的高,w = images[2]表示小图的宽，x\_h = size[0]表示最终大图height应该扩展的倍数,x\_w = size[1]表示最终大图width应该扩展的倍数。该函数最终生成一个高为h\*x\_h，宽为w\*x\_w的大图。表示大图的高度方向包含x\_h个小图,宽度方向包含x\_w个小图**

def merge(images, size):

h, w = images.shape[1], images.shape[2]

if (images.shape[3] in (3,4)):

c = images.shape[3]

img = np.zeros((h \* size[0], w \* size[1], c))

for idx, image in enumerate(images):

i = idx % size[1]

j = idx // size[1]

img[j \* h:j \* h + h, i \* w:i \* w + w, :] = image

return img

elif images.shape[3]==1:

img = np.zeros((h \* size[0], w \* size[1]))

for idx, image in enumerate(images):

i = idx % size[1]

j = idx // size[1]

img[j \* h:j \* h + h, i \* w:i \* w + w] = image[:,:,0]

return img

else:

raise ValueError('in merge(images,size) images parameter '

'must have dimensions: HxW or HxWx3 or HxWx4')

**定义imsave(images, size, path)函数。首先将merge()函数返回的图像，用 np.squeeze()函数移除长度为1的轴。然后利用scipy.misc.imsave()函数将新图像保存到指定路径中**

**注意 np.squeeze 可以去除数组中维度为1的那些维(降维),与之相反的操作是 np.expand\_dims(arr,axis) 函数，可以给指定的axis维度增加一维。**

def imsave(images, size, path):

image = np.squeeze(merge(images, size))

return imageio.imwrite(path,image)

**定义center\_crop(x, crop\_h, crop\_w,resize\_h=64, resize\_w=64)函数。对图像的H和W与crop的H和W相减，得到取整的值，根据这个值作为下标依据来scipy.misc.resize图像**

**中心化剪切处理,同时对图像进行了resize操作**

def center\_crop(x, crop\_h, crop\_w,

resize\_h=64, resize\_w=64):

if crop\_w is None:

crop\_w = crop\_h

h, w = x.shape[:2]

j = int(round((h - crop\_h)/2.))

i = int(round((w - crop\_w)/2.))

im = Image.fromarray(np.uint8(x[j:j + crop\_h, i:i + crop\_w]))

return np.array(im.resize([resize\_h, resize\_w], PIL.Image.BILINEAR))

#引用部分

**定义transform(image, input\_height, input\_width,resize\_height=64, resize\_width=64, crop=True)函数。对输入的图像进行裁剪，如果crop为true，则使用center\_crop()函数，对图像的H和W与crop的H和W相减，得到取整的值，根据这个值作为下标依据来scipy.misc.resize图像；否则不对图像进行其他操作，直接scipy.misc.resize为64\*64大小的图像。最后返回图像**

**对图像进行center\_crop(可选)以及resize操作，只不过它最后将image array的每个元素的取值范围从(0,255)映射到(-1,1),(-1,1)是tanh函数的取值范围**

def transform(image, input\_height, input\_width,

resize\_height=64, resize\_width=64, crop=True):

if crop:

cropped\_image = center\_crop(

image, input\_height, input\_width,

resize\_height, resize\_width)

h, w = image.shape[:2]

j = int(round((h - input\_height) / 2.))

i = int(round((w - input\_width) / 2.))

im = Image.fromarray(np.uint8(image[j:j + input\_height, i:i + input\_width]))

else:

im = Image.fromarray(image[j:j+crop\_h, i:i+crop\_w])

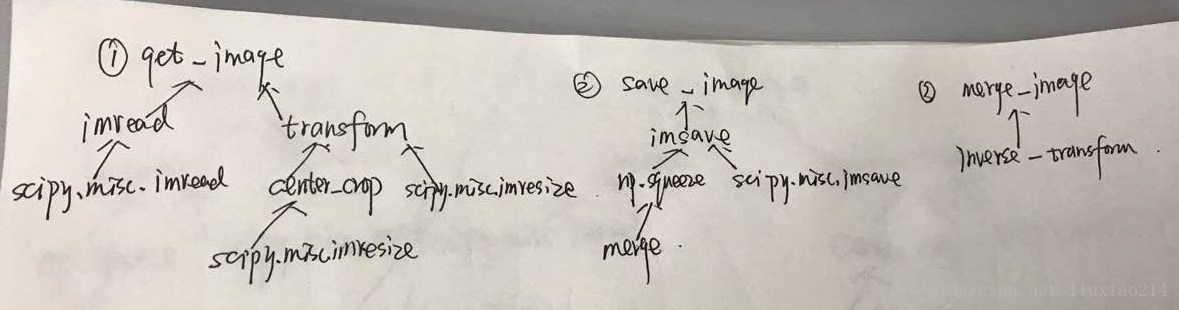
return np.array(im.resize([resize\_height, resize\_width]), np.uint8(Image.BILINEAR)) / 127.5 - 1.

**定义inverse\_transform(images)函数。对图像进行翻转后返回新图像**

def inverse\_transform(images):

return (images+1.)/2.

总结下来，这几个函数相互调用，主要实现了3个图像操作功能：获取图像get\_image()，负责读取图像，返回图像裁剪后的新图像；保存图像save\_images()，负责将一个batch中所有图像保存为一张大图像并返回；图像翻转merge\_images()，负责不知道怎么得翻转的，返回新图像。它们之间的相互关系如下图所示。



把这个图片用电脑整个放到ppt上

**定义visualize(sess, dcgan, config, option)函数。分为0、1、2、3、4种option。如果option=0，则之间显示生产的样本‘如果option=1，根据不同数据集不一样的处理，并利用前面的save\_images()函数将sample保存下来；等等。本次在main.py中选用option=1**

**用于测试阶段生成图像样本，可以是单个jpg格式的图像，也可以是gif图像，还可以是小图拼接成的大图。visualize函数通过option变量的取值(可以取0,1,2,3,4五个值)来控制以五种不同的方式保存结果。**

**option=0:这种情况只适用于dataset 不等于mnist的情况，直接将samples merge成一个大图，然后保存即可，其中大图共有batch\_size张小图，每行和每列各有ceil(sqrt(batch\_size))个;**

**option=1:这种情况和option=0类似，只是它考虑到了dataset为mnist的情况，如果是mnist,则会随机生成batch\_size个digit labels,然后从generator生成相应的数字，最后拼接成一个大图,这里我自己定义了一个save\_random\_digits函数用于将每次随机生成的数字保存到txt文件中去，这样后续可以验证生成的数字图像是否是我们希望生成的;**

**option=2:这种情况下，不会生成一张大图，而是生成含有batch\_size帧的gif图，默认时间是2s,如果生成gif失败，则会生成和option=1一样的大图;**

**option=3:不能是mnist数据集，生成和option=2一样的gif。**

**option=4:合成一张大图的gif,一共有batch\_size个大图,每个大图由z\_dim（生成样本数目)个小图组成。**

def visualize(sess, dcgan, config, option, sample\_dir='samples'):

image\_frame\_dim = int(math.ceil(config.batch\_size\*\*.5))

if option == 0:

z\_sample = np.random.uniform(-0.5, 0.5, size=(config.batch\_size, dcgan.z\_dim))

samples = sess.run(dcgan.sampler, feed\_dict={dcgan.z: z\_sample})

save\_images(samples, [image\_frame\_dim, image\_frame\_dim], os.path.join(sample\_dir, 'test\_%s.png' % strftime("%Y%m%d%H%M%S", gmtime() )))

elif option == 1:

values = np.arange(0, 1, 1./config.batch\_size)

for idx in xrange(dcgan.z\_dim):

print(" [\*] %d" % idx)

z\_sample = np.random.uniform(-1, 1, size=(config.batch\_size , dcgan.z\_dim))

for kdx, z in enumerate(z\_sample):

z[idx] = values[kdx]

samples = sess.run(dcgan.sampler, feed\_dict={dcgan.z: z\_sample})

save\_images(samples, [image\_frame\_dim, image\_frame\_dim], os.path.join(sample\_dir, 'test\_arange\_%s.png' % (idx)))

elif option == 2:

values = np.arange(0, 1, 1./config.batch\_size)

for idx in [random.randint(0, dcgan.z\_dim - 1) for \_ in xrange(dcgan.z\_dim)]:

print(" [\*] %d" % idx)

z = np.random.uniform(-0.2, 0.2, size=(dcgan.z\_dim))

z\_sample = np.tile(z, (config.batch\_size, 1))

#z\_sample = np.zeros([config.batch\_size, dcgan.z\_dim])

for kdx, z in enumerate(z\_sample):

z[idx] = values[kdx]

samples = sess.run(dcgan.sampler, feed\_dict={dcgan.z: z\_sample})

try:

make\_gif(samples, './samples/test\_gif\_%s.gif' % (idx))

except:

save\_images(samples, [image\_frame\_dim, image\_frame\_dim], os.path.join(sample\_dir, 'test\_%s.png' % strftime("%Y%m%d%H%M%S", gmtime() )))

elif option == 3:

values = np.arange(0, 1, 1./config.batch\_size)

for idx in xrange(dcgan.z\_dim):

print(" [\*] %d" % idx)

z\_sample = np.zeros([config.batch\_size, dcgan.z\_dim])

for kdx, z in enumerate(z\_sample):

z[idx] = values[kdx]

samples = sess.run(dcgan.sampler, feed\_dict={dcgan.z: z\_sample})

make\_gif(samples, os.path.join(sample\_dir, 'test\_gif\_%s.gif' % (idx)))

elif option == 4:

image\_set = []

values = np.arange(0, 1, 1./config.batch\_size)

for idx in xrange(dcgan.z\_dim):

print(" [\*] %d" % idx)

z\_sample = np.zeros([config.batch\_size, dcgan.z\_dim])

for kdx, z in enumerate(z\_sample): z[idx] = values[kdx]

image\_set.append(sess.run(dcgan.sampler, feed\_dict={dcgan.z: z\_sample}))

make\_gif(image\_set[-1], os.path.join(sample\_dir, 'test\_gif\_%s.gif' % (idx)))

new\_image\_set = [merge(np.array([images[idx] for images in image\_set]), [10, 10]) \

for idx in range(64) + range(63, -1, -1)]

make\_gif(new\_image\_set, './samples/test\_gif\_merged.gif', duration=8)

**定义image\_manifold\_size(num\_images)函数。首先获取图像数量的开平方后向下取整的h和向上取整的w，然后设置一个assert断言，如果h\*w与图像数量相等，则返回h和w，否则断言错误提示**

def image\_manifold\_size(num\_images):

manifold\_h = int(np.floor(np.sqrt(num\_images)))

manifold\_w = int(np.ceil(np.sqrt(num\_images)))

assert manifold\_h \* manifold\_w == num\_images

return manifold\_h, manifold\_w

**这就是全部utils.py全部内容，主要负责图像的一些基本操作，获取图像、保存图像、图像翻转，和利用moviepy模块可视化训练过程**