关于简单目标识别与意图分析的机器学习实战研究（神经网络目标识别——改写某博主的简单分类脚本）

这边顺利的完成了透视变换，同学高高兴兴的老师汇报问题解决了，然后又从头讲了遍目标识别的思路，问是不是能进行下一步的意图分析了。老师听完后说不急，告诉同学，要是在目标识别的时候就能用上深度学习的话就好了，可以尝试一下，确实这方面对没学过的同学来说有些难度，但趁现在多试试，多踩坑，不行的话也是有了这么一次经验。然后我就踩了一堆坑......

我寻思反正也没做过，找个前辈的研究改改也就能用吧，抱着这个想法，我找到了第一个开源项目，GitHub上代码地址：[https://github.com/527515025/My-TensorFlow-tutorials/tree/master/猫狗识别](https://github.com/527515025/My-TensorFlow-tutorials/tree/master/%E7%8C%AB%E7%8B%97%E8%AF%86%E5%88%AB)，作者用的是TensorFlow卷积神经网络，这里是作者的博客：<https://blog.csdn.net/u012373815/article/details/78768727>，不过正如大家所看到的那样，这里只能识别猫和狗两种，显然不符合我们的要求，所以我将代码改了改，将输入处改了一下能输入三种数据流，扩大了卷积层，增加了一个神经元输出，训练代码也做出同样的更改，使之能够识别三种物体状态，训练代码更改如下：

1. **import** os
2. **import** numpy as np
3. **import** tensorflow as tf
4. **import** input\_data
5. **import** model
7. N\_CLASSES = 3  # 增加1个输出神经元
8. IMG\_W = 208  # 重新定义图片的大小，图片如果过大则训练比较慢
9. IMG\_H = 208
10. BATCH\_SIZE = 32  #每批数据的大小
11. CAPACITY = 256
12. MAX\_STEP = 15000 # 训练的步数，应当 >= 10000
13. learning\_rate = 0.0001 # 学习率，建议刚开始的 learning\_rate <= 0.0001
14. **def** run\_training():
16. # 数据集
17. train\_dir = 'E:/bishework/lx/all-match/'   #My dir--20170727-csq
18. #logs\_train\_dir 存放训练模型的过程的数据，在tensorboard 中查看
19. logs\_train\_dir = 'E:/bishework/lx/train/'
20. # 获取图片和标签集
21. train, train\_label = input\_data.get\_files(train\_dir)
22. # 生成批次
23. train\_batch, train\_label\_batch = input\_data.get\_batch(train,
24. train\_label,
25. IMG\_W,
26. IMG\_H,
27. BATCH\_SIZE,
28. CAPACITY)
29. # 进入模型
30. train\_logits = model.inference(train\_batch, BATCH\_SIZE, N\_CLASSES)
31. # 获取 loss
32. train\_loss = model.losses(train\_logits, train\_label\_batch)
33. # 训练
34. train\_op = model.trainning(train\_loss, learning\_rate)
35. # 获取准确率
36. train\_\_acc = model.evaluation(train\_logits, train\_label\_batch)
37. # 合并 summary
38. summary\_op = tf.summary.merge\_all()
39. sess = tf.Session()
40. # 保存summary
41. train\_writer = tf.summary.FileWriter(logs\_train\_dir, sess.graph)
42. saver = tf.train.Saver()
44. sess.run(tf.global\_variables\_initializer())
45. coord = tf.train.Coordinator()
46. threads = tf.train.start\_queue\_runners(sess=sess, coord=coord)
48. **try**:
49. **for** step **in** np.arange(MAX\_STEP):
50. **if** coord.should\_stop():
51. **break**
52. \_, tra\_loss, tra\_acc = sess.run([train\_op, train\_loss, train\_\_acc])
54. **if** step % 50 == 0:
55. **print**('Step %d, train loss = %.2f, train accuracy = %.2f%%' %(step, tra\_loss, tra\_acc\*100.0))
56. summary\_str = sess.run(summary\_op)
57. train\_writer.add\_summary(summary\_str, step)
59. **if** step % 200 == 0 **or** (step + 1) == MAX\_STEP:
60. # 每隔200步保存一下模型，模型保存在 checkpoint\_path 中
61. checkpoint\_path = os.path.join(logs\_train\_dir, 'model.ckpt')
62. saver.save(sess, checkpoint\_path, global\_step=step)
64. **except** tf.errors.OutOfRangeError:
65. **print**('Done training -- epoch limit reached')
66. **finally**:
67. coord.request\_stop()
68. coord.join(threads)
69. sess.close()
70. # train
71. run\_training()

然后我兴致勃勃的开始了训练，大约训练了大约60张连线图（数据量确实有点小，这里就试试可靠性），结果却是不如人意，有很大的可能性训练错误，然后我又增加图片，训练了大约一天，得到的结果一直在50%左右晃，忽高忽低，极不理想。

我在评论区也看到了很多人和我一样没有得到想要的结果，所以可能是文篇文章里面的神经网络模型model.py还不够成熟。



如果大家需要，这是我改过的可以识别多种物体状态的代码，希望有人能够增加它的可靠性：

这篇不能就这么糊弄过去了，所以接下来还是好好研究一下关于目标检测的深度学习算法。我去找了下有竞赛经验的老师帮我辅导，老师给我的建议是用当前研究正火的深度学习来解决这道题，也就是说：

数据 + 预测 = 规则

同时给了我以下建议，首先为图片里的目标做标签，而后针对具体题目建模，最后通过模型训练数据，用实验数据进行预测。此外，由于我是在unity游戏环境中截图，比较麻烦，老师还推荐了我使用数据增强的办法，扩充我的数据集。对于我的小笔记本，老师也提供了一个办法，我的显卡还不错1070，而且是NVIDIA的，可以部署tensorflow-gpu环境，成倍的加强我的运算能力，用keras-yolo3训练自己的数据集。下一篇文章将介绍我在部署tensorflow-gpu环境时踩的新坑。