# 创新实验课 课题总结报告

课题名称： 垃圾分拣小车

小车驾驶与机械臂控制

网络训练与串口通信

课题名称（英文）：Garbage Sorting Car

Car Driving and Mechanical Arm Control

Network Training and Serial Communication

课题组长：  余康佳 罗如瑜

课题组成员： 杨奕冉 赵行越 李韫瑄 许竞舟

报告时间： 2019年 9 月 6 日

目录

[1. 课题研究的选题背景和意义](#_Toc2353_WPSOffice_Level1) [3](#_Toc2353_WPSOffice_Level1)

[2. 课题研究的主要内容](#_Toc13365_WPSOffice_Level1) [3](#_Toc13365_WPSOffice_Level1)

[3. 课题研究的特色](#_Toc24143_WPSOffice_Level1) [3](#_Toc24143_WPSOffice_Level1)

[3.1.DonkeyCar的自动驾驶](#_Toc13365_WPSOffice_Level2) [3](#_Toc13365_WPSOffice_Level2)

[3.2.机械臂](#_Toc24143_WPSOffice_Level2) [4](#_Toc24143_WPSOffice_Level2)

[4. 课题的设计思路、总体架构及主要步骤](#_Toc31904_WPSOffice_Level1) [4](#_Toc31904_WPSOffice_Level1)

[4.1.设计思路：](#_Toc31904_WPSOffice_Level2) [4](#_Toc31904_WPSOffice_Level2)

[4.2.总体架构：](#_Toc20679_WPSOffice_Level2) [5](#_Toc20679_WPSOffice_Level2)

[4.3.主要步骤：](#_Toc30769_WPSOffice_Level2) [6](#_Toc30769_WPSOffice_Level2)

[5. 各模块详细设计、实现及调试说明与分析](#_Toc20679_WPSOffice_Level1) [7](#_Toc20679_WPSOffice_Level1)

[5.1.硬件设计](#_Toc17090_WPSOffice_Level2) [7](#_Toc17090_WPSOffice_Level2)

[5.1.1.垃圾小车部分](#_Toc13365_WPSOffice_Level3) [7](#_Toc13365_WPSOffice_Level3)

[5.1.2机械臂](#_Toc24143_WPSOffice_Level3) [8](#_Toc24143_WPSOffice_Level3)

[5.1.3.树莓派3B](#_Toc31904_WPSOffice_Level3) [10](#_Toc31904_WPSOffice_Level3)

[5.1.4.小车训练过程](#_Toc20679_WPSOffice_Level3) [10](#_Toc20679_WPSOffice_Level3)

[5.2.软件设计及总体流程架构](#_Toc23380_WPSOffice_Level2) [11](#_Toc23380_WPSOffice_Level2)

[5.2.1. donkeycar路线规划](#_Toc30769_WPSOffice_Level3) [11](#_Toc30769_WPSOffice_Level3)

[5.2.2.yolov3垃圾分类图像识别](#_Toc17090_WPSOffice_Level3) [12](#_Toc17090_WPSOffice_Level3)

[5.2.3.树莓派与auduino的通信](#_Toc23380_WPSOffice_Level3) [15](#_Toc23380_WPSOffice_Level3)

[5.2.4.树莓派向服务器发送数据](#_Toc14532_WPSOffice_Level3) [17](#_Toc14532_WPSOffice_Level3)

[5.2.5.服务器向pc发送数据](#_Toc15153_WPSOffice_Level3) [18](#_Toc15153_WPSOffice_Level3)

[5.2.6.pc向树莓派发送数据](#_Toc29474_WPSOffice_Level3) [18](#_Toc29474_WPSOffice_Level3)

[5.2.7.donkeycar 自启（创新）](#_Toc8021_WPSOffice_Level3) [20](#_Toc8021_WPSOffice_Level3)

[5.2.8.donkeycar模型切换](#_Toc10650_WPSOffice_Level3) [20](#_Toc10650_WPSOffice_Level3)

[6. 课题成果及其性能分析](#_Toc30769_WPSOffice_Level1) [20](#_Toc30769_WPSOffice_Level1)

[6.1.已完成的功能：](#_Toc14532_WPSOffice_Level2) [20](#_Toc14532_WPSOffice_Level2)

[(1) 小车开始drive后自动进入模型开始自动驾驶](#_Toc32098_WPSOffice_Level3) [20](#_Toc32098_WPSOffice_Level3)

[(2) 小车未遇到垃圾时循迹](#_Toc3256_WPSOffice_Level3) [20](#_Toc3256_WPSOffice_Level3)

[(3) 小车遇到垃圾后驶向垃圾，并在垃圾面前停下](#_Toc22362_WPSOffice_Level3) [20](#_Toc22362_WPSOffice_Level3)

[(4) 小车摄像机对垃圾拍照，利用socket将图片传输到服务器](#_Toc884_WPSOffice_Level3) [20](#_Toc884_WPSOffice_Level3)

[(5) 服务器进行图像识别，将结果返回给PC](#_Toc14673_WPSOffice_Level3) [20](#_Toc14673_WPSOffice_Level3)

[(6) PC将结果返回给树莓派](#_Toc3755_WPSOffice_Level3) [20](#_Toc3755_WPSOffice_Level3)

[(7) 树莓派根据返回的结果行驶向不同的垃圾回收处](#_Toc31101_WPSOffice_Level3) [20](#_Toc31101_WPSOffice_Level3)

[备注：没有直接用服务器和树莓派进行通信的原因请见后文](#_Toc18777_WPSOffice_Level3) [20](#_Toc18777_WPSOffice_Level3)

[6.2.主要实验数据：](#_Toc15153_WPSOffice_Level2) [21](#_Toc15153_WPSOffice_Level2)

[6.3.实验方法：](#_Toc29474_WPSOffice_Level2) [24](#_Toc29474_WPSOffice_Level2)

[7. 课题成员分工与合作情况说明](#_Toc17090_WPSOffice_Level1) [25](#_Toc17090_WPSOffice_Level1)

[8. 课题所用器材列表及说明](#_Toc23380_WPSOffice_Level1) [25](#_Toc23380_WPSOffice_Level1)

[9. 课题成果链接（包括视频、代码）](#_Toc14532_WPSOffice_Level1) [26](#_Toc14532_WPSOffice_Level1)

[10. 参加本课程的收获、体会及对课程的建议](#_Toc15153_WPSOffice_Level1) [26](#_Toc15153_WPSOffice_Level1)

[10.1.收获](#_Toc8021_WPSOffice_Level2) [26](#_Toc8021_WPSOffice_Level2)

[10.2.体会](#_Toc10650_WPSOffice_Level2) [26](#_Toc10650_WPSOffice_Level2)

[10.3.建议](#_Toc32098_WPSOffice_Level2) [26](#_Toc32098_WPSOffice_Level2)

[11. 参考文献](#_Toc10504_WPSOffice_Level3) [27](#_Toc10504_WPSOffice_Level3)

[附件1：课题执行过程中的问题及解决方法](#_Toc29474_WPSOffice_Level1) [27](#_Toc29474_WPSOffice_Level1)

[附件2：有用文档及链接](#_Toc8021_WPSOffice_Level1) [27](#_Toc8021_WPSOffice_Level1)

# 课题研究的选题背景和意义

近年来，从包围城市的“垃圾山”到海上的“垃圾岛”，全球垃圾问题日益严峻，令人触目惊心。如何处理垃圾、对垃圾进行回收再利用，已成为摆在人类面前的一项迫切任务。境外媒体认为，作为中国人口最多的城市之一，上海此次以如此大的决心和力度开展垃圾分类工作。

今年7月1日起，上海迈入“垃圾分类强制时代”，46个重点城市也在加快垃圾分类的各项环节建设。上海的这一举措，带来了一股垃圾分类的热潮。然而我国大部分市民尚未养成垃圾分类的一贯，对于这一“突如其来”的政策，人们很难在短时间内熟悉各种垃圾所属的类型，这成为了广大市民普遍遇到的烦恼。为迎合这一社会发展趋势，同时解决人们垃圾分类的困难，我们预想设计一款垃圾分类拾取小车，它能够在指定场景（空旷室内）下自动寻找垃圾，利用卷积神经网络机器学习识别出垃圾的类型，并将垃圾带到指定的垃圾站投入对应类型的垃圾桶。这样便可将垃圾清理、垃圾分类两步工作自动化、一体化，避免了人工处理的困难与麻烦。

# 课题研究的主要内容

本项目主要内容是垃圾分拣小车，主要应用场景为应用于在较为空旷的场地，例如运动后的空旷室内体育馆。垃圾分拣小车将通过摄像头对路面垃圾情况进行监测，检测到指定类型的垃圾（易拉罐）等靠近，并在接近到一定距离时，垃圾分拣小车停止，摄像头进行拍照，并通过所拍照片判断所检垃圾的类型，用机械臂将垃圾铲起，不同的类型的垃圾放到不同的垃圾分类地点中，到达垃圾类型指定地点后垃圾分拣小车停下，并通过机械臂将所检的垃圾放下。

# 课题研究的特色

**3.1.DonkeyCar的自动驾驶**

DonkeyCar自动驾驶是通过应用卷积神经网络进行深度学习而实现的。通过人工操作时记录面对某种状况（具体体现为摄像头所拍摄的照片）时DonkeyCar的angle和throttle等参数，用大量的数据集进行训练得到模型以实现自动驾驶。

具体的实现方法是使用树莓派与DonkeyCar进行串口通信以控制DonkeyCar的电机和舵机，并在树莓派上连接一摄像头用于拍照记录。本项目中的路径规划利用由Python编写的Keras库作为Tensorflow的接口，进行深度学习模型的设计、评估、调试。图像分类应用较新的网络yolo v3，因为考虑到训练时的高效。

小车能够实现全程自动操作的功能，包括智能巡航寻找垃圾、自动驾驶接近垃圾并拾取、智能识别垃圾种类、自动将垃圾带到指定垃圾站并放在对应垃圾桶中，整个过程十分流畅。

**3.2.机械臂**

机械臂通过arduino开发板进行编程实现操控，与树莓派通过USB的连接进行串口通信。在DonkeyCar运行到throttle为0时，机械臂开始操作。使用机械臂进行垃圾清扫，提高了垃圾清理的精度和灵活性，可准确拾取摄像头识别出来的垃圾。

项目出发点很好地迎合了当下广受关注的垃圾分类问题，顺应绿色发展、保护环境的趋势，将垃圾拾取与垃圾分类自动化、一体化，并将机器学习中卷积神经网络应用到自动驾驶、图像识别等技术之上，在清理垃圾的同时解决了人工对垃圾进行分类的困难。

# 课题的设计思路、总体架构及主要步骤

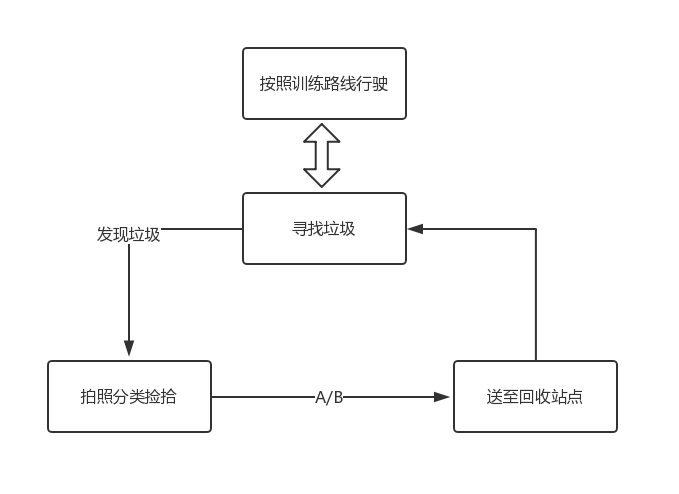
**4.1.设计思路：**

本项目一共分为三个模块，分别为小车寻找垃圾的路径，拍照垃圾分类捡拾垃圾，垃圾送至站点回收。

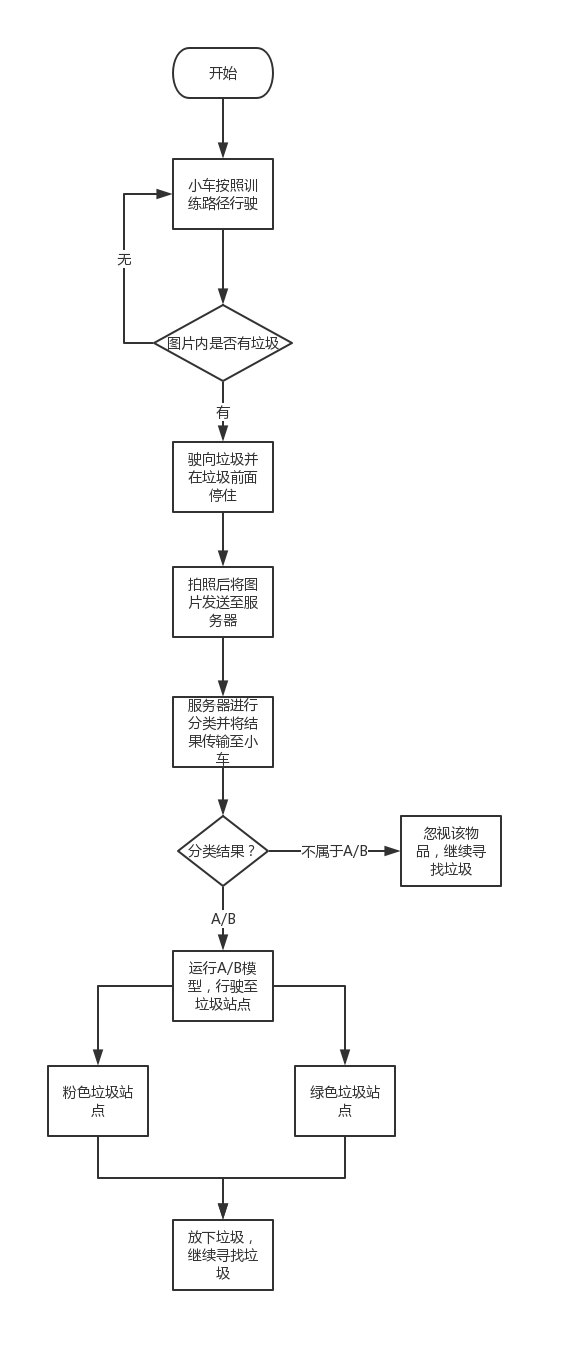
第一模块：运行开始时，小车根据摄像头的拍摄内容内是否存在垃圾来选择行为。若存在，则小车驶向垃圾，在垃圾面前停住，并进入垃圾分类识别的模块；若视野内不存在垃圾，则按照训练路线行驶，直至图片内出现垃圾后驶向垃圾。

第二模块：当小车在垃圾前停止时，可检测到马力值为0，凭借该信号，摄像头拍摄一张垃圾的近距离高清静止照片，并输出“拍摄成功”，之后将该照片发送至服务器进行分类（一共AB两类），分类结束后将分类结果发回至小车。若该垃圾不属于A/B类，则不对垃圾进行拾取，继续行驶小车；若接收到A/B分类信息，则启动机械臂。机械臂接收到开始信号后，按照既定输入程序运行动作，垃圾将被收入至框中，该模块执行完毕后进入垃圾送至站点回收模块。

第三模块：根据A/B信息，启动不同的A/B模型，小车将回归第一模块中的训练路径行驶，至视野中出现垃圾站点的标识，驶向目标垃圾站点且靠近后停止。A/B模型的区别仅为在不同颜色的站点前停止。停止后机械臂打开，小车退出，垃圾被留在垃圾站点，之后小车继续寻找垃圾。



**4.2.总体架构：**



**4.3.主要步骤：**

目前的应用场景为无人平坦的室内，按照两类垃圾进行训练。小车连接热点后，电脑Xshell 输入启动小车，小车开始按照既定轨迹行驶，摄像头拍照。小车根据摄像头的拍摄内容内是否存在垃圾来选择行为。若存在，则小车驶向垃圾，在垃圾面前停住，并进入垃圾分类识别的模块；若视野内不存在垃圾，则按照训练路线行驶，直至图片内出现垃圾后驶向垃圾。

小车驶向垃圾直到距离垃圾很近时（机械臂刚好将垃圾捕捉入垃圾筐中的距离）停下。当小车在垃圾前停止时，可检测到马力值为0，凭借该信号，摄像头拍摄一张垃圾的近距离高清静止照片，并输出“拍摄成功”，之后将该照片发送至服务器进行分类（一共AB两类），分类结束后将分类结果发回至小车。若该垃圾不属于A/B类，则不对垃圾进行拾取，继续行驶小车；若接收到A/B分类信息，则启动机械臂。机械臂接收到开始信号后，按照既定输入程序运行动作，垃圾将被收入至框中，该模块执行完毕后进入垃圾送至站点回收模块。

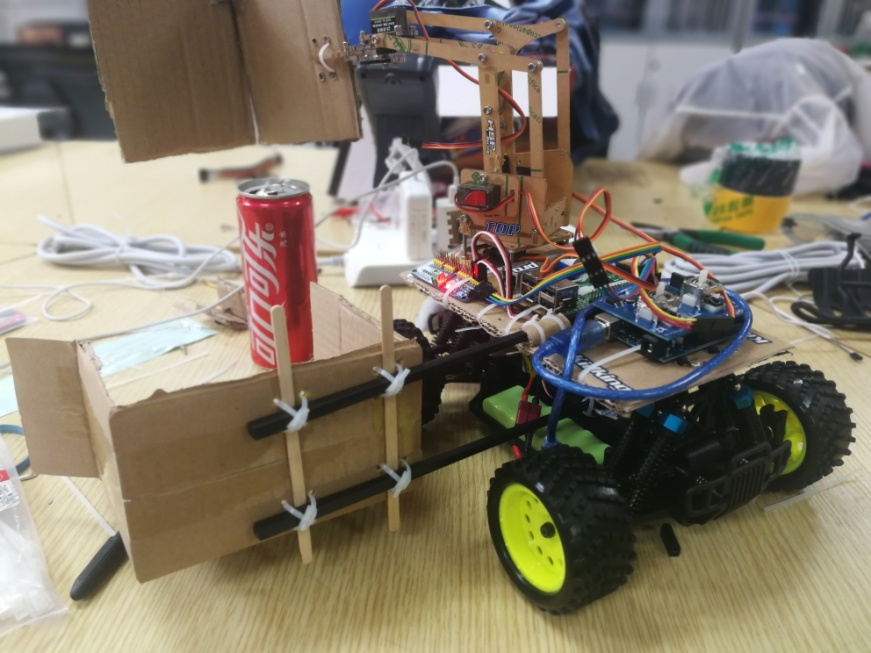
依据收到的的分类信息，启动不同的A/B模型，小车将回归第一模块中的训练路径行驶，至视野中出现垃圾站点的标识，驶向目标垃圾站点且靠近后停止。A/B模型的区别仅为在不同颜色的站点前停止。停止后机械臂打开，小车退出，垃圾被留在垃圾站点，之后小车继续寻找垃圾。

识别垃圾类别摄像头进行拍照一张，将图片发送至服务器进行分类，服务器将分类结果不停拍照，当识别出该垃圾为未某一类垃圾的可能性大于0.8时停止，将类别信息保存，且向机械臂发送信息。机械臂收到信息后开始按照已训练好的模型动作，将垃圾捕捉入小车中。判断垃圾捕捉成功后小车开始寻找垃圾站点，若未捕捉成功则继续重复捕捉动作至捕捉成功。此时摄像头拍照，若图中出现站点信息，则向站点方向驶去，否则在原地以最小半径旋转至发现垃圾站点。到达垃圾站点后，根据垃圾类别识别结果，将垃圾放入其所属类别站点中，一次垃圾的分类捕捉与回收即完成。当Xshell输入停止命令时停止。

# 各模块详细设计、实现及调试说明与分析

**5.1.硬件设计**

5.1.1.垃圾小车部分





小车基座为基础电动车底架，其上通过扎带以及胶带固定了树莓派3B，arudunio，机械臂相关arm-joystick板子,组装机械臂。

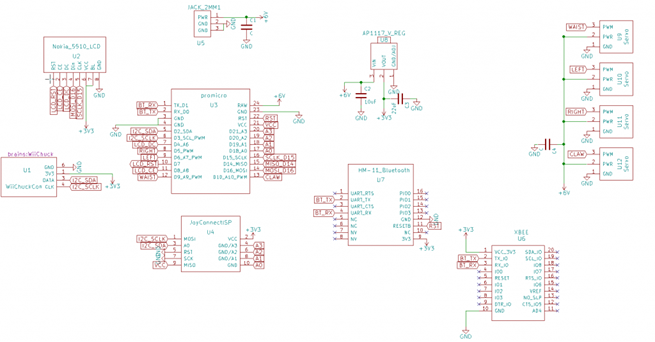
5.1.2机械臂

详细设计：

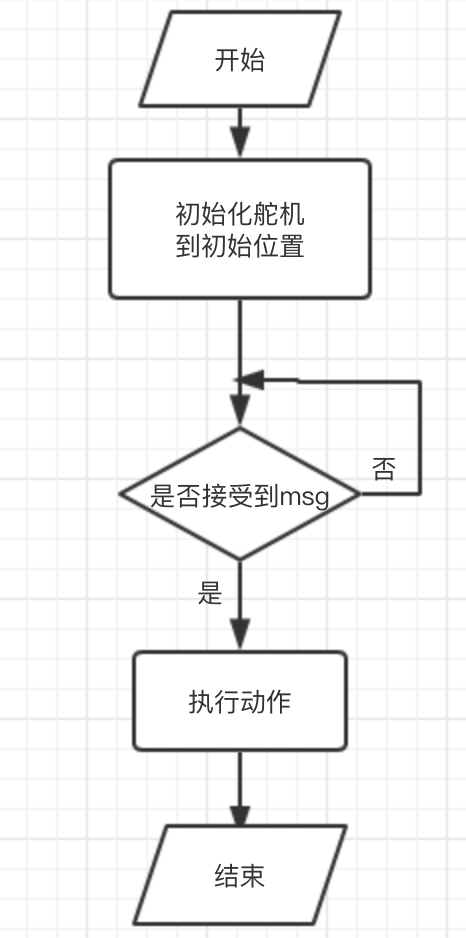
组装机械臂，其中包括为了执行动作涉及到的4个舵机，以及亚克力板子。arudunio，机械臂相关arm-joystick板子，通过arudunio控制机械臂四个舵机的转向（角度），从而达到完成“收垃圾”，“倒垃圾”的操作。arm\_joystick还可以用于摇杆实时操作机械臂，由于项目中并未使用手动遥杆，在此不多赘述。

根据机械臂官方给出的文档，我们可以得到舵机对应的引脚（如下图）和部分控制舵机角度的函数。要确定机械臂的动作，得首先确定机械臂几个关键位置的对应舵机的位置。经过层层调试，确定的几个动作对应的舵机的值之后，思考如何使得机械臂在正确的时间点执行一次指令。这时我们使用了Arduino与树莓派的串口通信，在接受到树莓派发出的特定的信号过后，机械臂执行刚刚预置好的指令。执行完之后机械臂保持最后一个姿势不动。

垃圾揽收部分，均为手工制作，机械臂上安装类似直角“爪子”以方便对垃圾进行收揽或放下 ，垃圾收集筐通过固定杆以及扎带等进行固定，并在对应对边采取均重措施以防止小车因为一边负重造成的倾斜。机械臂加爪子与垃圾筐能对垃圾进行控制。



具体的代码流程为：



5.1.3.树莓派3B

用于路径规划、控制信号发出端：控制相机对垃圾进行拍照、对机械臂的不同动作的触发控制、与服务器与串口通信等。

5.1.4.小车训练过程

训练小车是我们组投入时间精力很大的部分，在训练过程中遇到了层出不穷的问题，我们在训练过程中不断分析可能出现问题的原因，并针对问题提出各种可能的解决方案，以实现最好的训练效果。

无线控制树莓派：训练之初会出现树莓派无法连接热点的问题，经分析发现树莓派的sd卡中保存的wifi配置文件会不停的消失，但在与助教交流之后发现这个是插拔sd卡可能会出现的正常现象，我们又通过更改电脑热点的适配器选项，并且通过限制热点的网络频带来提高连接热点的速度和稳定性。

小车马力值调试：小车的马力控制机制也给训练带来很大的困扰，小车的马力输出数值会随电池电量的不同而会有不同的速度，因此在训练过程中需要不断地调整小车马力的预设值来实现小车的平稳行驶。而且在自动驾驶模式下，马力输出显得更不稳定，在短时间电量并没有明显变化的情况下小车的行驶速度也会有着较大幅度的变动，因此对模型测试产生很大的影响，我们在训练过程中需要不停的通过调参来强制对马力值进行微调才能使小车平稳行驶。

小车摄像头调试：小车的摄像头也是我们投入大量精力调整的部分，最开始我们为了追求返回照片的质量，将摄像头分辨率调整到320\*270甚至1920\*1080，在训练之初并没有发现什么问题，但是模型训练效果总是不理想，并且在运行一段时间之后树莓派的芯片会非常的烫。在询问助教了解情况之后发觉可能使图片质量太高，在执行自动驾驶模式的时候树莓派处理速度不够，导致不能对场景进行快速准确的识别，具体体现为在看到转弯标识之后有一点转弯的反应但是不能很好的将转弯进行到底。在将摄像头分辨率下调至160\*120之后，可以观察到在执行自动驾驶模式的时候屏幕输出马力值的速率明显加快，证明树莓派运行速率上升，相应的在遇到转弯等明显标识的时候做出的反应也会更加的灵敏。

模型融合与转换（难点）：最难解决的还是不同模型数据组合之后效果不好的问题。我们的小车在不同阶段需要加载不同的自动驾驶模型，不同阶段的任务完全不同，所以在训练阶段我们需要分任务训练不同的模型。这些模型在单独执行自动驾驶模式的时候基本都能实现预想的效果，但是将数据合起来训练之后生成的总模型效果就会变差，各阶段的任务都不能很好的实现。在与助教和老师交流之后，我们发现不同阶段数据合起来时的数据比例也很重要，并且在总模型效果不好的路段要进行强化训练，增加这部分路段的数据收集，加强训练。在这个思路下我们重新控制各阶段的数据比例进行训练，并且仔细观察小车行驶情况，在易出问题的路段进行强化训练，最终能够训练得到一个能较为满意的执行各阶段任务的总模型。

**5.2.软件设计及总体流程架构**

5.2.1. donkeycar路线规划

5.2.1.1.donkeycar模型简述

DonkeyCar小车是基于树莓派和Python，利用Keras深度学习框架实现自动驾驶小车。

本项目需要三套模型。

模型一：如果小车视野中没有垃圾，则按特定轨迹巡逻；小车视野中出现垃圾，小车走向垃圾，并停于距垃圾一段距离前。

模型二：若捡到垃圾为第一类垃圾，则待机械臂将垃圾收入后，将垃圾送入第一类垃圾对应垃圾存放区域（第一类垃圾桶）。

模型三：若捡到垃圾为第二类垃圾，则待机械臂将垃圾收入后，将垃圾送入第二类垃圾对应垃圾存放区域（第一类垃圾桶）。

其中某垃圾为第一类或第二类垃圾区分，由图像分类识别返回的种类决定。

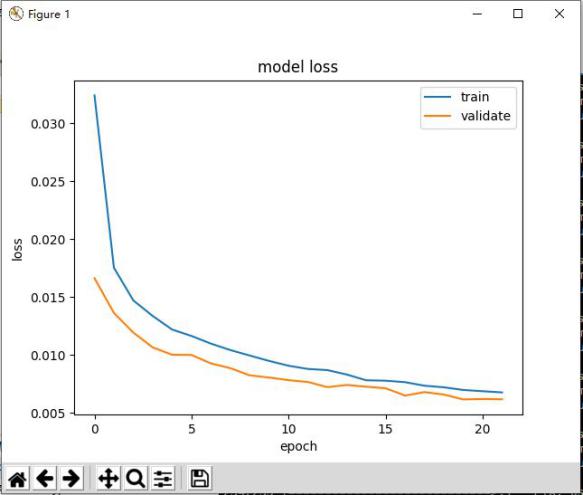
由于本项目需要实现全自动，所以需要能开机自启，并通过摄像头识别结果自动切换不同的模型。

5.2.1.2. donkeycar模型切换（创新）

由于本项目需要实现全自动，所以需要能开机自启，并通过摄像头识别结果自动切换不同的模型。

模型切换基于donkeycar开机自启，需要加全局变量对小车各个部分进行不同程度的时延控制，我们组第一组编出了开机自启，全自动加载模型的方法，并在助教提醒下也完成了小车模型切换的难点。具体切换方式将在后面说明。

5.2.1.3. donkeycar训练过程



由图可知，loss曲线经历了大约23次epoach收敛并停止了训练，不能过度训练以防止模型过拟合。

5.2.1.4. donkeycar 数据采集

我们在训练小车实现所需功能上花费了极多的精力与时间，期间经历了上一届的大车动机不稳定出问题，通过不断试验改变图像分辨率以获得最高效分辨率，早上晚上环境的变化，楼道环境干扰等。

5.2.2.yolov3垃圾分类图像识别

5.2.2.1.使用YOLOv3架构区分垃圾的种类

本项目中的图像分类框架为darknet上最高效的yolo v3，YOLO3采用了称之为Darknet-53的网络结构（含有53个卷积层），它借鉴了残差网络Residual network的做法，在一些层之间设置了快捷链路，采用了3个不同尺度的特征图来进行对象检测，采用K-means聚类得到先验框的尺寸，为每种下采样尺度设定3种先验框，总共聚类出9种尺寸的先验框，在精确度相当的情况下，YOLOv3的速度是其它模型的3、4倍。考虑到训练数据较大，采用了GPU服务器并自己配置环境进行训练。

具体实现方法：

① 收集图片，形成训练集；

② 训练模型，生成weight；

③ 接受树莓派发来的图片并返回该垃圾类别（使用socket通信）。

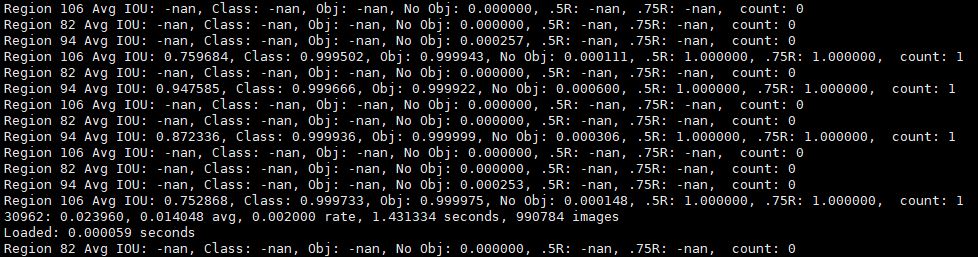
5.2.2.2.图像分类数据集处理

labellmg 用于人工处理收集到的数据集，首先生成对应图片txt格式 （yolo要求标准格式，xml也可），而后通过编写代码自动划分数据集以及配置数据集接口进行训练。

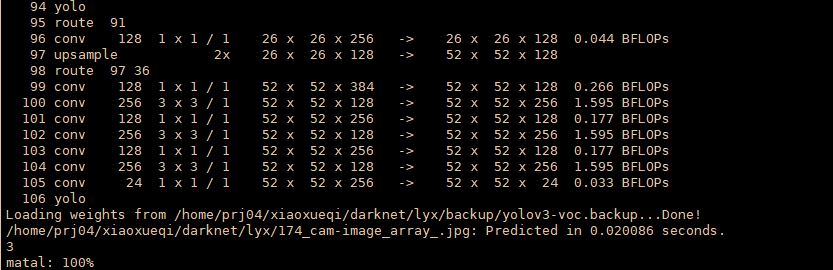
后期评估网络mAP时，需要将数据集转换为voc格式。

5.2.2.3.示例训练结果

训练过程图

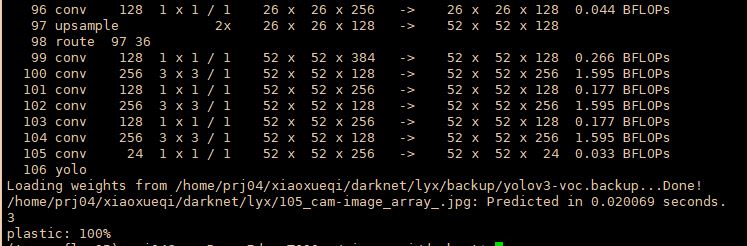


识别过程如下所示：

示例一：

由上图可知 ，174\_cam-image\_array\_.jpg图像识别为metal，预测时延为0.020086s。

示例二：



由上图可知 ，105\_cam-image\_array\_.jpg图像识别为metal，预测时延为0.020069s。

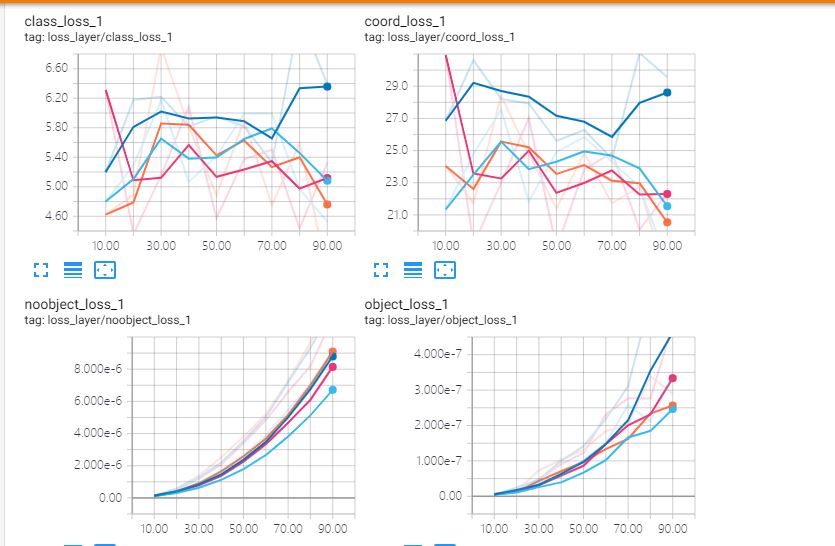
示例三：预测图片如下



5.2.2.2.简述使用过的网络及其优缺点

为了进一步优化垃圾识别分类，我们尝试过四个最基础的神经网络，即inception\_net，alexnet，restnet，mobilenet，vgg网络进行测试，虽然对设备要求低，但是考虑到后期进一步的优化潜力以及性价比，比较基础的网络有些低识别率而且再拓展性比较差，所以选择yolov1-v3。

由于yolov1的坐标的不准确性，甚至效率并不高于基础网络；yolov2只是增加了些训练方法，为跨数据集训练提供了思路；yolov3相比于就比较适合，而且由于darknet官网的进一步完善更新，其中的lite版本很适合高速轻量级网络。

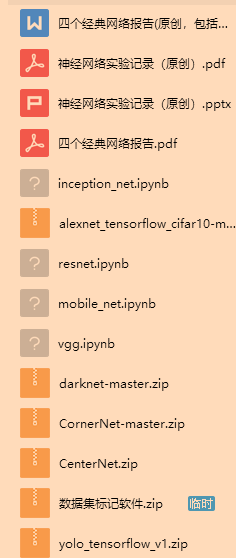


tensorboard展示训练过程 mAP约为89（yolo训练了2个小时）

其后我们还尝试了最新的centernet以及cornernet的2019年四五月份新出的onestage网络，与宣传的高效对应，这些网络对硬件要求比较高，运行时占用显存比较严重，而且对精准度提升不大，考虑到因为这个网络主要用于姿态检测，所以可能在图像分类上并没有很高效的表现。

5.2.2.3.资源共享及其相关

考虑到有别的组也做了图像分类识别，我们组将所有我们实验过的网络（inception\_net,alexnet,vgg\_net,moble\_net,yolov1-v3,centernet,cornernet）（绝对都可以百分之百训练以及运行），以及实验过程中的错误笔记总结等发到了群里。

部分文件资源共享展示

5.2.3.树莓派与auduino的通信

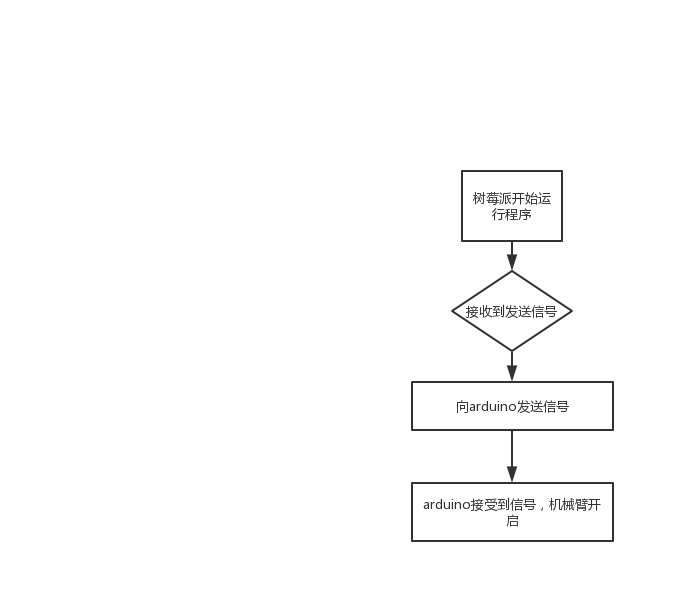
详细设计：

利用USB进行串口通信，先给树莓派发送指令，树莓派接收到指令后通过USB给arduino发送指令，arduino开始控制机械臂。

电路图：

Arduino

代码流程图：



实现：

a、树莓派部分

具体实现方法：

1. pip install pyserial ；pyserial是树莓派对串口的操作库
2. 在树莓派终端输入ls /dev/tty\*查看端口，发现端口是/dev/ttyUSB0，并修改波特 率为9600
3. 在Arduino中上传代码
4. 在树莓派终端运行并观察现象

部分代码：

import serial

# serial是树莓派对串口的操作库

ser = serial.Serial('/dev/ttyUSB0',9600,timeout=1)

# /dev/ttyUSB0 是串口代号，可以在树莓派上 ls /dev/ttyACM\* 看到连接，也可以在arduino IDE的 “Serial Port”上看到

# 9600是波特率，即串行每秒传递数据的频率，需要和 Arduino代码中串口的 波特率保持一致

if ans == str.encode(a) :

ser.write(str.encode("op"))

elif ans == str.encode(b) :

ser.write(str.encode("cl"))

#ser.write就是树莓派向串口发送数据，在这里是向连接着的arduino发送数据 "op"，"op"在arduino里有定义，即 open

b、Arduino部分

具体实现代码（部分）：

if(Serial.available()){     //即能接收到串口传来的信息  
        String ans = Serial.readString();   //读取string类型的信息  
        if(a == ans){ }//a是预先设置好的指令

调试说明与分析：

经过调试，发现不同的端口有不同的名字，而我们所使用的端口是/dev/ttyUSB0，而且插拔会导致端口名称发生变化，因此最好不要多次插拔。

5.2.4.树莓派向服务器发送数据

详细设计：

在小车拍照成功后储存为test0.jpg，使用socket通信将图片数据发送至服务器。

调试说明与分析：

###RespToSer.py

from socket import \*

import threading

s=socket()

s.connect(('10.110.210.24', 2578))

#连接服务器闲置端口

f=open(r'/home/pi/mycar/test0.jpg', 'rb')

#按照二进制位传输图片数据

while True:

data=f.read(1024)

if not data:

break

s.send(data)

#发送数据

f.close()

s.close()

###SerAccResp.py

from socket import \*

s=socket()

s.bind(('10.110.210.24',2578))

s.listen(5)

#实时监听该端口，随时准备接受图片数据

c,addr=s.accept()

print('connnect from ',addr)

f=open(r'/home/prj04/xiaoxueqi/darknet/test0.jpg','wb')

while True:

data=c.recv(1024)

#print(data)

if not data:

break

f.write(data)

#接收到图片数据后保存至服务器

f.close()

c.close()

s.close()

5.2.5.服务器向pc发送数据

详细设计：

当服务器得出图片分类结果后，储存在一个txt文件中，服务器运行一个将类别转化为数字的py文件，降低设备传输之间的时间损耗；之后更改WinSCP设置使其每间隔1s同步更新本地txt文件，实时将分类结果保存至pc端。

5.2.6.pc向树莓派发送数据

详细设计：

pc将分类结果传输至树莓派，树莓派依据收到的结果启动不同的模型。

调试说明与分析：

###PcToRasp.py

import time

from socket import\*

def read():

with open(r'C:\Users\liyunxuan\Desktop\test.txt','r',encoding='utf-8') as f:

lines=[] # 创建了一个空列表，里面没有元素

for line in f.readlines():

if line!='\n':

lines.append(line.strip('\n'))

print(lines)

return lines

def commun(line):

s=socket()

#ip地址写树莓派的

s.connect(('10.128.205.61',6665))

s.send((line).encode("utf-8"))

print(line)

s.close()

if \_\_name\_\_ == '\_\_main\_\_':

while True:

time.sleep(0.5)

read()

if read()[0]!=str(0):

commun(read()[0])

break

###RsapAccPc.py

import socket

import os

import sys

def socket\_service\_data():

try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

#s.bind(('127.0.0.1', 6666))

#树莓派ip

s.bind(('10.128.205.61', 6665))

s.listen(10)

except socket.error as msg:

print(msg)

sys.exit(1)

print("Wait for Connection..................")

ff=open(r'/home/pi/mycar/sort.txt','wb')

while True:

sock, addr = s.accept()

buf = sock.recv(1024)

if not buf:

break

buf = buf.decode()

ff.write(buf)

#将传输内容存储至sort.txt

print("The data from " + str(addr[0]) + " is " + str(buf))

print("Successfully")

# return buf

# sock.close()

ff.close()

if \_\_name\_\_ == '\_\_main\_\_':

socket\_service\_data()

5.2.7.donkeycar 自启（创新）

基于考虑到需要全自动的要求，需要不经过点击网页中“start vehicle”以及选择“local”，上电后就自动开始自动驾驶。

询问了之前并未有组实现过完全自启，我们组共同努力下通过对不同的模块，主要通过在manage.py等中全局变量，增加不同的延时以控制模型加载与图形输入的时间先后，以防出现“Nontype”等错误。

我们组在实现开机自启后也对代码进行了共享并及时与老师助教沟通。

5.2.8.donkeycar模型切换

通过对代码以及命令行中传参方式的学习，可以通过其中的self.on直接控制模型的开关，判断马力值连续小于某值时（因为小车会朝垃圾走去并停下）将第一个模式关闭，因为开机自启已实现，识别出垃圾后开机，即可通过不同模型的传参加载出不同垃圾对应的模型，从而走到对应不同垃圾地点。

因为donkeycar很多功能都依赖于网页以及自身算法程序比较密封，而且横跨了很多不同语言，刚开始浪费了很多时间用于找寻合适的数据接口，主要是找模式传递接口以及（通过模型判断出的）马力值输入接口。

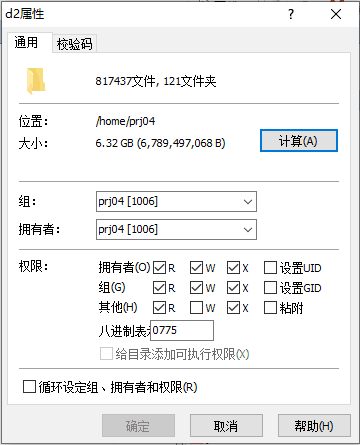
# 课题成果及其性能分析

**6.1.已完成的功能：**

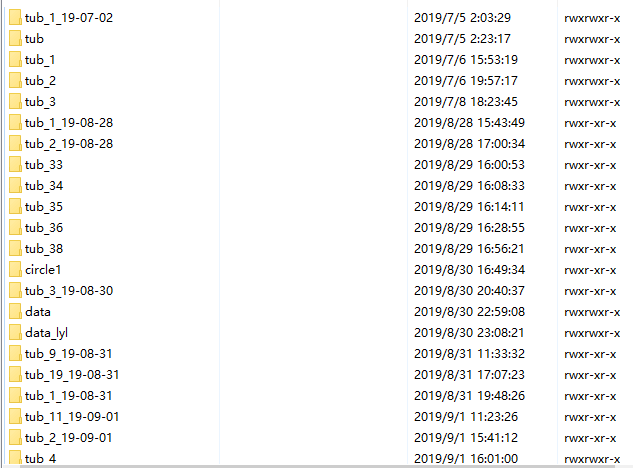
1. 小车开始drive后自动进入模型开始自动驾驶
2. 小车未遇到垃圾时循迹
3. 小车遇到垃圾后驶向垃圾，并在垃圾面前停下
4. 小车摄像机对垃圾拍照，利用socket将图片传输到服务器
5. 服务器进行图像识别，将结果返回给PC
6. PC将结果返回给树莓派
7. 树莓派根据返回的结果行驶向不同的垃圾回收处

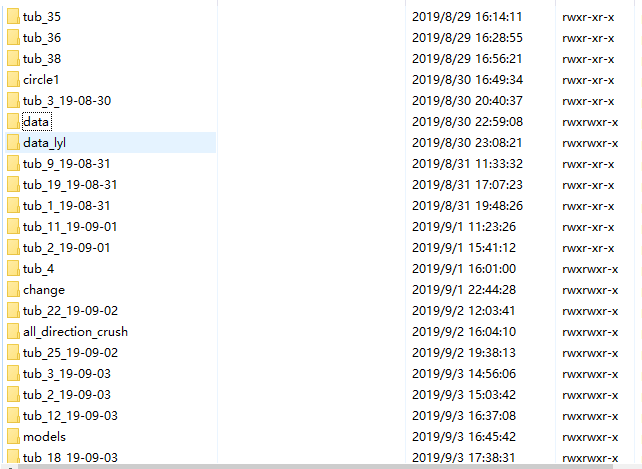
备注：没有直接用服务器和树莓派进行通信的原因请见后文

**6.2.主要实验数据：**

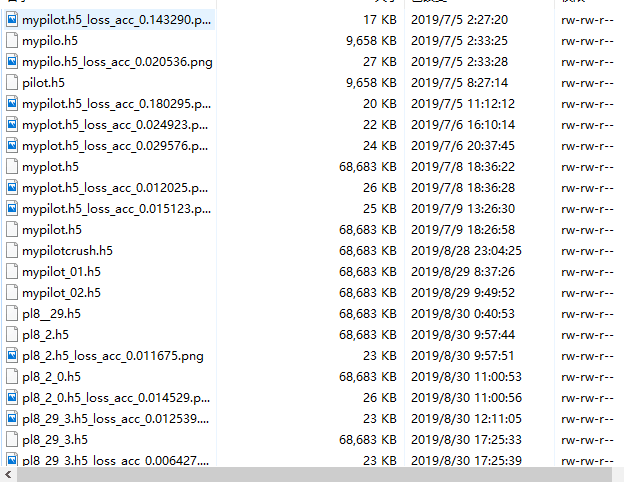
本课题主要包括两部分：训练小车、树莓派与服务器与PC与arduino的通信，d2包含了部分使用的数据

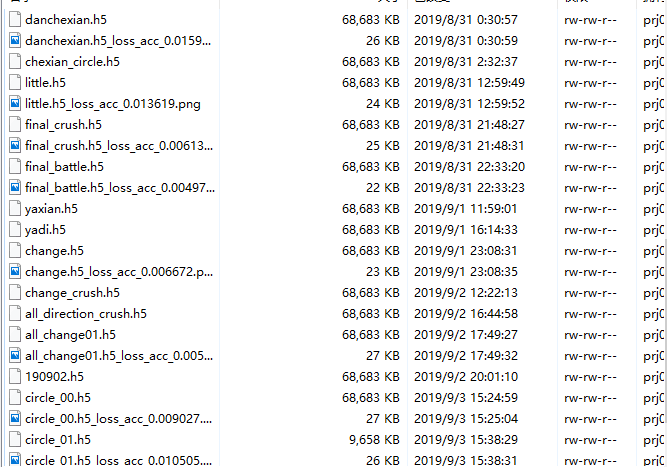
部分数据集：





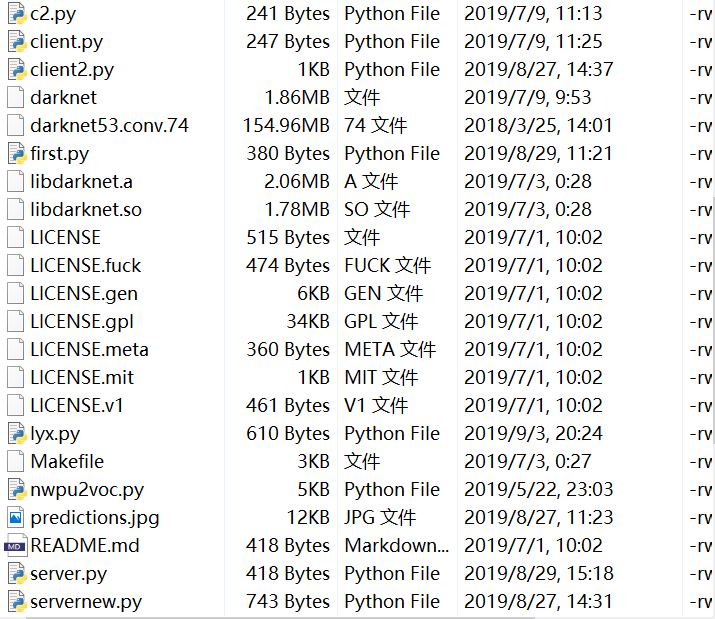
部分训练出来的模型：





通信部分：

部分通信所需的文件（所有的python文件）



**6.3.实验方法：**

阅读代码

我们的课题需要多个模型，因此必须实现模型切换。而donkeycar原本必须在网页上将模式从user改成local pilot才能开启自动驾驶。为了能够实现模型的切换，我们通过阅读代码找到了不能直接开始自动驾驶的原因，发现是因为小车各个模块的开启顺序导致的，于是我们加入了延时，以开启变慢为代价实现了小车能够直接开启自动驾驶。

控制变量：

由于经历了将近一周半的训练后我们发现呈现的训练效果并不好，而且都是无用功。因此我们为了鉴别问题所在我们使用了控制变量法。

判断是否是数据集的问题：在训练初期，我们为了提高照片的清晰度把像素从160x120调整为了320x270。照片的清晰度提高了很多但是同样模型的大小也提升了很多，结果发现在运行模型的时候处理速度很慢，小车在遇到拐弯的时候经常会因为处理速度太慢而发生拐弯拐不过来的现象，于是我们重新将像素调回了160x120。

专项训练：

在初期训练小车循迹时，小车在拐弯后很难回到原轨道直走，因此我们将拐弯过后回到原轨道，再继续直走的部分进行了多次训练，与之前的数据集放在一起训练，发现训练效果有显著的提升。

除此以外我们对发生问题较为频繁的问题都进行了专项训练。

迁移学习：

由于视野内若未出现垃圾或垃圾回收处时小车都将沿固定轨道寻迹，通过迁移学习，转圈的数据可以被有效地运用于训练别的模型。

通信部分：

树莓派与arduino的通信：在网上搜寻资料，编写代码，多次调试。树莓派与服务器，服务器与pc，pc与树莓派的通信：原本在理想情况下我们希望树莓派能够在照片拍摄结束后直接在树莓派上进行图像识别，这样既可以减少时延也可以让捡垃圾小车成为一个独立的个体。但是由于往树莓派上安装库非常困难，而且以树莓派的运行速度处理图像识别会非常缓慢，因此我们想利用GPU来代替树莓派进行图像识别，然后将结果返回给树莓派。但是服务器与树莓派不在一个子网下，无法利用socket通信，于是我们又借助了pc，通过xftp让pc上的文件实时更新，再利用socket通信将图像识别的结果回传给树莓派。

# 课题成员分工与合作情况说明

杨奕冉、余康佳：主要负责donkeycar操作，环境配置、训练小车采集数据、实现基本自动驾驶功能。

赵行越：负责机械臂的控制

罗如瑜：负责小车自动驾驶数据模型训练、垃圾分类数据模型训练

李韫瑄、许竞舟：负责全车各模块整体调控、包括小车靠近垃圾后调用摄像头拍摄并识别、发出信号让机械臂开始工作、在小车回到指定位置之后根据垃圾的类别放到不同的位置。

# 课题所用器材列表及说明

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 名称 | 型号 | 数量 | 单价 | 主要功能 |
| 1 | Donkeycar小车 |  | 1 | - | 小车主体 |
| 2 | 树莓派 |  | 1 | - | 控制小车 |
| 3 | Arduino及机械臂 | meArm树莓派机械臂 | 1 | 203 | 用于控制机械臂 |

# 课题成果链接（包括视频、代码）

https://github.com/shengangxiong/xiaoxueqi

# 参加本课程的收获、体会及对课程的建议

**10.1.收获**

1、对于神经网络模型有了一定了解，浅显掌握其构造层与部分参数的意义。

2、可以修改pi中的一些文件和参数以达到不同的训练效果。

3、对于软硬件结合的项目有了实践操作认识。

4、学会了跨硬件的串口通信，树莓派与电脑、电脑与服务器、电脑与arduino UNO板子的5、通信、树莓派与arduino UNO的通信。

6、虽然在第一个模型中尝试了许多方法都没有得到很好的训练效果，但是熟练了这个过程7、的操作，以及不同训练方法的侧重训练点。

8、组装机械臂时和改装小车时增强了动手能力。

9、学会了基础的linux命令操作。

10、进一步提升了用深度学习网络的技能，之前都是只简单的学习基础知识和复现网上现有代码。

11、从自己收集数据，用labelimg处理自己收集到的数据，配自己服务器的接口以及数据对应的接口，从最基础的图像识别Restnet到yolo，再到读今年四五月论文读到的比较新的cornernet等，自己的读论文的速度，以及对图像识别网络整体认知进一步提升，自己编程能力也提升很多，知道了很多错误的解决方案。

**10.2.体会**

1、在不明白神经网络底层构架时的训练，就是碰运气，因为不明白其内部特性，所以做的大部分尝试都是浪费时间。

2、神经网络训练出的模型具有不确定性，在同一个环境条件下，小车可能做出不同的反应。

3、尽管组内分工明确，但最后大家都会去集中完成难点部分。

4、通过本课程我对神经网络以及深度学习有了简单的认识，并通过自己操作实践，体会到了编写程序的人对神经网络的理解，同时对raspberry pi和arduino都有了更深的认识。

5、有关donkeycar的训练，这个用的基础网络比较简单，再加上硬件所限并不能跑很大的网络（由于延迟），因此这个在已有数据集以后自己训练并不难，但收集数据比较困难，因为分辨率，大车故障，相机等硬件问题一定程度上限制住了组内的进度。

6、在研究小车代码方面，在助教提出了通过延时控制时间，以及一点点改代码看效果，终于在通宵后大致改出了可以自启的代码，组内大家也都一起在研究，能帮忙的地方大家都会互相帮忙。

7、不能只在软件层面停留，与硬件相结合很重要，项目的流程一定要清楚，再加上大家都很棒很配合，这次课程受益匪浅。

**10.3.建议**

1、在课程前几多一些理论知识方面的培训，虽然有几次助教讲解神经网络模型，但是效果和实用性都不是很好。只记得一位老师讲解模型时讲得十分详尽，非常喜欢她。

2、老师在和我们交流时都十分耐心，但是也更加希望老师了解一些具体实现中的阻碍，就像有些项目需求是不切实际的，也并没有不了解技术时想象的简单。

3、希望助教可以将理论与代码相结合，将某一层与donkeycar文件里某一行代码对应讲解，这些虽然我们自己也可以找到，但是会话费很多不必要的时间。

4、希望课程能够在课程开始时向同学们介绍donkey car时更为详细，告诉同学们donkey car的学习程度与学习方法。

# 参考文献

【1】Hei Law, Jia Deng，CornerNet: Detecting Objects as Paired Keypoints，3 Aug 2018 (v1)，[arXiv:1808.01244](https://arxiv.org/abs/1808.01244) [cs.CV]

【2】Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, Qi Tian ，CenterNet: Keypoint Triplets for Object Detection，[arXiv:1904.08189](https://arxiv.org/abs/1904.08189) [cs.CV]

【3】Joseph Redmon, Ali Farhadi，YOLOv3

【4】R. Girshick. Fast R-CNN. In ICCV, 2015.

【5】Karen Simonyan& Andrew Zisserman Visual Geometry Group, Department of Engineering Science, University of OxfordVGG(very deep convolutional networks for large-scale image recognition) [cs.CV]2015

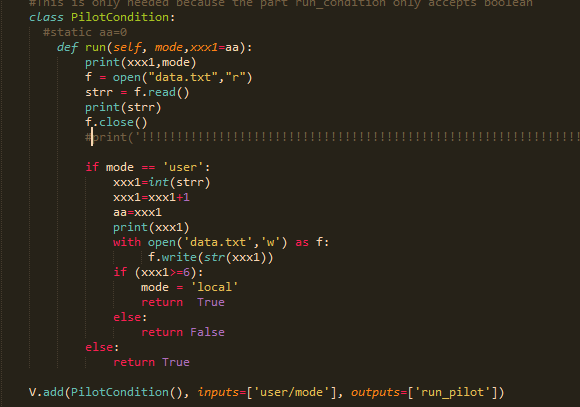
【6】无数CSDN博客

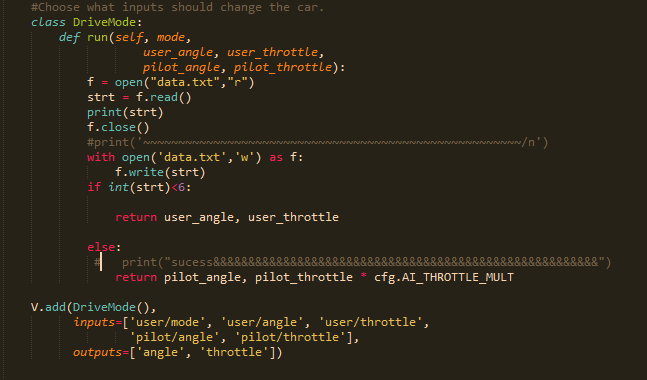
# 附件1：课题执行过程中的问题及解决方法

一、开机自启

donkeycar 自启（创新）  
基于考虑到需要全自动的要求，需要不经过点击网页中“start vehicle”以及选择“local”，上电后就自动开始自动驾驶。  
  
询问了之前并未有组实现过完全自启，我们组共同努力下通过对不同的模块，主要通过在manage.py等中全局变量，增加不同的延时以控制模型加载与图形输入的时间先后，以防出现“Nontype”等错误。  
  
我们组在实现开机自启后也对代码进行了共享并及时与老师助教沟通。

部分代码如下



二、切换模型

我们的课题有在同一个场景下需要给出两种模型，因此存在模型切换的问题。我们训练了两种完全不同的模型，因为不能通过命令行进行切换，我们结合开机自启加入执行条件实现了模型切换。

三、训练小车

训练小车是我们组投入时间精力很大的部分，在训练过程中遇到了层出不穷的问题，我们在训练过程中不断分析可能出现问题的原因，并针对问题提出各种可能的解决方案，以实现最好的训练效果。  
  
1、无线控制树莓派：训练之初会出现树莓派无法连接热点的问题，经分析发现树莓派的sd卡中保存的wifi配置文件会不停的消失，但在与助教交流之后发现这个是插拔sd卡可能会出现的正常现象，我们又通过更改电脑热点的适配器选项，并且通过限制热点的网络频带来提高连接热点的速度和稳定性。  
  
2、小车马力值调试：小车的马力控制机制也给训练带来很大的困扰，小车的马力输出数值会随电池电量的不同而会有不同的速度，因此在训练过程中需要不断地调整小车马力的预设值来实现小车的平稳行驶。而且在自动驾驶模式下，马力输出显得更不稳定，在短时间电量并没有明显变化的情况下小车的行驶速度也会有着较大幅度的变动，因此对模型测试产生很大的影响，我们在训练过程中需要不停的通过调参来强制对马力值进行微调才能使小车平稳行驶。  
  
3、小车摄像头调试：小车的摄像头也是我们投入大量精力调整的部分，最开始我们为了追求返回照片的质量，将摄像头分辨率调整到320\*270甚至1920\*1080，在训练之初并没有发现什么问题，但是模型训练效果总是不理想，并且在运行一段时间之后树莓派的芯片会非常的烫。在询问助教了解情况之后发觉可能使图片质量太高，在执行自动驾驶模式的时候树莓派处理速度不够，导致不能对场景进行快速准确的识别，具体体现为在看到转弯标识之后有一点转弯的反应但是不能很好的将转弯进行到底。在将摄像头分辨率下调至160\*120之后，可以观察到在执行自动驾驶模式的时候屏幕输出马力值的速率明显加快，证明树莓派运行速率上升，相应的在遇到转弯等明显标识的时候做出的反应也会更加的灵敏。  
  
4、模型融合与转换（难点）：最难解决的还是不同模型数据组合之后效果不好的问题。我们的小车在不同阶段需要加载不同的自动驾驶模型，不同阶段的任务完全不同，所以在训练阶段我们需要分任务训练不同的模型。这些模型在单独执行自动驾驶模式的时候基本都能实现预想的效果，但是将数据合起来训练之后生成的总模型效果就会变差，各阶段的任务都不能很好的实现。在与助教和老师交流之后，我们发现不同阶段数据合起来时的数据比例也很重要，并且在总模型效果不好的路段要进行强化训练，增加这部分路段的数据收集，加强训练。在这个思路下我们重新控制各阶段的数据比例进行训练，并且仔细观察小车行驶情况，在易出问题的路段进行强化训练，最终能够训练得到一个能较为满意的执行各阶段任务的总模型。

四、图像分类

为了进一步优化垃圾识别分类，我们尝试过四个最基础的神经网络，即inception\_net，alexnet，restnet，mobilenet，vgg网络进行测试，虽然对设备要求低，但是考虑到后期进一步的优化潜力以及性价比，比较基础的网络有些低识别率而且再拓展性比较差，所以选择yolov1-v3。   由于yolov1的坐标的不准确性，甚至效率并不高于基础网络；yolov2只是增加了些训练方法，为跨数据集训练提供了思路；yolov3相比于就比较适合，而且由于darknet官网的进一步完善更新，其中的lite版本很适合高速轻量级网络。  
    tensorboard展示训练过程mAP约为89（yolo训练了2个小时） 其后我们还尝试了最新的centernet以及cornernet的2019年四五月份新出的onestage网络，与宣传的高效对应，这些网络对硬件要求比较高，运行时占用显存比较严重，而且对精准度提升不大，考虑到因为这个网络主要用于姿态检测，所以可能在图像分类上并没有很高效的表现。

数据采集制作也略困难，需要人工处理收集到的数据集，首先生成对应图片txt格式 （yolo要求标准格式，xml也可），而后通过编写代码自动划分数据集以及配置数据集接口进行训练。  
  
后期评估网络mAP时，需要将数据集转换为voc格式。

五、通信部分

树莓派实现图像识别得到结果

1、树莓派内实现拍照后进行图像识别，自行返回结果。后发现树莓派内存小、运行速度慢、库难安。

2、借助服务器的高速处理能力，我们改为树莓派向服务器发送图片，服务器图像识别结束后将结果返回给树莓派。我们想使用socket通信，但是服务器和树莓派不在一个子网下，无法回传结果给树莓派。

3、我们找到PC是能够和服务器以及树莓派同时通信的载体，因此我们以PC作为中介，通过树莓派传照片给服务器，服务器传给PC，PC再传给树莓派。

六、机械臂部分  
1.使用官方给出的案例文件，机械臂达不到预期的位置，效果较差。  
 解决：查询相关资料，重新整理一遍官方文档之后，发现是舵机未初始化的问题。拆下舵机，上电使其回到初始的位置，再装入机械臂中，效果较为理想，机械臂能够被控制。  
  
2．收到来自树莓派信号之后开始预置的动作，动作完毕后未停止在最后的位置。  
解决：使用一个变量，当动作开始时置为1，每一遍循环判断其值是否为1，若为1则强行置为最后位置。

# 附件2：有用文档及链接

https://github.com/shengangxiong/xiaoxueqi