lable_process.py

```
import csv
import os

"""

数据分类

"""

# 根据csv文件中的属性标签，将图片根据其类别分配到相应的文件中去，映射关系使用map做存储
label_csv_file_path =
"../FashionAI_data/warm_up_train_20180201/web/Annotations/skirt_length_labe
ls.csv"
map_label_picture = {}
with open(label_csv_file_path, 'rt') as label_csv_file:
    rows = csv.reader(label_csv_file, delimiter=',')
    for row in rows:
        picture = row[0]
        category = row[1]
        labels = row[2]
        label = str(labels).index('y')
        if label not in map_label_picture:
            map_label_picture[label] = [picture]
        else:
            map_label_picture[label].extend([picture])

print(map_label_picture.keys())

# 根据map中k-v的映射关系，将图片复制到对应的类别文件中去
for k, v in map_label_picture.items():
    label_dir =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed/%d/" % k
    if not os.path.exists(label_dir):
        os.mkdir(label_dir)
    print(label_dir)
    for p in v:
        picture_name = p.split("/")[-1]
        related_picture_path =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels/"
\
                               + picture_name
        if not os.path.exists(related_picture_path):
            continue
        os.rename(related_picture_path, label_dir + picture_name)
```

data_augmentation.py

```
import tensorflow as tf
import os
import cv2


"""

对数据做预处理

"""



flags = tf.app.flags
FLAGS = flags.FLAGS
flags.DEFINE_boolean('random_flip_up_down', True, 'If uses flip')
flags.DEFINE_boolean('random_flip_left_right', True, 'If uses flip')
flags.DEFINE_boolean('random_brightness', True, 'If uses brightness')
flags.DEFINE_boolean('random_contrast', True, 'If uses contrast')
flags.DEFINE_boolean('random_saturation', True, 'If uses saturation')
flags.DEFINE_integer('image_size', 2048, 'image size.')

"""
#flags examples
flags.DEFINE_float('learning_rate', 0.01, 'Initial learning rate.')
flags.DEFINE_integer('max_steps', 2000, 'Number of steps to run trainer.')
flags.DEFINE_string('train_dir', 'data', 'Directory to put the training
data.')
flags.DEFINE_boolean('fake_data', False, 'If true, uses fake data for unit
testing.')
"""

new_size = tf.constant(value=[FLAGS.image_size, FLAGS.image_size],
dtype=tf.int32)

# 图片预处理,增加随机性
def pre_process(images_):
    # if FLAGS.random_flip_up_down:
    #     images_ = tf.image.random_flip_up_down(images_)
    if FLAGS.random_flip_left_right:
        images_ = tf.image.random_flip_left_right(images_)
    # 在某范围随机调整图片亮度
    if FLAGS.random_brightness:
        images_ = tf.image.random_brightness(images_, max_delta=0.1)
    # 在某范围随机调整图片对比度
    if FLAGS.random_contrast:
        images_ = tf.image.random_contrast(images_, 1.0, 1.3)
    # 在某范围随机调整图片饱和度
    if FLAGS.random_saturation:
        tf.image.random_saturation(images_, 0.2, 0.4)
    # 改变图像大小为[Flags.image_size,Flags.image_size]
    images_ = tf.image.resize_images(images_, new_size)
    return images_


# 将原数据集扩大9倍
SIZE = 9

# image = tf.Variable(raw_image)
image = tf.placeholder("uint8", [None, None, 3])
images = pre_process(image)
count = 1
with tf.Session() as session:

    for i in range(0, 6):
        dir_non_cancer_jpeg =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed/%d/" % i
        dir_non_cancer_jpeg_result =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed_data_augmentation/%d/" % i
        if not os.path.exists(dir_non_cancer_jpeg_result):
            os.makedirs(dir_non_cancer_jpeg_result)
        for c_non_cancer in os.listdir(dir_non_cancer_jpeg):
            raw_image = cv2.imread(os.path.join(dir_non_cancer_jpeg,
c_non_cancer))
            for s in range(SIZE):
                result = session.run(images, feed_dict={image: raw_image})
                cv2.imwrite(os.path.join(dir_non_cancer_jpeg_result, str(s)
+ c_non_cancer), result)
            print(str(count) + "\t" + c_non_cancer)
            count += 1

# cv2.imshow("image", result.astype(np.uint8))
# cv2.waitKey(1000)
```

medule.py

```
# coding: utf-8

import tensorflow as tf
from PIL import Image
import numpy as np
import os
import cv2
import random

"""

前向传播、反向传播、模型训练

"""

BATCH_SIZE = 1


# 权值
def weight_variable(shape):
    # tf.truncated_normal(shape, mean, stddev)产生服从正态分布的数据
    initial = tf.truncated_normal(shape=shape, stddev=0.01)
    return tf.Variable(initial_value=initial)


# 偏置值
def bias_variable(shape):
    initial = tf.constant(0.001, shape=shape)
    return tf.Variable(initial_value=initial)


# 卷积层
def conv2d_2x2(x_conv2d, w_conv2d):
    # x:       `[batch, in_height, in_width, in_channels]`
    #           [批次大小, 输入图片的长和宽, 通道数 (黑白:2; 彩色: 3)]
    # W:       `[filter_height, filter_width, in_channels, out_channels]`
    #           [滤波器长, 宽, 输入通道数, 输出通道数]
    # strides: `[1, stride, stride, 1]`
    #           [固定为1, x/y方向的步长, 固定为1]
    # padding:  是否在外部补零
    return tf.nn.conv2d(x_conv2d, w_conv2d, strides=[1, 2, 2, 1],
padding='SAME')


def conv2d_1x1(x_conv2d, w_conv2d):
    return tf.nn.conv2d(x_conv2d, w_conv2d, strides=[1, 1, 1, 1],
padding='SAME')


# 池化层
def avg_pool_2x2(x_pool):
    # x:       `[batch, in_height, in_width, in_channels]
    #           [批次大小, 输入图片的长和宽, 通道数 (黑白:2; 彩色: 3)]
    # ksize:    [固定为1, 窗口大小, 固定为1]
    # strides: `[1, stride, stride, 1]`
    #           [固定为1, x/y方向的步长, 固定为1]
    # padding:  是否在外部补零
    return tf.nn.avg_pool(x_pool, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
padding='SAME')


# 池化层
def max_pool_2x2(x_pool):
    return tf.nn.max_pool(x_pool, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
padding='SAME')


# 池化层
def avg_pool_4x4(x_pool):
    return tf.nn.avg_pool(x_pool, ksize=[1, 4, 4, 1], strides=[1, 4, 4, 1],
padding='SAME')


# 池化层
def avg_pool_5x5(x_pool):
    return tf.nn.avg_pool(x_pool, ksize=[1, 5, 5, 1], strides=[1, 5, 5, 1],
padding='SAME')


# Place Holder
x = tf.placeholder(tf.float32, [2048, 2048, 3])
y = tf.placeholder(tf.float32, [1, 6])

# Learn Rate学习率
lr = tf.Variable(0.0001, dtype=tf.float32,trainable=True)

# 将 x的转化为 4D向量
# [batch, in_height, in_width, in_channels]
x_image = tf.reshape(x, [-1, 2048, 2048, 3])

x_image_min =
max_pool_2x2(max_pool_2x2(max_pool_2x2(max_pool_2x2(x_image))))

# 【第0个池化层】
```

```
W_conv0 = weight_variable([3, 3, 3, 64])
b_conv0 = bias_variable([64])
# (2048-3+1)/2 = 1023
conv2d_0 = conv2d_2x2(x_image_min, W_conv0) + b_conv0
h_conv0 = tf.nn.relu(conv2d_0)
W_conv0 = weight_variable([3, 3, 64, 64])
b_conv0 = bias_variable([64])
# (1023-3+1)/1 = 1021
conv2d_01 = conv2d_1x1(h_conv0, W_conv0) + b_conv0
h_conv01 = tf.nn.relu(conv2d_01)
# 1021/2 = 510.5 = 511
h_pool0 = max_pool_2x2(h_conv01)

# 【第1个池化层】
W_conv1 = weight_variable([3, 3, 64, 128])
b_conv1 = bias_variable([128])
# (511-3+1)/1 = 509
conv2d_1 = conv2d_1x1(h_pool0, W_conv1) + b_conv1
h_conv1 = tf.nn.relu(conv2d_1)
W_conv1 = weight_variable([3, 3, 128, 128])
b_conv1 = bias_variable([128])
# (509-3+1)/2 = 253.5 = 254
conv2d_1 = conv2d_1x1(h_conv1, W_conv1) + b_conv1
h_conv11 = tf.nn.relu(conv2d_1)
# 254/2 = 127
h_pool1 = max_pool_2x2(h_conv11)

# 【第2个卷积层】
W_conv2 = weight_variable([3, 3, 128, 256])
b_conv2 = bias_variable([256])
# (127-3+1)/2 = 62.5 = 63
conv2d_2 = conv2d_1x1(h_pool1, W_conv2) + b_conv2
h_conv2 = tf.nn.relu(conv2d_2)
W_conv2 = weight_variable([3, 3, 256, 256])
b_conv2 = bias_variable([256])
# (63-3+1)/1 = 61
conv2d_21 = conv2d_1x1(h_conv2, W_conv2) + b_conv2
h_conv21 = tf.nn.relu(conv2d_21)
W_conv2 = weight_variable([1, 1, 256, 256])
b_conv2 = bias_variable([256])
# (61-3+1)/2 = 29.5 = 30
conv2d_22 = conv2d_1x1(h_conv21, W_conv2) + b_conv2
h_conv22 = tf.nn.relu(conv2d_22)
# 30/2 = 15
h_pool2 = max_pool_2x2(h_conv22)

# 【第3个卷积层 + 激活函数 + 池化层】
W_conv3 = weight_variable([3, 3, 256, 512])
b_conv3 = bias_variable([512])
```

```
# (15-3+1)/2 = 6.5 = 7
conv2d_3 = conv2d_1x1(h_pool2, W_conv3) + b_conv3
h_conv3 = tf.nn.relu(conv2d_3)

W_conv3 = weight_variable([3, 3, 512, 512])
b_conv3 = bias_variable([512])
# (7-3+1)/1 = 5
conv2d_31 = conv2d_1x1(h_conv3, W_conv3) + b_conv3
h_conv31 = tf.nn.relu(conv2d_31)

W_conv3 = weight_variable([1, 1, 512, 512])
b_conv3 = bias_variable([512])
# (5-3+1)/1 = 3
conv2d_32 = conv2d_1x1(h_conv31, W_conv3) + b_conv3
h_conv3 = tf.nn.relu(conv2d_32)
# 3/2 = 1.5 = 2
h_pool3 = max_pool_2x2(h_conv3)

# 【第4个卷积层 + 激活函数 + 池化层】
W_conv4 = weight_variable([3, 3, 512, 512])
b_conv4 = bias_variable([512])
# (2-3+1)/2 =
conv2d_4 = conv2d_1x1(h_pool3, W_conv4) + b_conv4
h_conv4 = tf.nn.relu(conv2d_4)

W_conv4 = weight_variable([3, 3, 512, 512])
b_conv4 = bias_variable([512])
conv2d_41 = conv2d_1x1(h_conv4, W_conv4) + b_conv4
h_conv41 = tf.nn.relu(conv2d_41)

W_conv4 = weight_variable([1, 1, 512, 512])
b_conv4 = bias_variable([512])
conv2d_42 = conv2d_1x1(h_conv41, W_conv4) + b_conv4
h_conv42 = tf.nn.relu(conv2d_42)
h_pool4 = max_pool_2x2(h_conv42)

# 【第1层 全连接层】
# 全连接层一共有 100个神经元，连接上一层的 13 * 13 * 120 = 20280个神经元
W_fc1 = weight_variable([2 * 2 * 512, 1024])
b_fc1 = bias_variable([1024])

# 把上一层的池化层，转化为 1维 (-1代表任意值)
h_pool_flat = tf.reshape(h_pool4, [-1, 2 * 2 * 512])
# 矩阵相乘，并加上偏置值
wx_plus_b1 = tf.matmul(h_pool_flat, W_fc1) + b_fc1
# ReLU激活函数
h_fc1 = tf.nn.relu(wx_plus_b1)
# dropout正则化，Dropout层减轻过拟合。Dropout，通过一个placeholder传入keep_prob比
率控制
```

```
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

# 【第2层 全连接层】
W_fc2 = weight_variable([1024, 512])
b_fc2 = bias_variable([512])
wx_plus_b2 = tf.matmul(h_fc1_drop, W_fc2) + b_fc2
h_fc2 = tf.nn.relu(wx_plus_b2)

# dropout正则化
h_fc2_drop = tf.nn.dropout(h_fc2, keep_prob)

# 【第3层 全连接层】
W_fc3 = weight_variable([512, 6])
b_fc3 = bias_variable([6])
# 计算输出
wx_plus_b3 = tf.matmul(h_fc2_drop, W_fc3) + b_fc3
prediction = tf.nn.softmax(wx_plus_b3)

# 交叉熵 Loss Function
cross_entropy =
tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y,
logits=wx_plus_b3))

# Adam优化器，配合一个不断下降的学习率
train_step = tf.train.AdagradOptimizer(lr).minimize(cross_entropy)

# argmax方法，会返回一维张量中最大值所在的位置
# 计算正确率
correct_prediction = tf.equal(tf.argmax(prediction, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# 胃癌数字病理样本，为常规 HE 染色，放大倍数 20×，图片大小为 2048×2048 = 4194304像素
pictures_0 =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed_data_augmentation/0"
pictures_1 =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed_data_augmentation/1"
pictures_2 =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed_data_augmentation/2"
pictures_3 =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed_data_augmentation/3"
pictures_4 =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed_data_augmentation/4"
```

```
pictures_5 =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_in
dexed_data_augmentation/5"

# defaults to saving all variables - in this case w and b
saver = tf.train.Saver()

# 560 + 140，分两组，就是 训练 :  测试 = 400 + 160 : 100 + 40
with tf.device('/cpu'):
    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    with tf.Session(config=config) as sess:
        tf.global_variables_initializer().run()

        label_0 = np.reshape([[1, 0, 0, 0, 0, 0]], (1, 6))
        label_1 = np.reshape([[0, 1, 0, 0, 0, 0]], (1, 6))
        label_2 = np.reshape([[0, 0, 1, 0, 0, 0]], (1, 6))
        label_3 = np.reshape([[0, 0, 0, 1, 0, 0]], (1, 6))
        label_4 = np.reshape([[0, 0, 0, 0, 1, 0]], (1, 6))
        label_5 = np.reshape([[0, 0, 0, 0, 0, 1]], (1, 6))
        print("label 0", label_0)
        print("label 1", label_1)
        print("label 2", label_2)
        print("label 3", label_3)
        print("label 4", label_4)
        print("label 5", label_5)

        # 获取到每个目录下的图片：picture0、picture1 ..... picture5
        label_0_list = os.listdir(pictures_0)
        label_0_list_len = len(label_0_list)
        label_1_list = os.listdir(pictures_1)
        label_1_list_len = len(label_1_list)
        label_2_list = os.listdir(pictures_2)
        label_2_list_len = len(label_2_list)
        label_3_list = os.listdir(pictures_3)
        label_3_list_len = len(label_3_list)
        label_4_list = os.listdir(pictures_4)
        label_4_list_len = len(label_4_list)
        label_5_list = os.listdir(pictures_5)
        label_5_list_len = len(label_5_list)

        bucket = int(label_0_list_len / BATCH_SIZE)

        count = 0

        # 可以一直训练下去
        for _ in range(999999):
            # tf.assign(A, new_number): 这个函数的功能主要是把A的值变为
new_number
```

```
            sess.run(tf.assign(lr, 0.001 * (0.95 ** _)))

            # sample(seq, n) 从序列seq中选择n个随机且独立的元素；这里是对每一bucket
中的数据进行随机抽取训练
            for bucket_index in random.sample(range(bucket), bucket):
                # label_0
                image_label_0 = None
                for b_label_0 in range(bucket_index * BATCH_SIZE,
(bucket_index + 1) * BATCH_SIZE):
                    if b_label_0 < label_0_list_len:
                        file = label_0_list[b_label_0]
                        line = np.array(cv2.imread(os.path.join(pictures_0,
file)))
                        if image_label_0 is None:
                            image_label_0 = line
                            continue
                        np.row_stack((image_label_0, line))
                if image_label_0 is None:
                    continue
                sess.run(train_step, feed_dict={x: image_label_0, y:
label_0, keep_prob: 0.5})

                if count % 100 == 0:
                    test_acc, pred, y_ = sess.run([accuracy, prediction,
y],
                                                  feed_dict={x:
image_label_0, y: label_0, keep_prob: 1.0})
                    print("Iterator:", _ + 1, "Count:", count,
                          "Bucket:", bucket_index, "Accuracy:", test_acc,
"label:", y_,
                          "Prediction", pred, "File:", file)

                # label_1
                image_label_1 = None
                for b_label_1 in range(bucket_index * BATCH_SIZE,
(bucket_index + 1) * BATCH_SIZE):
                    if b_label_1 < label_1_list_len:
                        file = label_1_list[b_label_1]
                        line = np.array(cv2.imread(os.path.join(pictures_1,
file)))
                        if image_label_1 is None:
                            image_label_1 = line
                            continue
                        np.row_stack((image_label_1, line))
                if image_label_1 is None:
                    continue
                sess.run(train_step, feed_dict={x: image_label_1, y:
label_1, keep_prob: 0.5})
```

```
                if count % 100 == 0:
                    test_acc, pred, y_ = sess.run([accuracy, prediction,
y],
                                                  feed_dict={x:
image_label_1, y: label_1, keep_prob: 1.0})
                    print("Iterator:", _ + 1, "Count:", count,
                          "Bucket:", bucket_index, "Accuracy:", test_acc,
"label:", y_,
                          "Prediction", pred, "File:", file)

                # label_2
                image_label_2 = None
                for b_label_2 in range(bucket_index * BATCH_SIZE,
(bucket_index + 1) * BATCH_SIZE):
                    if b_label_2 < label_2_list_len:
                        file = label_2_list[b_label_2]
                        line = np.array(cv2.imread(os.path.join(pictures_2,
file)))
                        if image_label_2 is None:
                            image_label_2 = line
                            continue
                        np.row_stack((image_label_2, line))
                if image_label_2 is None:
                    continue
                sess.run(train_step, feed_dict={x: image_label_2, y:
label_2, keep_prob: 0.5})

                if count % 100 == 0:
                    test_acc, pred, y_ = sess.run([accuracy, prediction,
y],
                                                  feed_dict={x:
image_label_2, y: label_2, keep_prob: 1.0})
                    print("Iterator:", _ + 1, "Count:", count,
                          "Bucket:", bucket_index, "Accuracy:", test_acc,
"label:", y_,
                          "Prediction", pred, "File:", file)

                # label_3
                image_label_3 = None
                for b_label_3 in range(bucket_index * BATCH_SIZE,
(bucket_index + 1) * BATCH_SIZE):
                    if b_label_3 < label_3_list_len:
                        file = label_3_list[b_label_3]
                        line = np.array(cv2.imread(os.path.join(pictures_3,
file)))
                        if image_label_3 is None:
                            image_label_3 = line
                            continue
                        np.row_stack((image_label_3, line))
                if image_label_3 is None:
                    continue
                sess.run(train_step, feed_dict={x: image_label_3, y:
label_3, keep_prob: 0.5})

                if count % 100 == 0:
                    test_acc, pred, y_ = sess.run([accuracy, prediction,
y],
                                                  feed_dict={x:
image_label_3, y: label_3, keep_prob: 1.0})
                    print("Iterator:", _ + 1, "Count:", count,
                          "Bucket:", bucket_index, "Accuracy:", test_acc,
"label:", y_,
                          "Prediction", pred, "File:", file)

                # label_4
                image_label_4 = None
                for b_label_4 in range(bucket_index * BATCH_SIZE,
(bucket_index + 1) * BATCH_SIZE):
                    if b_label_4 < label_4_list_len:
                        file = label_4_list[b_label_4]
                        line = np.array(cv2.imread(os.path.join(pictures_4,
file)))
                        if image_label_4 is None:
                            image_label_4 = line
                            continue
                        np.row_stack((image_label_4, line))
                if image_label_4 is None:
                    continue
                sess.run(train_step, feed_dict={x: image_label_4, y:
label_4, keep_prob: 0.5})

                if count % 100 == 0:
                    test_acc, pred, y_ = sess.run([accuracy, prediction,
y],
                                                  feed_dict={x:
image_label_4, y: label_4, keep_prob: 1.0})
                    print("Iterator:", _ + 1, "Count:", count,
                          "Bucket:", bucket_index, "Accuracy:", test_acc,
"label:", y_,
                          "Prediction", pred, "File:", file)

                # label_5
                image_label_5 = None
                for b_label_5 in range(bucket_index * BATCH_SIZE,
(bucket_index + 1) * BATCH_SIZE):
                    if b_label_5 < label_5_list_len:
                        file = label_5_list[b_label_5]
                        line = np.array(cv2.imread(os.path.join(pictures_5,
file)))
                        if image_label_5 is None:
                            image_label_5 = line
                            continue
                        np.row_stack((image_label_5, line))
                if image_label_5 is None:
                    continue
                sess.run(train_step, feed_dict={x: image_label_5, y:
label_5, keep_prob: 0.5})

                if count % 100 == 0:
                    test_acc, pred, y_ = sess.run([accuracy, prediction,
y],
                                                  feed_dict={x:
image_label_5, y: label_5, keep_prob: 1.0})
                    print("Iterator:", _ + 1, "Count:", count,
                          "Bucket:", bucket_index, "Accuracy:", test_acc,
"label:", y_,
                          "Prediction", pred, "File:", file)

                count += 1

            final_result_dir_base = 'net1/'
            if not os.path.exists(final_result_dir_base):
                os.makedirs(final_result_dir_base)
            final_result_dir = final_result_dir_base + str(_)
            if not os.path.exists(final_result_dir):
                os.makedirs(final_result_dir)
            saver.save(sess, final_result_dir + '/fashion.ckpt')
```

tfrecord.py

```
# coding: utf-8

import tensorflow as tf
import numpy as np
import os
import cv2
import random



# 将输入转化成TFRecord格式并保存
# 定义函数转化变量类型。
```

```
def _int64_list_feature(value):
    return tf.train.Feature(int64_list=tf.train.Int64List(value=value))


def _int64_feature(value):
    return tf.train.Feature(int64_list=tf.train.Int64List(value=[value]))


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


# 胃癌数字病理样本，为常规 HE 染色，放大倍数 20×，图片大小为 2048×2048 = 4194304像素
dir_cancer_jpeg =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_10
0/"
dir_non_cancer_jpeg =
"../FashionAI_data/warm_up_train_20180201/web/Images/skirt_length_labels_10
0/"
# 输出TFRecord文件的地址
tf_record_dir =
"../FashionAI_data/warm_up_train_20180201_tf_record/web/Images/skirt_length
_labels/"
if not os.path.exists(tf_record_dir):
    os.makedirs(tf_record_dir)
tf_record_filename = tf_record_dir + "output-%.5d.tfrecords"
tf_record_filename_like = tf_record_dir + "*"

label = np.reshape([[1, 0]], (1, 2))
non_label = np.reshape([[0, 1]], (1, 2))
label = label.astype(np.uint8)
non_label = non_label.astype(np.uint8)
print("label", label)
print("non label", non_label)

non_cancer_list = os.listdir(dir_non_cancer_jpeg)
non_cancer_list_len = len(non_cancer_list)
cancer_list = os.listdir(dir_cancer_jpeg)
cancer_list_len = len(cancer_list)

if non_cancer_list_len != cancer_list_len:
    print("non cancer : cancer =", non_cancer_list, " :", cancer_list_len)

RECORD_SIZE = 1
bucket = int(non_cancer_list_len / RECORD_SIZE)
for bucket_index in random.sample(range(bucket), bucket):
    tf_record_filename_tmp = tf_record_filename % bucket_index
    writer = tf.python_io.TFRecordWriter(tf_record_filename_tmp)
    for file_index in range(bucket_index * RECORD_SIZE, (bucket_index + 1)
* RECORD_SIZE):
        cancer_non_jpeg = cv2.imread(os.path.join(dir_non_cancer_jpeg,
non_cancer_list[file_index]))
        cancer_non_example =
tf.train.Example(features=tf.train.Features(feature={
            'label': _bytes_feature(non_label.tostring()),
            'image_raw': _bytes_feature(cancer_non_jpeg.tostring())
        }))
        writer.write(cancer_non_example.SerializeToString())

        cancer_jpeg = cv2.imread(os.path.join(dir_cancer_jpeg,
cancer_list[file_index]))
        cancer_example =
tf.train.Example(features=tf.train.Features(feature={
            'label': _bytes_feature(label.tostring()),
            'image_raw': _bytes_feature(cancer_jpeg.tostring())
        }))
        writer.write(cancer_example.SerializeToString())
    writer.close()
    print(tf_record_filename_tmp + " Saved.")

print("TFRecord All Saved.")

# 创建文件列表，通过文件列表创建输入文件队列
tf_records =
tf.train.match_filenames_once([("../FashionAI_data/warm_up_train_20180201_t
f_record/web/Images/skirt_length_labels/output-%s.tfrecords" %
str(i).zfill(5)) for i in range(300)])
# tf_records = tf.train.match_filenames_once("Records/output-*.tfrecords")
filename_queue = tf.train.string_input_producer(tf_records, shuffle=True)

# 读取TFRecord文件
reader = tf.TFRecordReader()
_, serialized_example = reader.read(filename_queue)

# 解析读取的样例
features = tf.parse_single_example(
    serialized_example,
    features={
        'image_raw': tf.FixedLenFeature([], tf.string),
        'label': tf.FixedLenFeature([], tf.string),
    })

images = tf.decode_raw(features['image_raw'], tf.uint8)
labels = tf.decode_raw(features['label'], tf.uint8)
labels = tf.reshape(labels, (1, 2))

with tf.device('/cpu'):
    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    with tf.Session(config=config) as sess:
        tf.global_variables_initializer().run()

        # 启动多线程处理输入数据
        coord = tf.train.Coordinator()
        threads = tf.train.start_queue_runners(sess=sess, coord=coord)

        for i in range(10):
            images_, labels_, = sess.run([images, labels])
            print(images_, labels_)

        coord.request_stop()
        coord.join(threads)
```