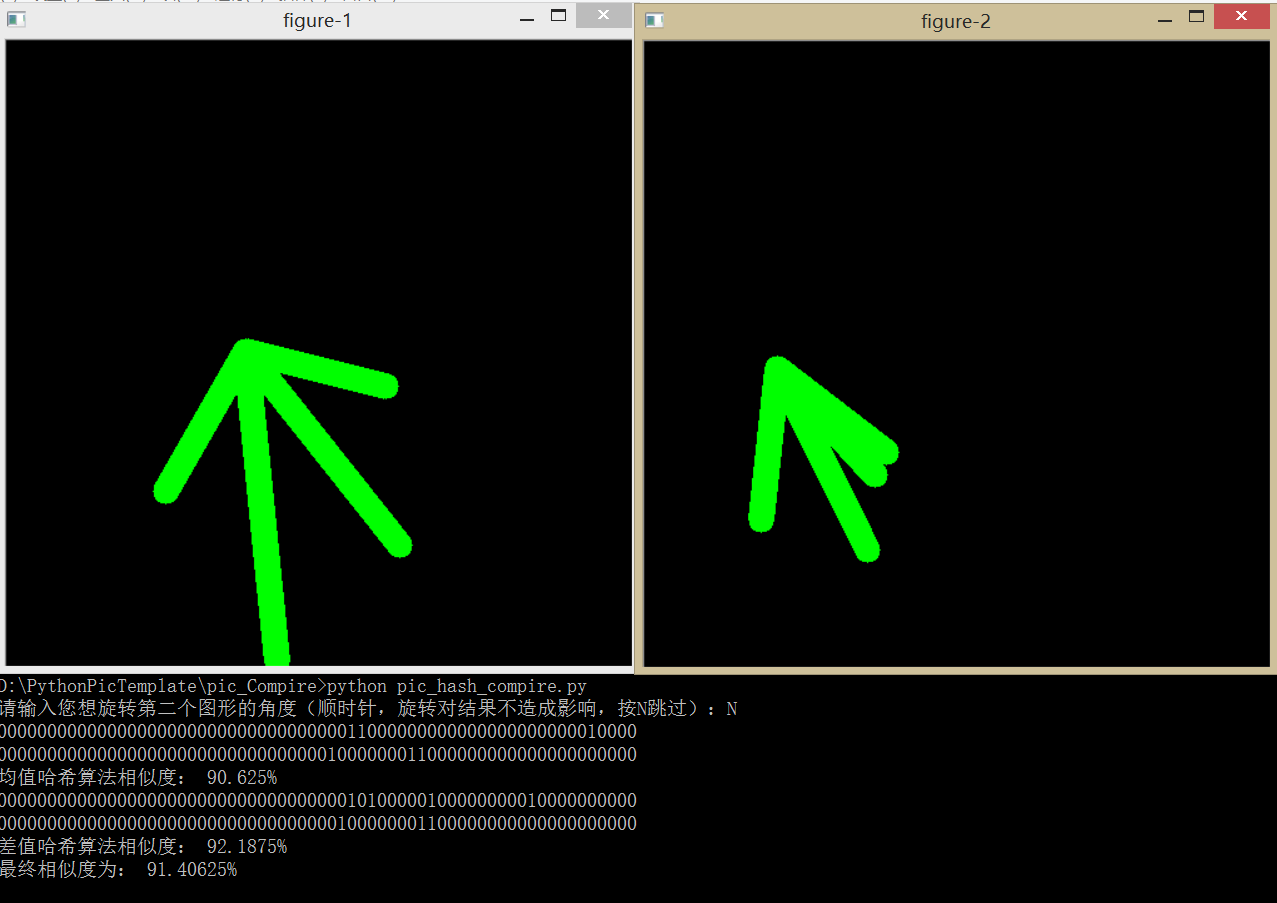
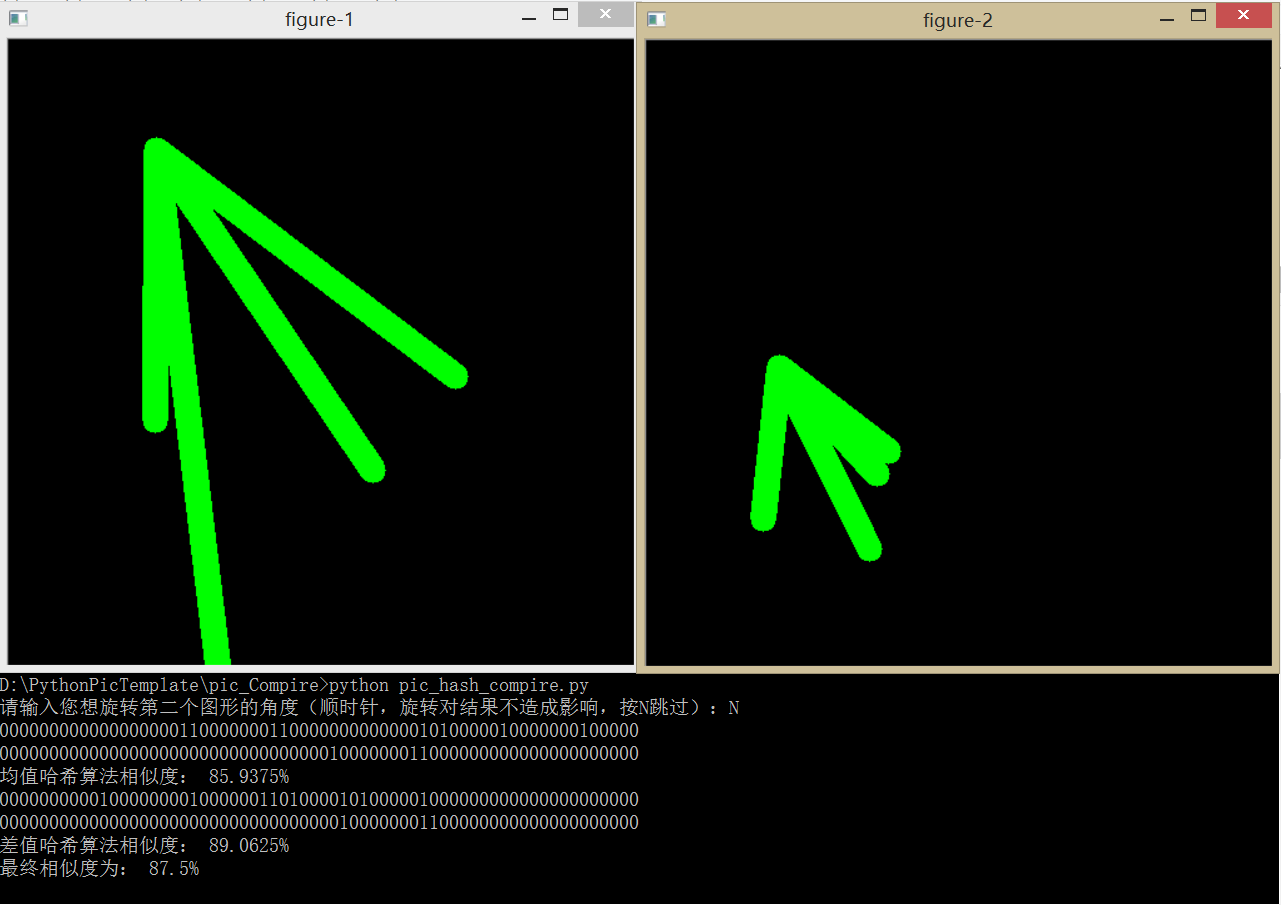
关于简单目标识别与意图分析的机器学习实战研究（神经网络目标识别——基于inception v3模型的分类器）

这一节主要讲怎样判断意图，我们的思路是这样的：首先以意图命名文件夹（这里我们还是以最初定下的attack、combat和march），而后在各文件夹下对应放置各简图，然后用分类器分类并计算这些样本与测试图片间的相似度。

刚开始我们想着直接各图之间作比较，在网上查了一下，有灰度的、cos的、均值hash和感知hash算法的（可以忽视图像的旋转角度）比较，按比较的方向不同，计算出来的相似度也不同，但数值不具有普遍性，同一张图片多次比较得到的结果不同。下面我选用了均值hash和感知hash的平均和来作为相似度，得到的结果分别是91.40625%和87.5%：



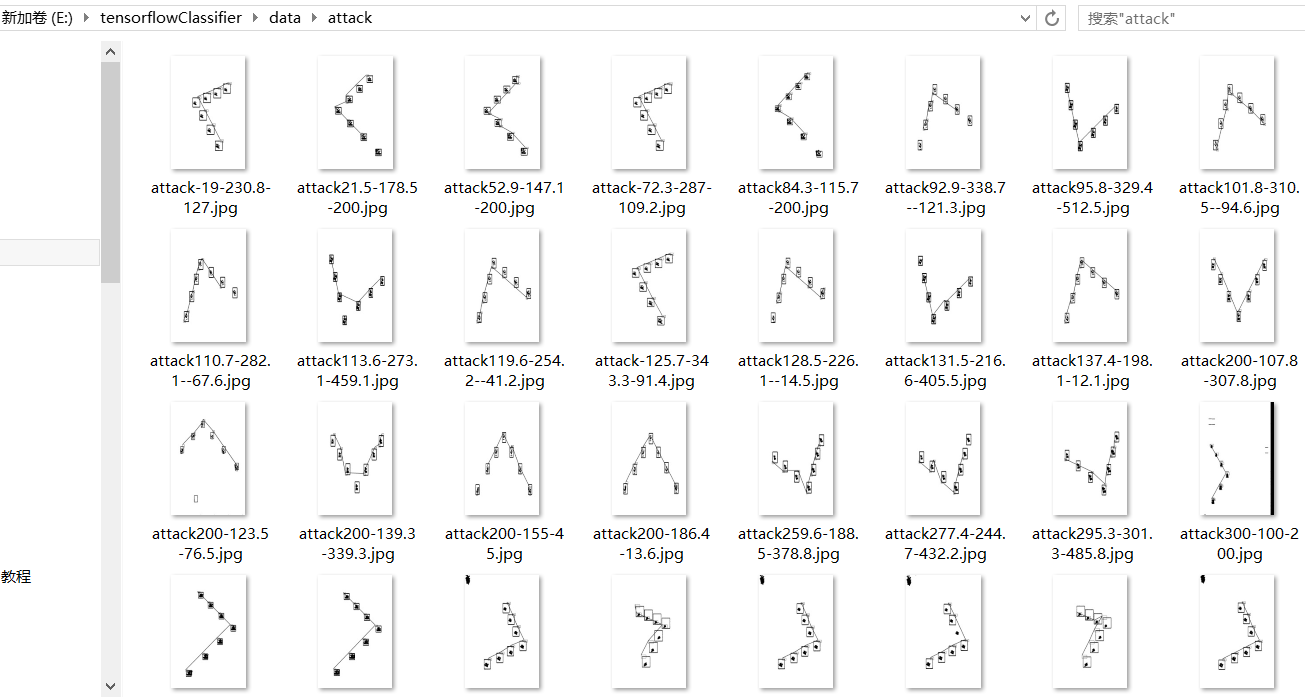


为了得到更局普遍意义上的结果，这里还是选用深度学习框架来对大量训练数据进行分析，用整个训练集来与实验图片比较相似度，得到的结果才能让同学老师满意。下面是使用inception v3模型，将原先的softmax输出抛弃，自定义分类的种类，训练参数，得到自己的分类模型，顺便打印出相似度。

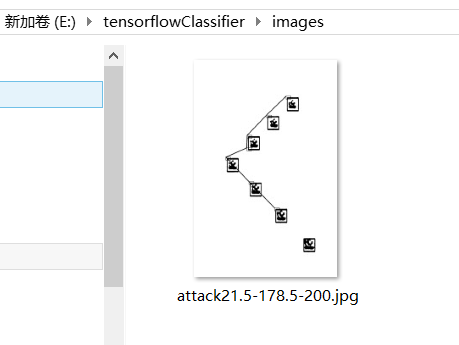
按照这个思路，首先还是制作特定的文件夹存放数据：

bottleneck(空文件夹)  
data(存放要训练的图片数据)  
images(用来测试的图片数据)

将我们之前准备好的attack、combat和march文件夹放在data下面：



image文件夹下放一张准备测试的图片：



而后下载retrain.py 程序，它是用于下载inception v3模型以及训练后面的分类器，我们会在inception模型的基础上进行自己图片分类的代码。附上retrain.py的链接：<https://github.com/tensorflow/hub/blob/master/examples/image_retraining/retrain.py>

下载完成后，运行retrain.py，下载inception v3模型并训练分类器，它的命令格式具体如下：

# 1.模型和样本路径的设置

bottleneck\_dir # 训练数据存放

how\_many\_training\_steps # 训练次数

MODEL\_DIR = 'inception\_model' # inception模型位置

output\_graph # 输出标签的pb文件

output\_labels # 输出检测标签，这里为attackcombatmarch

image\_dir # 测试用图片位置

# 2. 神经网络参数的设置（默认）

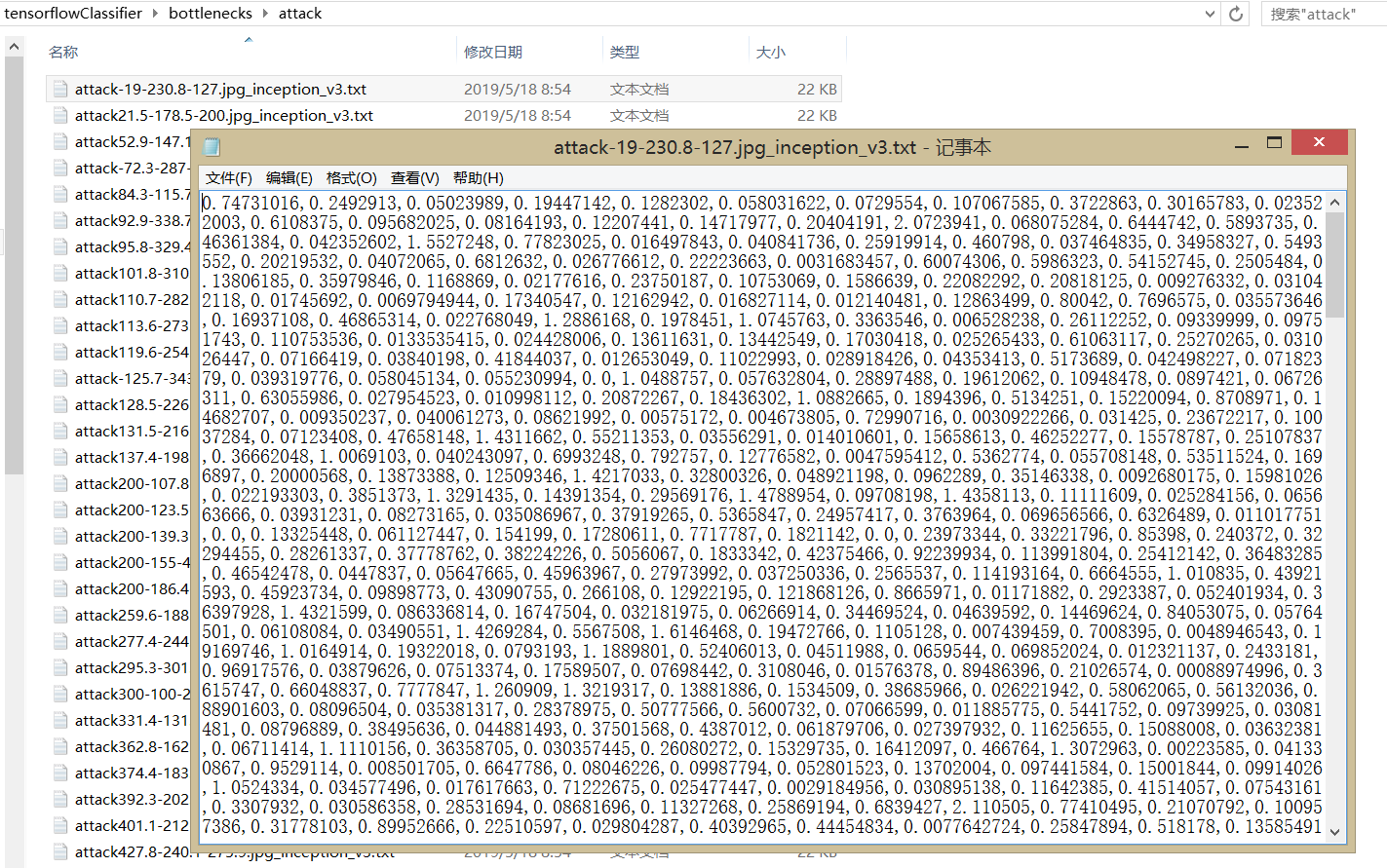
LEARNING\_RATE = 0.01

STEPS = 4000

BATCH = 100

这里我的运行命令如下，运行后会生成output\_graph.pb和output\_labels.txt两个文件，我们的分类效果就是靠这两个文件内的训练数据，bottleneck\_dir文件夹中会对应每张图片生成一个txt文档，计算的是训练参数：





好的，我们的训练很快就完成了，下面就可以进行坦克军团的意图识别了：

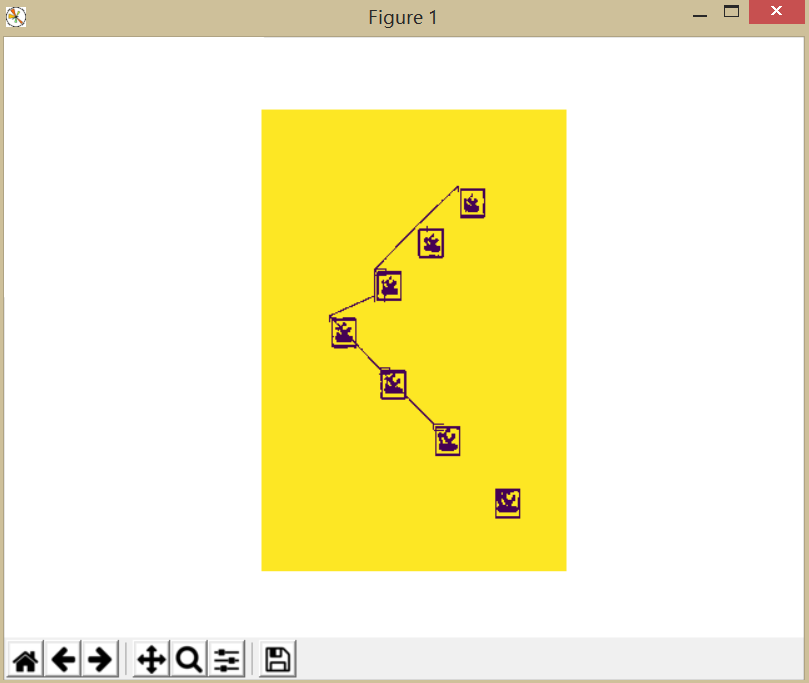
首先写一个数据接收并调用模型的脚本：

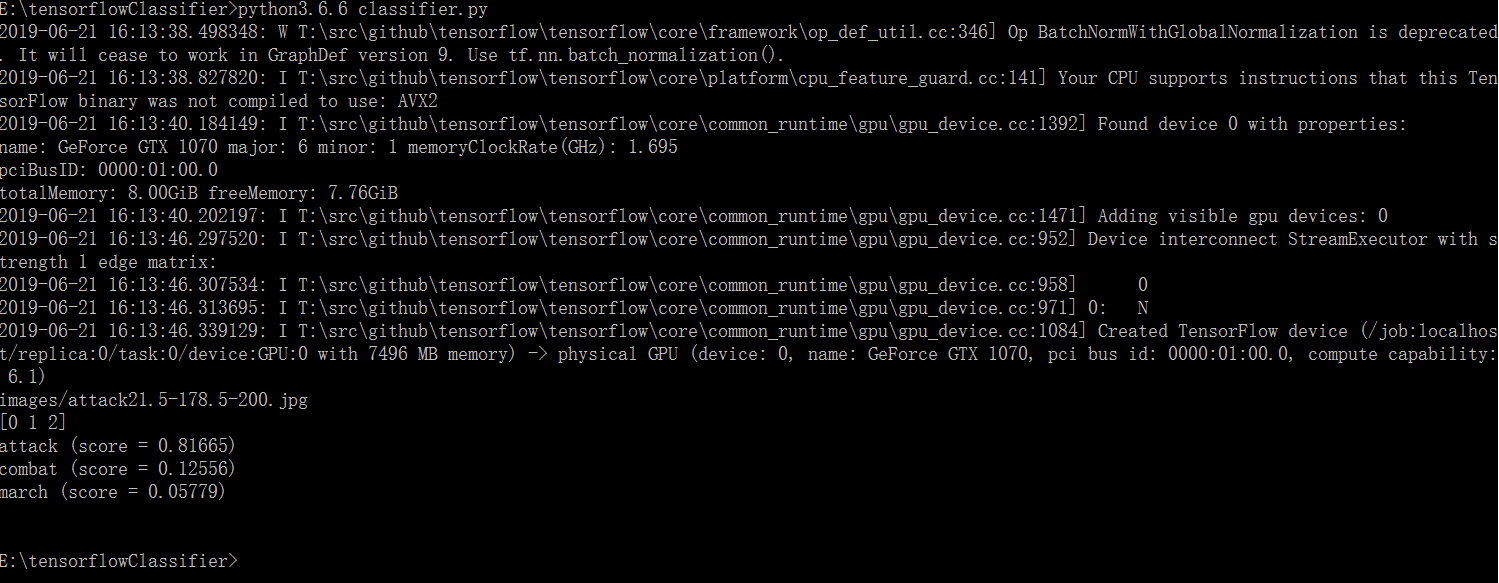
1. **import** tensorflow as tf
2. **import** os
3. **import** numpy as np
4. **import** re
5. **from** PIL **import** Image
6. **import** matplotlib.pyplot as plt
7. lines = tf.gfile.GFile('retrained\_labels.txt').readlines()
8. uid\_to\_human ={}
9. #读取参数中的数据
10. **for** uid,line **in** enumerate(lines):
11. line=line.strip('\n')
12. uid\_to\_human[uid]=line
13. **def** id\_to\_string(node\_id):
14. **if** node\_id **not** **in** uid\_to\_human:
15. **return** ''
16. **return** uid\_to\_human[node\_id]
17. #创建图来存放训练好的模型参数
18. with tf.gfile.FastGFile('retrained\_graph.pb','rb') as f:
19. graph\_def = tf.GraphDef()
20. graph\_def.ParseFromString(f.read())
21. tf.import\_graph\_def(graph\_def,name='')
22. #测试图片分类
23. with tf.Session() as sess:
24. softmax\_tensor = sess.graph.get\_tensor\_by\_name('final\_result:0')
25. #遍历目录
26. **for** root,dirs,files **in** os.walk('images/'):
27. **for** file **in** files:
28. #载入图片
29. image\_data = tf.gfile.FastGFile(os.path.join(root,file),'rb').read()
30. #jpeg格式的图片
31. predictions = sess.run(softmax\_tensor,{'DecodeJpeg/contents:0':image\_data})
32. #结果转为1维度
33. predictions = np.squeeze(predictions)
34. #打印图片信息
35. image\_path = os.path.join(root,file)
36. **print** (image\_path)
37. #显示图片
38. img=Image.open(image\_path)
39. plt.imshow(img)
40. plt.axis("off")
41. plt.show()
42. #排序
43. top\_k = predictions.argsort()[::-1]
44. **print**(top\_k)
45. **for** node\_id **in** top\_k:
46. human\_string =id\_to\_string(node\_id)
47. #置信度
48. score = predictions[node\_id]
49. **print** ('%s (score = %.5f)' % (human\_string, score))
50. **print**()

我们传入一张坦克军团图片（attack意图），然后将它转成连线简图放在images文件夹中，然后运行命令：



分类器输出结果如下：





正如输出打印所示，分类器将三种意图的可能性打印了出来，其中attack占0.81665，combat占0.12556，march占0.05779，效果极其明显。

以上便是对图片内目标进行识别和意图分析的主要过程，同学的汇报也受到了老师的表扬，看来是时候让他准备请客啦。最后一节，将会继续改良脚本，将整个项目的过程关联在一起。