```python
import os
import tensorflow as tf
os.environ["CUDA_VISIBLE_DEVICES"] = "1"
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import random,math,sys
from PIL import Image
import numpy as np

gpu_options = tf.compat.v1.GPUOptions(per_process_gpu_memory_fraction=0.333)

_NUM_TEST = 500
_RANDOM_SEED = 0
_NUM_SHARDS = 5

#dataset path
DATASET_DIR = r'captcha/images/'
#tfrecord where to save
TFRECORD_DIR = 'captcha/'

#判断 tfrecord is exists
def _dataset_exists(dataset_dir):
    for split_name in ['train', 'test']:
        output_filename = os.path.join(dataset_dir, split_name, '.tfrecords')
        if not tf.gfile.Exists(output_filename):
            return  False
    return True

#get all v_code's path
def _get_filename_and_classes(dataset_dir):
    photo_filenames = []
    for filename in os.listdir(dataset_dir):
        path = os.path.join(dataset_dir,filename)
        photo_filenames.append(path)
    return photo_filenames

def int64_feature(values):
    if not isinstance(values, (tuple,list)):
        values = [values]
    return tf.train.Feature(int64_list = tf.train.Int64List(value=values))

def bytes_feature(values):
    return  tf.train.Feature(bytes_list = tf.train.BytesList(value=[values]))

def image_to_tfexample(image_data, label0,label1,label2,label3,label4):
    return tf.train.Example(features=tf.train.Features(feature={
        'image':bytes_feature(image_data), #bytes类型   int bytes float 可以有三种类型
        'label0':int64_feature(label0),
        'label1': int64_feature(label1),
        'label2': int64_feature(label2),
```

```
        'label3': int64_feature(label3),
        'label4': int64_feature(label4),
    }))
#为什么要拆成5位呢？ 而不是1位呢？ 是为了多任务的方式。

#把数据转换为TFRecord格式
def _covert_dataset(split_name, filenames, dataset_dir):
    assert split_name in ['train', 'test']
    #计算每个数据块有多少数据（数据量比较大的时候才需要切分）
    with tf.compat.v1.Session(config=tf.compat.v1.ConfigProto(gpu_options=gpu_options)) as sess:
        output_filename = os.path.join(TFRECORD_DIR,split_name+'.tfrecords')
        with tf.python_io.TFRecordWriter(output_filename) as tfrecord_writer:
            for i, filename in enumerate(filenames):
                try:
                    sys.stdout.write('\r >> Converting image %d/%d %s' % (i+1, len(filenames), filename))
                    sys.stdout.flush()

                    #读取图片
                    image_data = Image.open(filename)
                    image_data = image_data.resize((224,224)) # 160*60
                    image_data = np.array(image_data.convert('L')) #灰度化
                    image_data = image_data.tobytes() #转化为bytes

                    #获取label
                    labels = filename.split('/')[-1][0:5]
                    num_labels = []
                    for j in range(5):
                        str = labels[j]
                        if str.isdigit():
                            num_labels.append(int(str))
                        elif str.isalpha():
                            num_labels.append(ord(str))

                    #生成protocol数据
                    example = image_to_tfexample(image_data, num_labels[0], num_labels[1], num_labels[2], num_labels[3],num_labels[4])
                    tfrecord_writer.write(example.SerializeToString())
                except IOError as err:
                    print("Could not read:", filenames[i])
                    print("Erroe:", err)
                    print("skip it \n")

    sys.stdout.write('\n')
    sys.stdout.flush()


if __name__ == '__main__':
    #判断tfrecord是否存在
    if _dataset_exists(DATASET_DIR):
        print('tfrecord is Exists')
    else:
        #获得所有图片以及分类
        photo_filenames = _get_filename_and_classes(DATASET_DIR)
```

```
        #把分类转换为字典格式，类似于{'house':3, 'flower':1, 'plane':4,
'guitar':2, 'animal':0}
        class_name_to_ids = dict(zip(class_name, range(len(class_names))))

        #把数据切分为训练集和测试集
        random.seed(_RANDOM_SEED)
        random.shuffle(photo_filenames)
        training_filenames = photo_filenames[_NUM_TEST:]
        testing_filenames  = photo_filenames[:_NUM_TEST]

        #数据转换
        _covert_dataset('train', training_filenames, DATASET_DIR)
        _covert_dataset('test', testing_filenames, DATASET_DIR)

        #输出labels文件
        labels_to_class_names =
dict(zip(range(len(class_names)),class_names))
        write_label_file(labels_to_class_names, DATASET_DIR)
    print('produce tfrecord sucessful')


```