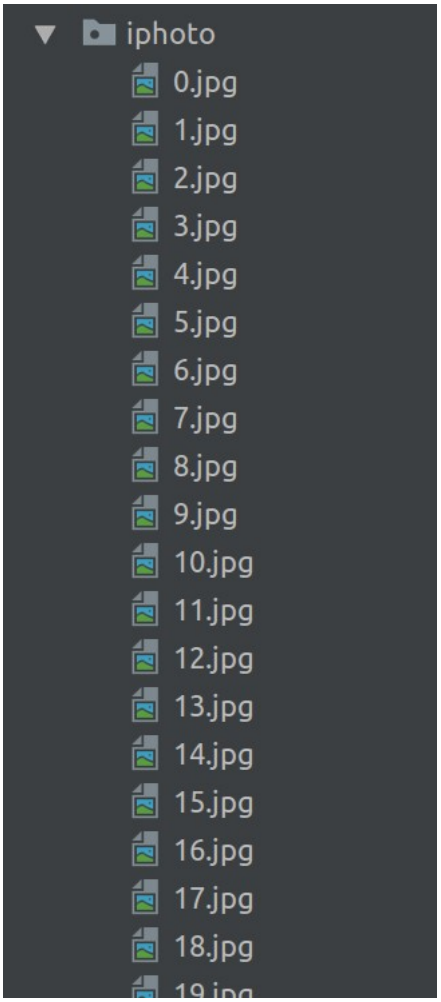

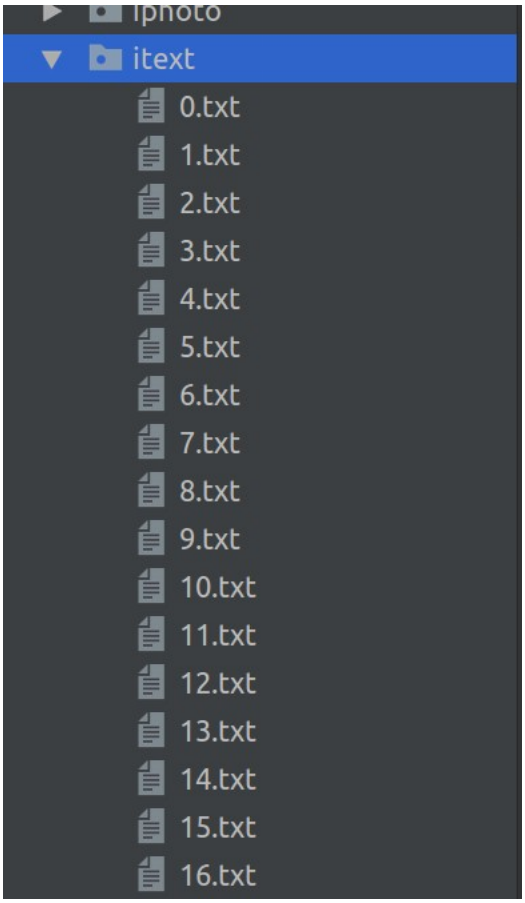

----------保存 tfrecord----------

```
import tensorflow as tf
import numpy as np
import os
from PIL import Image
import sys
TFRCORD_FILE_DIR = 'tfrecord/'#tf文件目录
IMAGE_DIR = '../iphoto/'      #图像所在
LABEL_DIR = '../itext/'           #标签所在
NUM_TEST = 200                    #测试集大小
```

```
filePath = "../txt/"   #所有的汉字的目录文件
def _gen_dict():
  dic = []
  for filename in os.listdir(filePath):
    if not filename.startswith("."):
      with open(filePath + filename,
encoding='utf-8') as file:
        for line in file.readlines():
          content = line.split(",")[8]
          if "###" not in content:
            for c in content:
              if c != '\n':
                dic.append(c)
  for c in [chr(x) for x in range(33, 127)]:
    dic.append(c)
  d = list(set(dic))
  with open("tf_dictset.txt", 'w+',
encoding='utf-8') as out:  #所有汉字集中存放文件
    for c in d:
      out.write(c + '\n')
  return d
def _tfrecord_exist(check_dir):
  for type in ['train','test']:
    path_name =
os.path.join(check_dir,type+'.tfrecords')
    if tf.gfile.Exists(path_name):
      return True
  return False
def _get_paths_image_or_label(root_dir):
  path_list =[]
  for filename in os.listdir(root_dir):
    dir = os.path.join(root_dir,filename)
    path_list.append(dir)
  return path_list
```

```
def _bytes_feature(value):
  return
tf.train.Feature(bytes_list=tf.train.BytesList(val
ue=[value]))
def _int64_feature(value):
  if not isinstance(value,(tuple,list)):
    value=[value]
  return tf.train.Feature(int64_list =
tf.train.Int64List(value=value))
def _gen_example_by_data(image_data,label_data):
  return
tf.train.Example(features=tf.train.Features(featur
e = {
'image':_bytes_feature(np.array(image_data).tobyte
s()),
'image_h':_int64_feature(np.shape(image_data)[0]),
'image_w':_int64_feature(np.shape(image_data)[1]),
'image_channel':_int64_feature(np.shape(image_data
)[2]),
'label':_int64_feature(label_data)
                                        }))
def _gen_tfrecords(type,image_list,label_list):
  assert type in ['train','test']
  dic = _gen_dict()
  with tf.Session() as sess:
    tfrecord_path_name =
os.path.join(TFRCORD_FILE_DIR,type+'.tfrecords')
    with
tf.python_io.TFRecordWriter(tfrecord_path_name) as
tfile:
      for i,image_name in
zip(range(len(image_list)),image_list):
```

```
            try:
                label_name =
'../itext/'+image_name.split('/')[-1].split('.')
[0]+'.txt'  #一个照片对应相应的label
                print(image_name,label_name)
                sys.stdout.write('\n>已经保存: %d/%d' %
(i+1,len(image_list)))
                sys.stdout.flush()
                image = Image.open(image_name)
                #---
                f = open(label_name, 'r',
encoding='utf-8')
                text = f.readline().strip()
                label_data = [dic.index(x) + 1 for x
in text]
                f.close()
                #---
                example =
_gen_example_by_data(image,label_data)
tfile.write(example.SerializeToString())
            except:
                print('转化tfrecords文件出错')
                tfile.close()
                return
    tfile.close()
if __name__ =='__main__':
  if _tfrecord_exist(TFRCORD_FILE_DIR):
    print('tfrecord file exist!')
  else:
    #获取所有图片、标签的路径
    all_image_paths=
_get_paths_image_or_label(IMAGE_DIR)
```

```
    all_label_paths=
_get_paths_image_or_label(LABEL_DIR)
    #划分训练集和测试集
train_image_paths_list=all_image_paths[NUM_TEST:]
test_image_paths_list=all_image_paths[:NUM_TEST]
train_label_paths_list=all_label_paths[NUM_TEST:]
test_label_paths_list=all_label_paths[:NUM_TEST]
    #生成训练集的TF文件
_gen_tfrecords('train',train_image_paths_list,trai
n_label_paths_list)
    #生成测试集的TF文件
_gen_tfrecords('test',test_image_paths_list,test_l
abel_paths_list)
    print('\n完毕! ')
```

----------读取 tfrecord----------

```
import tensorflow as tf
from matplotlib import pyplot as plt
batch_size=200
min_after_dequeue=1000
reader = tf.TFRecordReader()
filename_queue =
tf.train.string_input_producer(['tfrecord/train.tf
records'])
_,serialized_example =
reader.read(queue=filename_queue)
features =
tf.parse_single_example(serialized_example,feature
s = {
        'image':tf.FixedLenFeature([],tf.string),
```

```
        'image_h':tf.FixedLenFeature([],tf.int64),
        'image_w':tf.FixedLenFeature([],tf.int64),
'image_channel':tf.FixedLenFeature([],tf.int64),
        'label':tf.VarLenFeature(tf.int64)
        })
image = tf.decode_raw(features['image'],tf.uint8)
image_h = tf.cast(features['image_h'],tf.int64)
image_w = tf.cast(features['image_w'],tf.int64)
image_channel =
tf.cast(features['image_channel'],tf.int64)
shape = tf.stack([image_h,image_w,image_channel])
image = tf.reshape(image,[32,192,3])
label_data = features['label']
image_batch,label_batch  =
tf.train.shuffle_batch([image,label_data],batch_si
ze=batch_size,min_after_dequeue=1000,capacity=min_
after_dequeue+3*batch_size,num_threads=1)
with tf.Session() as sess:
  coord = tf.train.Coordinator()
  threads =
tf.train.start_queue_runners(sess=sess,coord=coord
)
  for _ in range(10):
    img,lab = sess.run([image_batch,label_batch])
print(sess.run(tf.sparse_to_dense(lab.indices,lab.
dense_shape,lab.values)))
    print('----------')
  coord.request_stop()
  coord.join(threads)
```