

# 人工智能课程总结

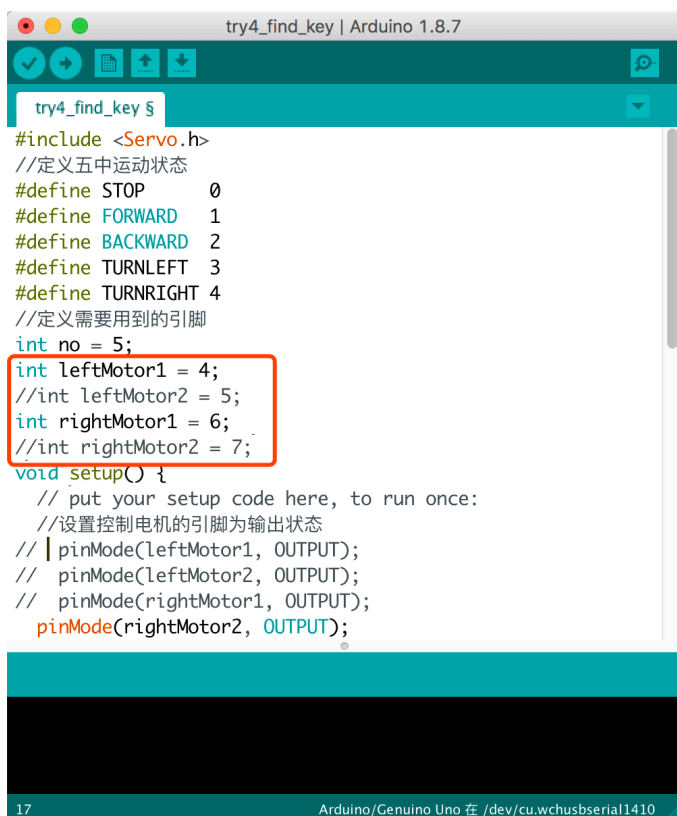
薛畅 2016202117

第2组 组长 薛畅 组员 陈洁婷 姚林丽 苏吉雅 张倩

## 小车项目各阶段进展

### 1. 第一阶段

首先，我们按照老师在ppt上的建议购买了小车主体及其附属零件。虽然价格低廉，但是测试时发现一系列问题：马达端口不定、电线接触不良等。这些问题使我们的进展停滞不前，小车连走都走不了。



```
try4_find_key | Arduino 1.8.7

try4_find_key $
#include <Servo.h>
//定义五中运动状态
#define STOP 0
#define FORWARD 1
#define BACKWARD 2
#define TURNLEFT 3
#define TURNRIGHT 4
//定义需要用到的引脚
int no = 5;
int leftMotor1 = 4;
//int leftMotor2 = 5;
int rightMotor1 = 6;
//int rightMotor2 = 7;
void setup() {
    // put your setup code here, to run once:
    //设置控制电机的引脚为输出状态
    // pinMode(leftMotor1, OUTPUT);
    // pinMode(leftMotor2, OUTPUT);
    // pinMode(rightMotor1, OUTPUT);
    pinMode(rightMotor2, OUTPUT);
}
```

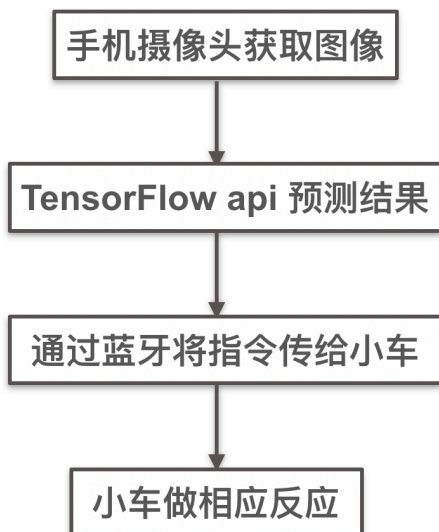
左图即为我们的测试代码。从图中可见，我们在红色方框圈出的区域内尝试更换leftMotor和rightMotor的值。我们试遍了所有端口号的排列组合之后，小车的马达依然没有呈现出任何规律。同时由于手法生疏、缺乏实践经验，小车的多个接线处不时有松动，导致连接非常不稳定。

为了解决这一问题，我们纷纷询问其它小组，试图获得他们用的端口。结果有一个小组也遇到了同样的问题，另一个小组则直接购买了200元的豪华版小车。于是我们决定也购入新版小车，以解决端口的迷之问题。

### 2. 第二阶段

购入新款小车后，我们顺利的完成了小车走直线、避障等基础功能。于是深深地感受到，我们所做的一切都是在硬件的基础上完成的，如果没有性能良好的硬件，我们无论有多么宏伟的设想、远大的抱负，终究巧妇难为无米之炊。当然另一方面是我们确实不够熟悉硬件，导致最简单的硬件问题也无从下手（不过我还是怀疑是板子本身的问题）。豪华版小车不仅硬件豪华，而且还赠送安装教程、示例代码、全天候客服等，这些更帮助我们踏平通往写程序的路。

本阶段的目标是实现基础“智能”，我和张倩负责图像处理部分，其余组员负责语音和蓝牙。我们使用DroidCam软件来实现安卓手机实时传输摄像头视频给windows电脑。然而受限于网速和软件本身，视频传输，就像老师提到过的，有较长的延迟，这将会为之后的图像方面的任务添麻烦。根据老师的建议，TensorFlow提供手机端接口，可以在手机上跑模型做预测。于是我决定一试。与此同时，张倩同学继续用视频传输的方法做图像处理。



左图展示的是app的运行流程，首先从手机摄像头获取照片，之后将该图片作为输入，传给训练好的模型，模型预测出结果，将结果对应的指令（前进/后退等）通过蓝牙发送给小车。

虽然这项任务听起来有些不切实际，但事实上，开发ios app比想象中的要简单，因为：

- 有友好的开发平台xcode

xcode为了满足广大个人开发者的真机测试需求，已经免去了注册开发者账号的要求，使得我能免费在自己的手机上运行自己的app demo来调试代码。

- 有大量tensorflow lite文档

tensorflow lite是一个轻量级的版本，专用于移动平台或嵌入式设备。

在它的主页<https://www.tensorflow.org/lite/overview>上可以看到，它分别支持Android和ios平台，并且有示例代码。如今的手机已经能承担不少计算任务了，可见tensorflow为了充分利用这些计算能力而做出的努力。

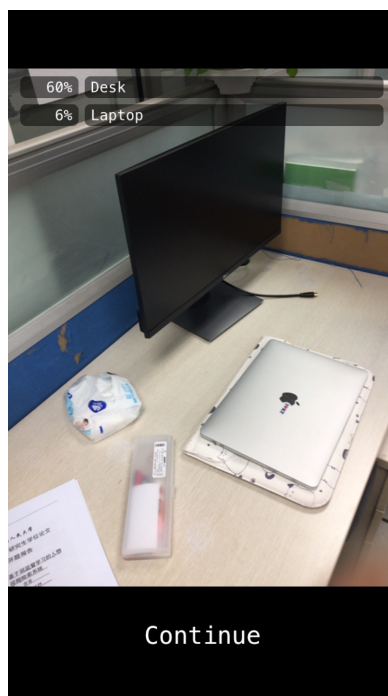
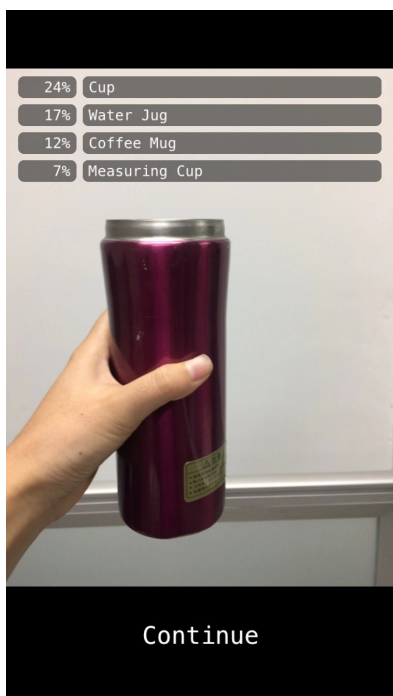
我选择的图像处理任务是物品识别，目标是在手机端打开摄像头后，对准一个物体，能够识别出该物体的种类。首先，需要在电脑上训练出一个物品识别的模型，接着如下表所示，用命令tflite\_convert将其转化成tensorflow lite版本。

```
tflite_convert \
  --graph_def_file=tf_files/retrained_graph.pb \
  --output_file=tf_files/optimized_graph.lite \
  --input_format=TENSORFLOW_GRAPHDEF \
  --output_format=TFLITE \
  --input_shape=1,{IMAGE_SIZE},{IMAGE_SIZE},3 \
  --input_array=input \
  --output_array=final_result \
  --inference_type=FLOAT \
  --input_data_type=FLOAT
```

下一步就要把它移植到我们的app中。我们要安装cocoapods，这是一个十分便捷的工具，尽管它并不那么灵活，不适于太多的个性化设置，但是对于我们的项目来说还是够用的。之后就部署我们的app，设计用户界面等。

最后把它放在手机上测试。在此之前，要确保你有一台iphone手机，系统更新到最新版本，并且手机的apple账户和电脑上的账户相同。之后用数据线连接手机和电脑，在xcode界面最上方的device区域选择你的手机，并在项目的certificate生成自己的证书，在手机的通用设置中同意部署测试应用，这时就可以开始build了。

以下是两张demo app的截屏。左边图中的是我的保温杯，在屏幕顶端分两列显示出置信度和标签。比如第一行代表这张图片中的物品有24%的可能是一个杯子；右边的图片中有许多物品，我希望由此看出该模型对这种多物品场景的识别准确度，由屏幕上方可见，它认为有60%的可能是一个桌子，6%的可能是一个笔记本电脑。事实上，这张图片里两者都出现了，而且桌子确实占了图片比较大的部分，所以该模型准确度还是较为满意的。



做好了物体识别app，之后便是将蓝牙功能嵌入app，这样才能实现对小车的实时控制。然而遗憾的是，ios的蓝牙模块的整合过程困难重重，迟迟不能跟小车连上，最终只好放弃，于是之后的图像识别模块全部是基于DroidCam的视频传输来做的。

从本阶段来看，虽然用手机端跑模型做图像识别的方法是解决视频传输延迟问题的很好的思路，但是ios的蓝牙模块非常复杂，导致最后没能成功用上。所以之后的学弟学妹如果要做app的话，可以考虑做Android版本的tensorflow lite，因为安卓手机对蓝牙的支持程度很高，也能做更多的个性化。由于我的设备都是苹果的，没有安卓设备，所以当时并没有考虑这一点，以后应当注意。

### 3. 第三阶段

在本阶段，我实现了新功能“小车看交警”，即小车能够识别人物的肢体动作，从而判断该左转/右转/前进/后退。该模块的实施流程是首先使用Xception网络和AI challenger人体骨骼点数据集<https://challenger.ai/dataset/keypoint>来做人物的骨骼点识别，接着从手机上获取摄像头的照片，将该照片作为输入，传给训练好的骨骼点模型，得到各骨骼点的坐标信息，最后按照预定义的“交警”动作来判断这些照片中的人物属于哪一个动作，并将相应指令通过蓝牙传给小车。



```
{
  {
    "image_id": "a0f6bdc065a602b7b84a67fb8d14ce403d902e0d",
    "human_annotations": {
      "human1": [178,250,290,522],
      "human2": [293,274,352,473],
      "human3": [315,236,389,495],
      ...
    },
    "keypoint_annotations": {
      "human1": [261, 294, 1, 281, 328, 1, 259, 314, 2,
                213, 295, 1, 208, 346, 1, 192, 335, 1,
                245, 375, 1, 255, 432, 1, 244, 494, 1,
                221, 379, 1, 219, 442, 1, 226, 491, 1,
                226, 256, 1, 231, 284, 1],
      "human2": [313, 301, 1, 305, 337, 1, 321, 345, 1,
                331, 316, 2, 331, 335, 2, 344, 343, 2,
                313, 359, 1, 320, 409, 1, 311, 454, 1,
                327, 356, 2, 330, 409, 1, 324, 446, 1,
                337, 284, 1, 327, 302, 1],
      "human3": [373, 304, 1, 346, 286, 1, 332, 263, 1,
                363, 308, 2, 342, 327, 2, 345, 313, 1,
                370, 385, 2, 368, 423, 2, 370, 466, 2,
                363, 386, 1, 361, 424, 1, 361, 475, 1,
                365, 273, 1, 369, 297, 1],
      ...
    }
  },
  ...
}
```

上图中展示的是数据集的示例。左边的是一张用来做训练的标注照片，其中标注了关键点如头部、肩部等，具体骨骼点及其编号请见下表。右边的图是数据集中的json文件样例，该文件包含每张照片的信息，如有几个人，每个人的骨骼点的坐标等。

1/右肩	2/右肘	3/右腕	4/左肩	5/左肘
6/左腕	7/右髌	8/右膝	9/右踝	10/左髌
11/左膝	12/左踝	13/头顶	14/脖子	

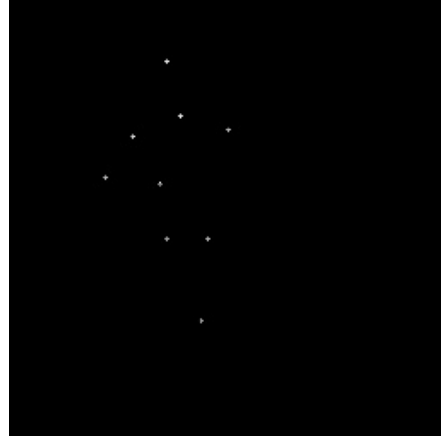
Xception网络主要是采用depthwise separable convolution来替换原来Inception v3中的卷积操作，进一步减少了参数量，同时保证很好的识别效率。我选择其用来识别骨骼点，也是因为易于训练，而且就目前的结果来看，它的输出更为平滑、准确。在keras中已经做好Xception网络的包，我只需根据keras的文档，调用相应函数来建立、训练模型即可。

由于该模型参数较多，所以我先在服务器上用gpu训练，之后将训练好的模型参数保存为h5文件，方便之后做预测的时候直接调用，而且也无需gpu，直接在普通笔记本上就能完成预测。

在测试环节，我尝试了多套“交警”动作来保证模型能准确找到我的肩、肘、腕关节，下半身则不那么重要。在手机拍的图片传到电脑之后，要将图片调整到跟训练数据同样的大小。世间的图片尺寸那么多，我们在训练的时候不可能将各种尺寸的图片都训练一遍，只能训练边长固定的正方形图片。因此预测时传给模型的图片也必须跟训练集图片大小相同。



下图是我设计的第一套交警动作，这一套尽量遵循真实世界的交警动作。左边的是原始图resize成正方形后的样子，该动作希望小车右转；右边的是输出的骨骼点的图，可以看到大部分的骨骼点都能识别出来，但是左手的手腕没能识别，这说明该结果并不理想，因为手腕是判断交警动作的重要因素。



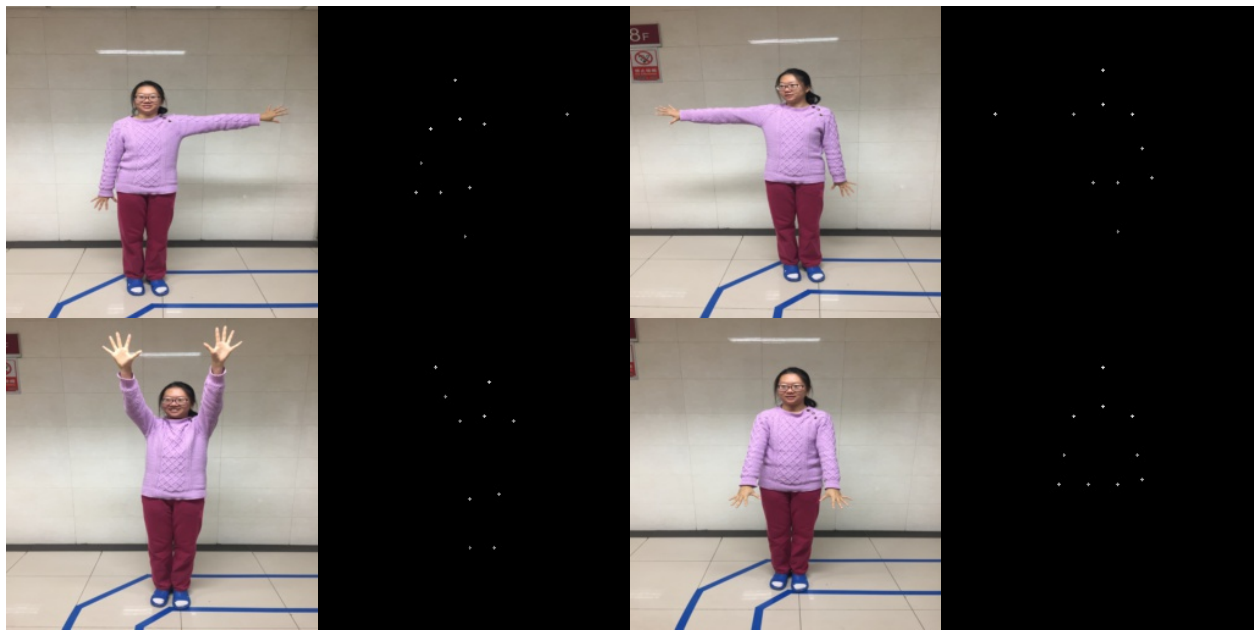
分析看来，上一套动作不理想的原因可能是我撸起了袖子、灯光昏暗、图片不够清晰、墙上有影子。当然这些是外界因素，还有更主要的问题就是我的模型不够好，还有优化空间。但受限于项目期限，没有足够的时间来进一步优化模型，只好改善外部条件。

于是我设计了第二套交警动作。这回我没有遵循真实世界的交警的规范，而是尝试了一种更为直接的方式。此时，它虽然能找到全部的上身节点，但是由于我是侧身面对摄像头，所以并不能确定是左肩还是右肩、左手还是右手，这为之后的工作带来了不稳定性，于是这套动作也不够好。



经过上两套动作的实验，我发现必须让模型尽可能地识别出我的手才行。尽管我把袖子恢复到了正常位置，站在了明亮的位置，也去除了影子的影响，但还是需要尽可能地露出手掌和手背，这样才能更好地让模型识别出来上半身的骨骼点。

针对以上经验教训，第三套动作如下：



可见这套动作能够很好地识别出上半身的全部骨骼点，于是我们最终采取的是这一套作为小车的指令动作。左上角的动作为右转，右上角为左转，左下角为后退，右下角为前进。

识别出骨骼点后，只需根据各个骨骼点的坐标，即可计算得到相应的指令。接下来就需要跟小车相连，而这需要通过windows电脑上的蓝牙，遗憾的是，由于我一直在自己的mac电脑上做实验，临近小组展示时才在小组成员的windows电脑上做衔接工作，结果运行环境始终无法安装好，最后也没能在课堂上展示，非常遗憾。

## 小车项目总结与展望

通过本次项目的实践，我收获了如下经验教训：

- 硬件是软件的前提，好的硬件才能将好的软件实现。
- ios app的蓝牙模块很复杂，之后可以尝试安卓的tensorflow lite版本。
- 配环境往往费时费力，一定要尽早。

除此之外，我学会了：

- 如何从0开始做ios app
- 如何在服务器上用gpu训练xception模型
- 如何调参数、存参数

小车是一个载体，它让我们对人工智能这一领域有了基本的认识。之后我会继续完成交警模块和其它更多有意思的功能。