

## -目录/Index

---

智能车简介 .....	3
第一节：整车系统概要 .....	4
1.1 整体外观.....	4
1.2 硬件明细.....	4
1.2.1.基本硬件组成 .....	4
1.2.2 硬件连接关系 .....	5
1.2.3 实现功能原理 .....	6
第二节：连接小车 .....	8
2.1 开机.....	8
2.2 连接到 Edegboard.....	8
第三节：车道线识别_数据采集 .....	11
3.1 开启控制程序 .....	11
3.2 手柄控制移动 .....	11
3.3 下载数据集 .....	12
第四节：车道线识别_数据处理+模型训练.....	14
4.1 项目准备+上传数据集.....	14
4.2 数据处理+模型训练.....	14
4.3 下载模型.....	15
第五节：车道线识别_自主移动 .....	16

5.1 上传模型.....	16
5.2 自主运行.....	16
第六节：标志物检测_数据采集 .....	18
6.1 开启控制程序 .....	18
6.2 下载数据集.....	18
第七节：标志物检测_数据标注 .....	19
7.1 开启 labeling 软件 .....	19
7.2 整理文件目录 .....	19
7.3 标注 .....	19
第八节：标志物检测_模型训练 .....	21
8.1 构建项目 .....	21
8.2 上传数据集.....	21
8.3 打开项目 .....	22
8.4 训练模型.....	23
8.5 下载模型.....	25
第九节：标志物检测_自主移动 .....	27
9.1 上传模型和 labellist.....	27
9.2 自主运行.....	27

## 智能车简介

2019 年的特斯拉自动驾驶开放日上，特斯拉人工智能高级主管 Andrej Karpathy 强调特物理数据无法代替，对于依赖虚拟仿真自动驾驶，特斯拉更相信现实物理数据。也就是说，看图比雷达更真实。在发布会后环节中，马斯克也再次重申自己的态度，我们不用激光雷达，这就是态度。

无人驾驶（深度学习）智能车采用 Python 编程语言，以深度学习开源框架百度飞桨 paddlepaddle 为基础，高度集成硬件驱动模块，分布式结构化软件设计框架，可实现数据采集、数据模型构建、自主识别弯道、无人驾驶验证等功能，是一套学习深度学习开发的最优平台。

PaddlePaddle，中文名称：飞桨；作为国内唯一功能完备的端对端开源深度学习平台，集深度学习训练和预测框架、模型库、工具组件、服务平台为一体，其兼具灵活和效率的开发机制、工业级应用效果的模型、超大规模并行深度学习能力、推理引擎一体化设计以及系统化的服务支持,致力于让深度学习技术的创新与应用更简单。

AI Studio 是基于深度学习平台飞桨的一站式 AI 开发平台，提供在线编程环境、免费 GPU 算力、海量开源算法和开放数据，帮助理发者快速创建和部署模型。

# 第一节：整车系统概要

## 1.1 整体外观



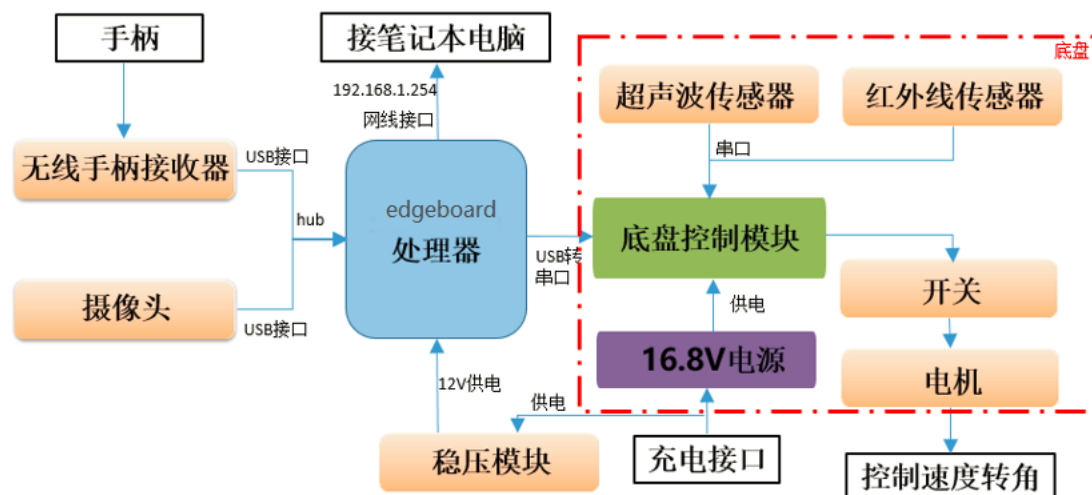
## 1.2 硬件明细

### 1.2.1.基本硬件组成

序号	类型	参数
1	底盘	四轮带编码器差速底盘。 尺寸:340mm*270mm*300mm（长宽高）
2	电机	G37-520B 编码器直流电机 12V 空载转速：178rpm
3	处理器	edgeboard
4	摄像头	像素 720基本P 对角 70 度 水平 55 度 YUY2/10-15 帧/S
5	稳压模块	输入17~58V 输出12V
6	手柄	2. 4GHz RF无线传输

		2节AA电池（7号）
7	充电电源	16.8V 1A

## 1.2.2 硬件连接关系



电源管理部分：

充电：

通过 16.8V 适配器插入家用电 220V 中，另外一头来连接电源充电口(1.1 中标出)。

供电：

内置电源直接为底盘控制模块供电，通过开关间接为电机供电，通过开关可以控制电机是否为上电状态

内置电源通过电源充电口，稳压模块给 Edegboard 处理器供电，通过插拔电源充电口来决定是否为板子上电。

信号传递方式：

笔记本电脑利用网线 ssh 访问 Edegboard 处理器本地，控制小车运行不同的程序；

手柄，摄像头，将数据传递到 Edegboard 处理器，用于进一步处理；

Edegboard 处理器通过 USB 转串口模块向底盘控制模块发送信号进而控制小车移动的速度和角度

超声波传感器、红外传感器直接与底盘控制模块相连，将数据传递到底盘控制模块，方便用于学习嵌入式相关知识。

### 1.2.3 实现功能原理

车道线识别：

**目的：**通过卷积神经网络，将摄像头获取的图像信息转化为小车自主运行的转弯角度。

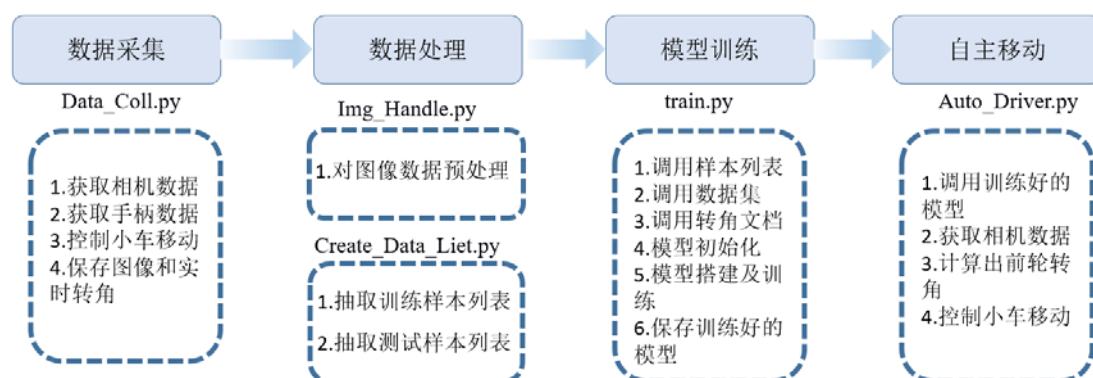
**实现方法：**

第一步：通过手柄控制小车在赛道上移动，获取摄像头图像以及小车转弯角度信息；并将信息保存到 Edegboard 处理器上；

第二步：将数据上传到线上 AIstudio 平台，对数据进行处理，提取出车道线信息；

第三步：在 AIstudio 平台上，搭建卷积神经网络，训练出用于自主移动的模式；

第四步：将模型移到 Edegboard 处理器上，移除手柄；通过摄像头获取的图像，利用训练好的模型计算出当前的角度值，进而控制小车自主移动。



## 标志物检测:

**目的:** 通过常见的目标检测架构, 提取出摄像头获取的图像信息中存在的标志物(人行道, 限速标志, 取消限速标志, 左右转弯, 取消左右转弯, 停车标志等)。

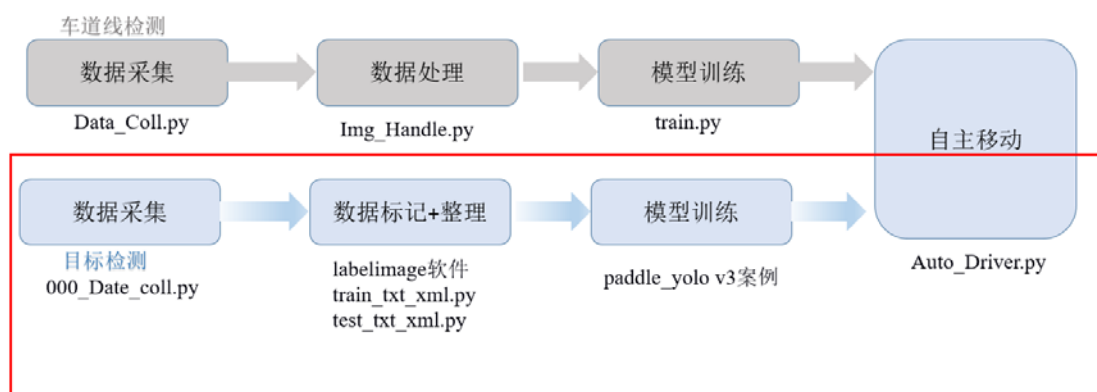
### 实现方法:

第一步: 通过人工在赛道上摆动小车, 获取包含标志物的图像;

第二步: 将数据下载到本地电脑上, 利用 **labelimg** 软件对图像数据进行标注, 标注后整理目录上传 **AIstudio**;

第三步: 在 **AIstudio** 平台上, 利用自己优化后的目标检测架构, 训练出用于标志物检测的模型;

第四步: 将模型移到 **Edeboard** 处理器上, 结合车道线识别部分的内容; 实现小车沿车道线移动的同时, 检测到周边标志物, 并对识别的标志物信息做出相应反应(遇到减速标识慢行, 遇到停车标识停车, 遇到车位入库等操作)。





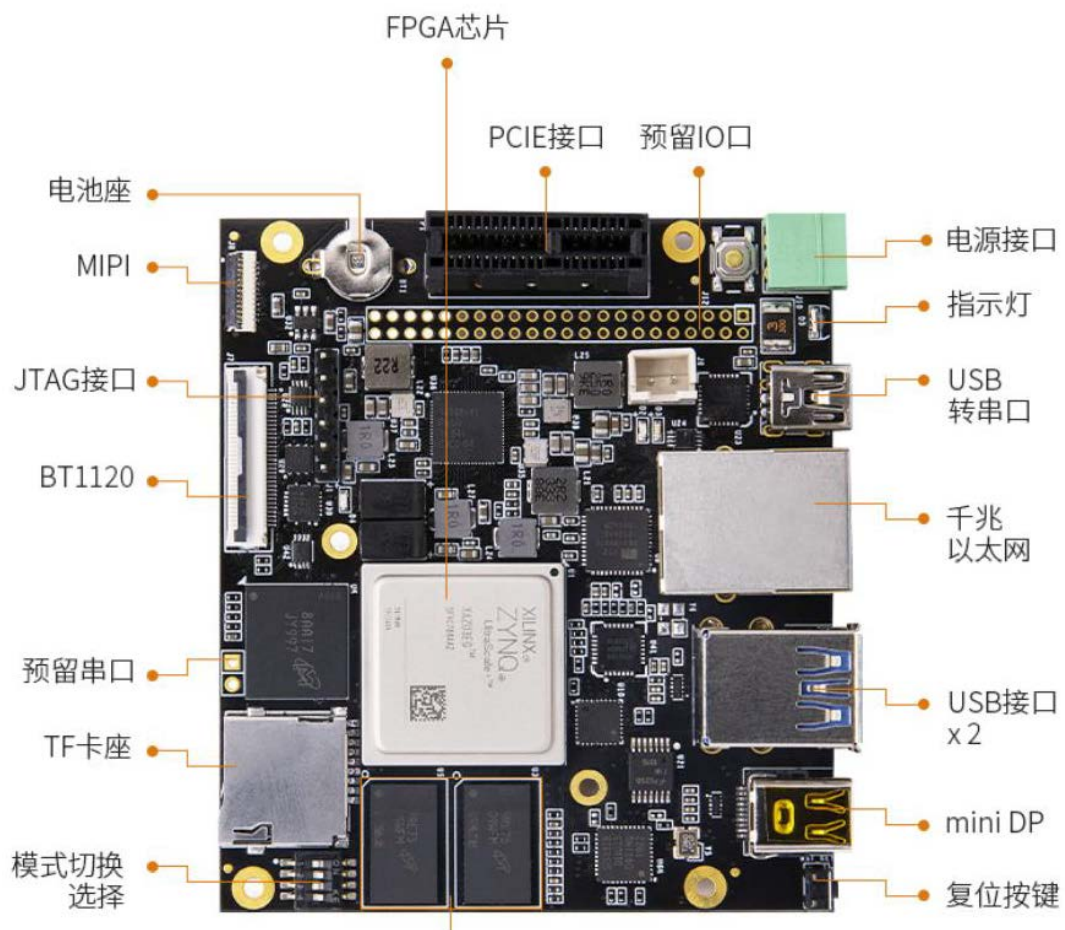
## 第二节：连接小车

### 2.1 开机

开机时，将稳压模块插入电源充电口，此时 Edegboard 上电。  
注意此时不需要开启底盘开关。

### 2.2 连接到 Edegboard

配置电脑网口 IP：

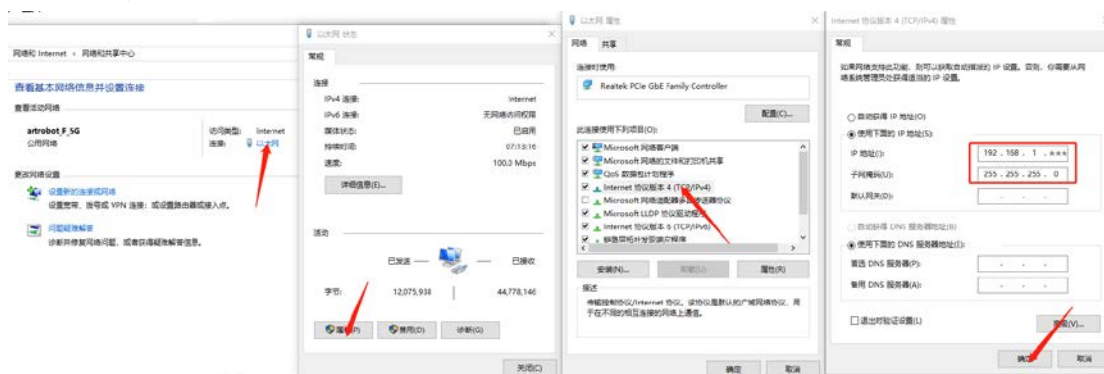


将网线插到 Edegboard 的千兆以太网口上，将网线另一端插在笔记本电脑上，单击“网络和 Internet 设置”后，打开“网络和共享中心”；



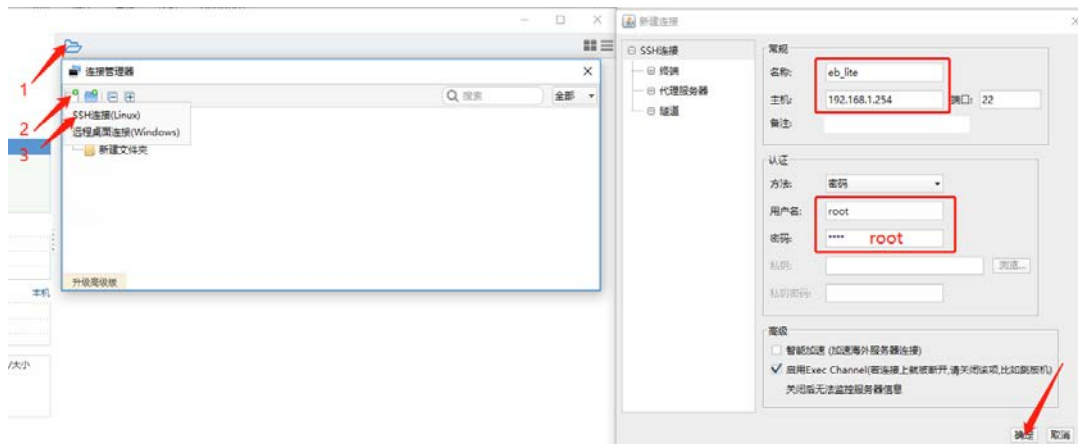


在网络和共享中心中，依次打开 “以太网” → “属性” → “Internet 协议版本 4 (TCP/IPv4)” → “使用下面的 IP 地址” (输入 IP 地址 192.168.1.\*\*\*, \*\*\*可以为 1~253 任一整数);最后单击“确定”，配置完成。

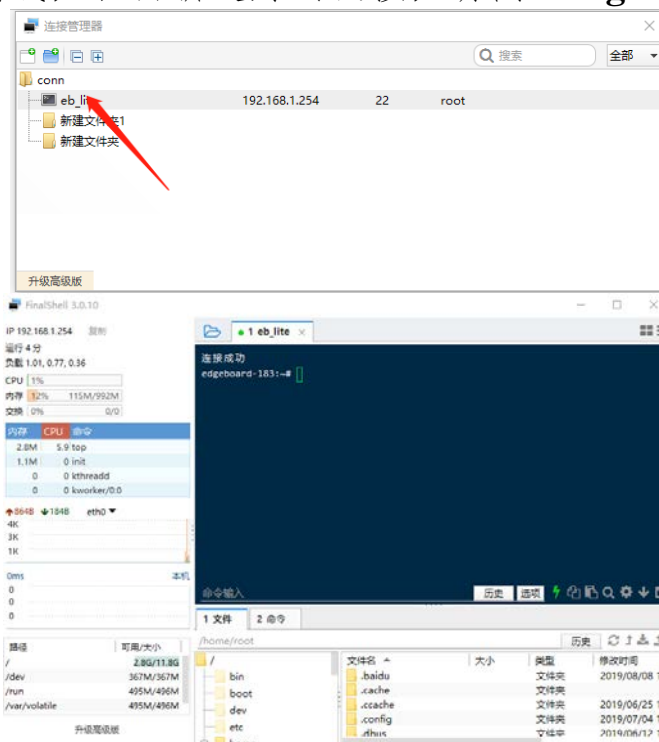


## 连接到 Edeboard:

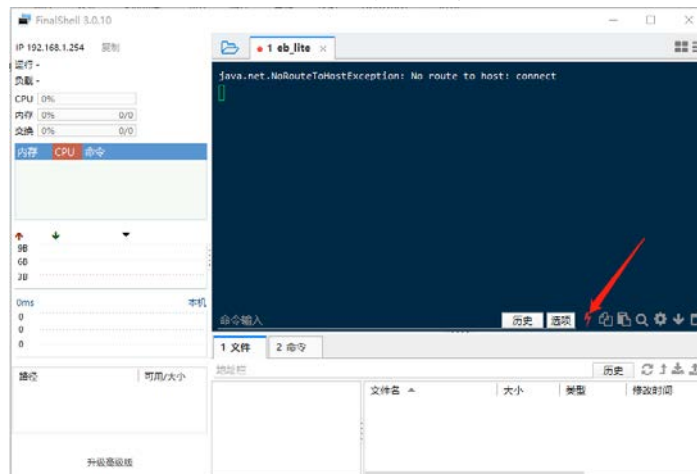
下载软件 finalshell 并打开，打开文件夹（1 位置），新建 ssh 连接（2，3 位置），在跳出框中输入相应信息，名称：\*\*\*（英文即可）；主机：192.168.1.254；用户名：root;密码：root。



连接上网线，双击新建好的链接，访问 **Edeboard** 本地



若出现图示问题，表示没有连接成功，检查网线来连接问题后，单击箭头指的按钮，进行再次连接。



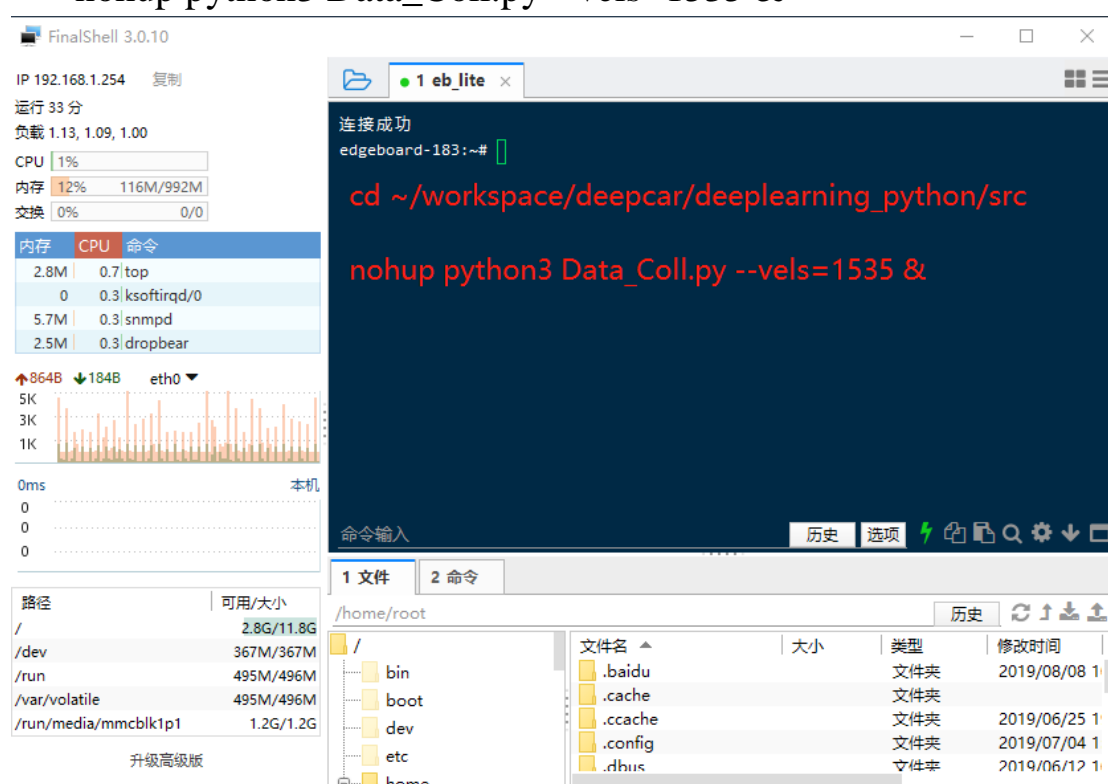
## 第三节：车道线识别\_数据采集

### 3.1 开启控制程序

确保 Edeboard 上电，底盘开关为关的状态下，按照第二节的内容连接到小车，在终端输入：

```
cd ~/workspace/deepcar/deeplearning_python/src(回车)
```

```
nohup python3 Data_Coll.py --vels=1535 &
```



### 3.2 手柄控制移动

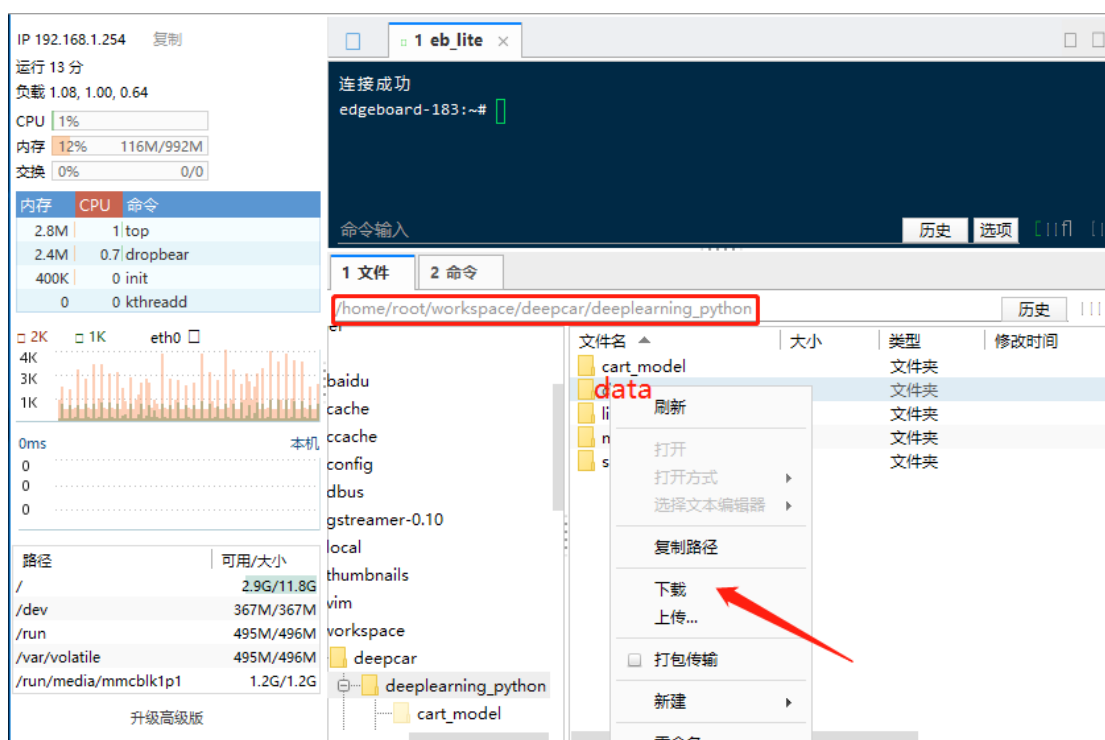
接着拔掉网线，将小车放在搭建好的跑道上，开启底盘开关，插上手柄接收头。通过手柄控制小车移动，首先将总开关调到 ON 档位，按上电按钮 **START**，再进行模式切换，指示灯红绿两个灯都亮表示模式切换正确。



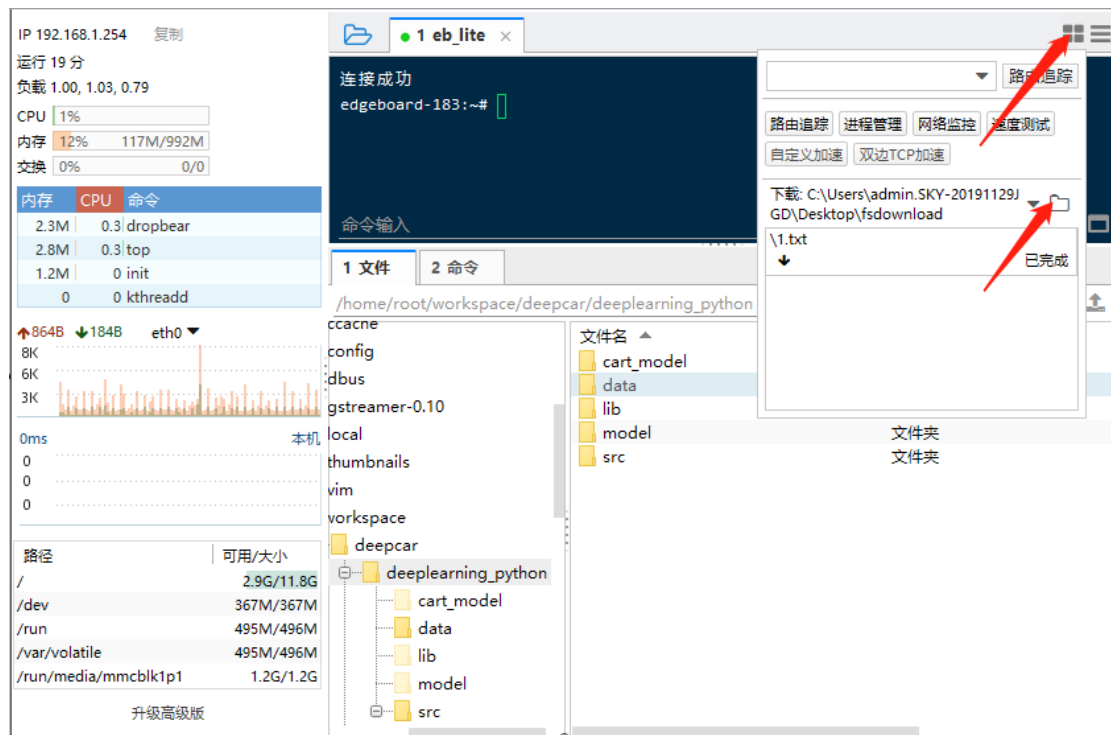
按动启动键，小车将开始跑起来采集数据了，通过转向遥感控制小车左右转弯。

采集结束后，通过按 4 次结束键结束采集数据。

### 3.3 下载数据集



插上网线，利用第二节的内容连接到小车，在 `/home/root/workspace/deepcar/deeplearning_python` 目录下生成了一个 `data` 文件夹，右击此文件夹，单击“下载”，通过右上角可以查看下载进度，以及下载的文件所在位置。



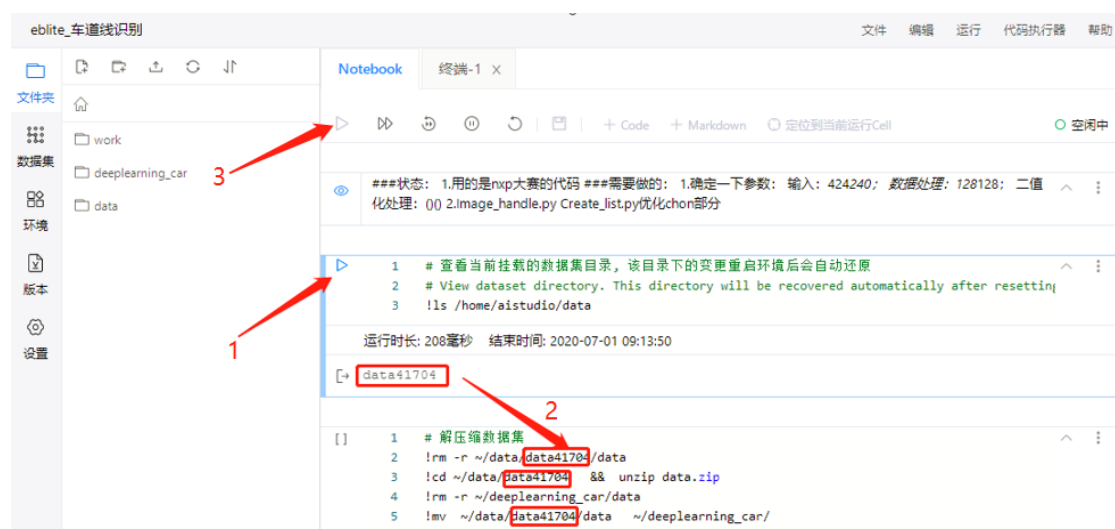
## 第四节：车道线识别\_数据处理+模型训练

### 4.1 项目准备+上传数据集

该部分准备工作与标志物检测的线上训练相似，此处仅提供基本流程，细节参考 8.1~8.3 的内容；

首先需要注册一个百度账号；登录 AIstudio 官网 (<https://aistudio.baidu.com/aistudio/index>)；将公开项目 (<https://aistudio.baidu.com/aistudio/projectdetail/563147>) fork 到自己的项目下；然后在车载电脑上找到~/deeplearning\_car/目录下将 data 文件夹压缩成 data.zip 文件，并将文件作为数据集上传至 AIstudio 上，用 gpu 资源打开刚才 fork 好的工程，将数据集换成上传的数据集并打开。

### 4.2 数据处理+模型训练

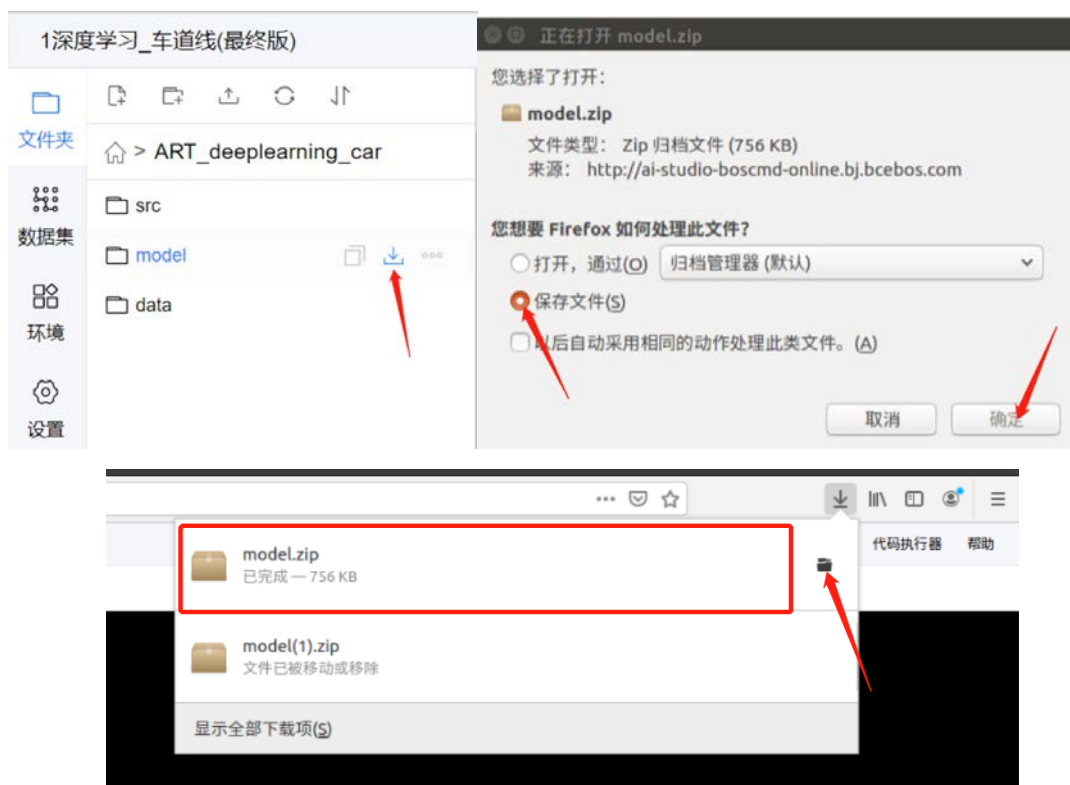


点击箭头“1”处运行第一个 cell，输出上传数据集的名称

更改第二个 cell 里面的内容，接下来点击“3”处，依次运行如下 cell。

### 4.3 下载模型

训练后在左侧目录栏中~/deeplearning\_car 目录下，可以找到 model 文件夹，单机“下载按钮”将文件夹下载到车载电脑上；选中“保存文件”并单击确定；此时保存到本地；通过单击“文件夹”图标可显示当前下载的 model.zip 文件。

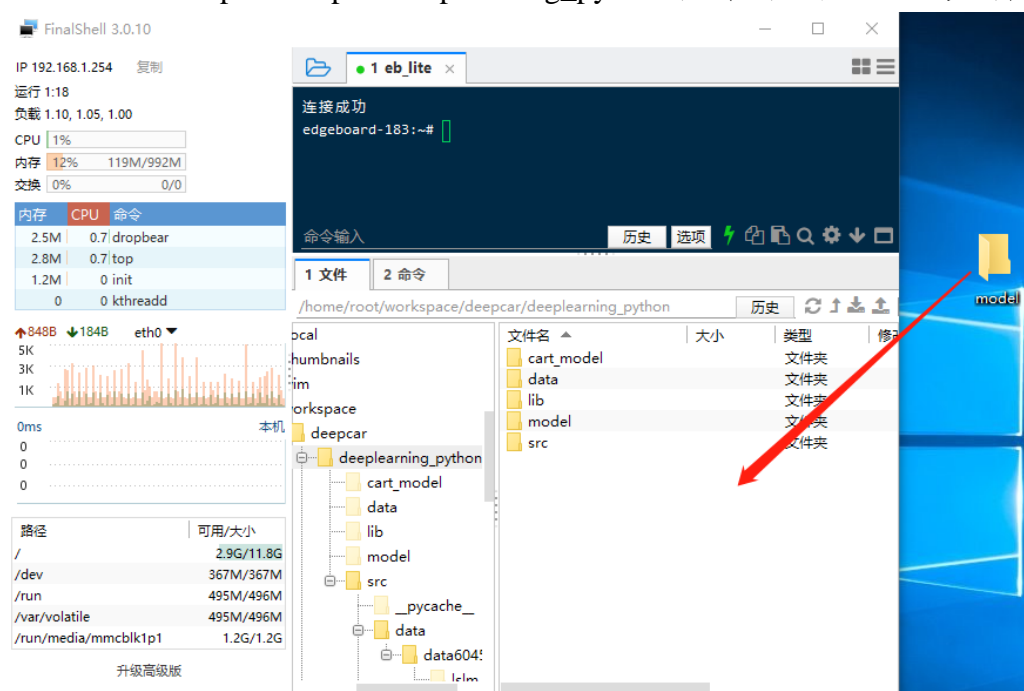




## 第五节：车道线识别\_自主移动

### 5.1 上传模型

将从 AIstudio 训练生成的 model.zip 文件在笔记本电脑上解压缩，并将解压缩以后的 model 文件夹替换 edgeboard 下 /home/work/workspace/deepcar/deeplearning\_python 目录下的 model 文件夹；



### 5.2 自主运行

确保 Edgeboard 上电，底盘开关为关的状态下，按照第二节的内容连接到小车，在终端输入：

```
cd ~/workspace/deepcar/deeplearning_python/src(回车)
```

```
nohup python3 Auto_Driver.py --vels=1535 &
```

拔掉网线，将小车放在赛道上，开启底盘开关，小车开始自主移动；

停止时，关闭底盘开关，将 edgeboard 上的摄像头插拔一下。

FinalShell 3.0.10

IP 192.168.1.254 复制

运行 1:15

负载 1.02, 1.04, 1.00


CPU 2%

内存 12% 119M/992M

交换 0% 0/0

内存	CPU	命令
2.5M	0.7	dropbear
2.8M	0.7	top
0	0.3	rcu_sched
1.2M	0	init

↑950B ↓334B eth0 ▼



本机

路径	可用/大小
/	2.9G/11.8G
/dev	367M/367M
/run	495M/496M
/var/volatile	495M/496M
/run/media/mmcblk1p1	1.2G/1.2G

升级高级版

1 eb\_lite x

连接成功

edgeboard-183:~#

cd ~/workspace/deepcar/deeplearning\_python/src

nohup python3 Auto\_Driver.py --vels=1535 &

命令输入

历史 选项

1 文件 2 命令

/home/root/workspace/deepcar/pd

历史

data

data604!

ls1m

ls1m-

freeze\_model

image

logs

pd

data

.ipynb\_check

freeze\_model

driver

paddle\_mobile

PaddleLite

pin-packages

文件名	大小	类型	修改
data		文件夹	
freeze_model		文件夹	
0.py	1.5 KB	PY 文件	
1_result.jpg	7.5 KB	JPG 文件	
1.jpg	19.6 KB	JPG 文件	
test.py	8.7 KB	PY 文件	

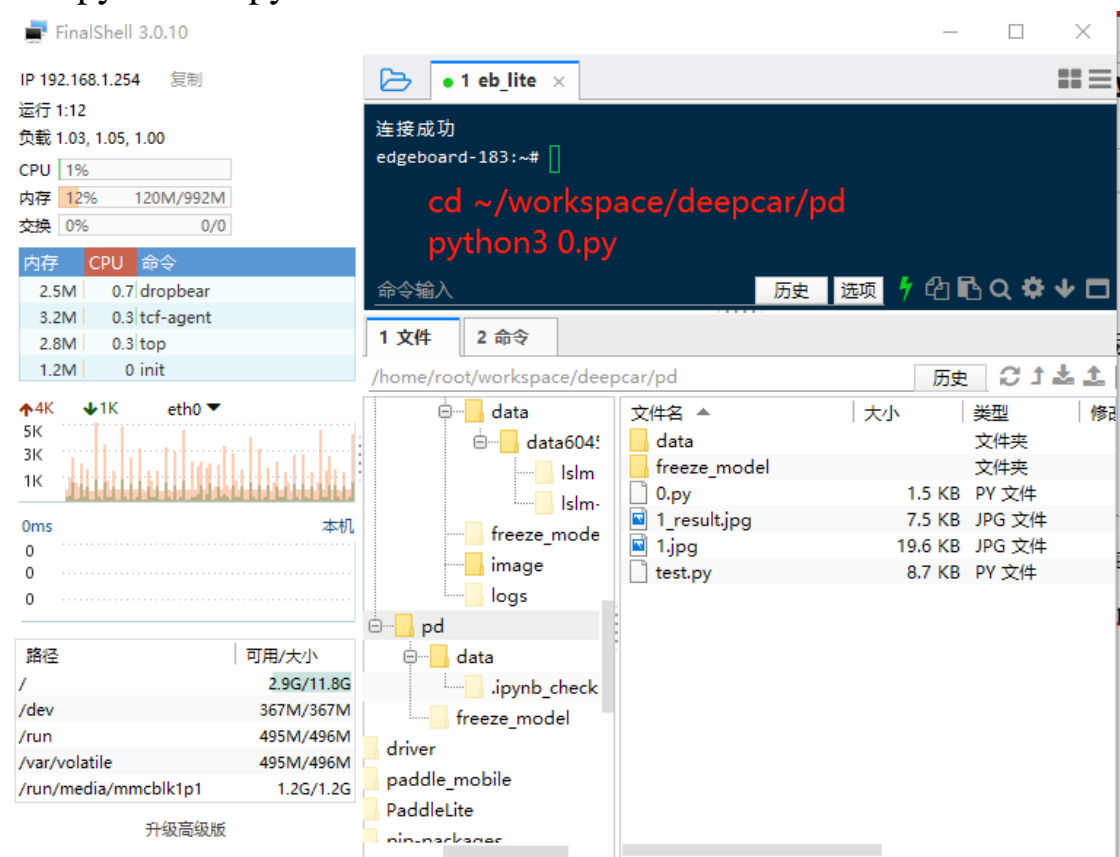
## 第六节：标志物检测\_数据采集

### 6.1 开启控制程序

确保 Edeboard 上电，底盘开关为关的状态下，按照第二节的内容连接到小车，在终端输入：

```
cd ~/workspace/deepcar/pd(回车)
```

```
python3 0.py
```



### 6.2 下载数据集

Edeboard 的 ~/workspace/deepcar/pd 目录下下载 data 文件夹，里面为采集的图像。

## 第七节：标志物检测\_数据标注

### 7.1 开启 labeling 软件

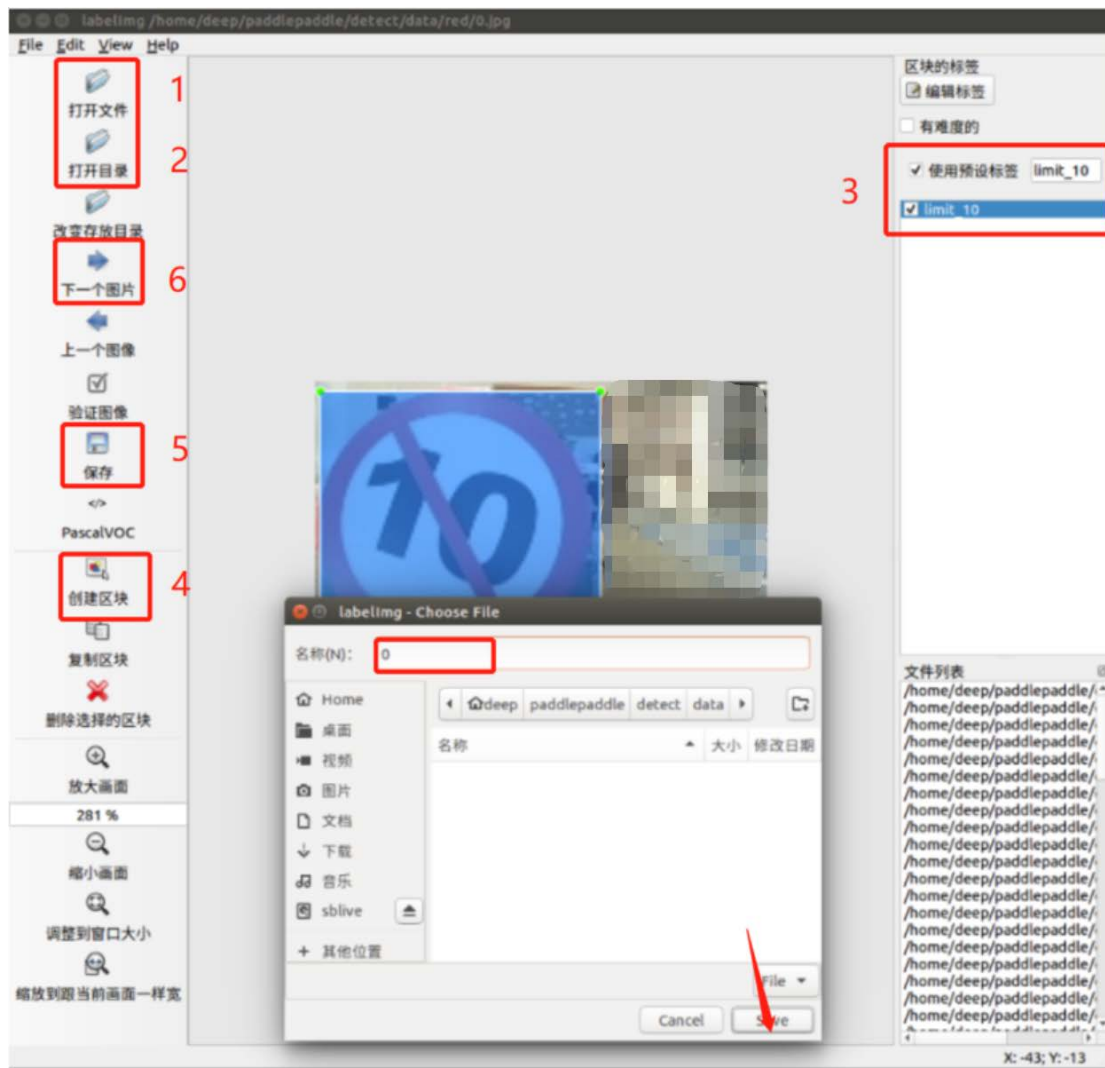
下载开源的软件 <https://github.com/tzutalin/labelImg/releases>;  
下载后解压到本地(本地路径中不能包含汉字),

### 7.2 整理文件目录

```
---data-----{ 标签 1 }----- rgb   #放标签 1 的图片
|
|           |           |-----1.jpg
|           |           |...
|           |           |-----n.jpg
|           |-----xml   #放标签 1 的标签（目前为空）
|
|-----{ 标签 2 }
|...
```

### 7.3 标注

运行软件： 1.单击“打开文件”图标，选择图像目录  
~/paddlepaddle/detect/data/{标签名}/rgb；此时会显示目录下的图像； 2.单击“打开目录”图标，选择图像标签保存目录  
~/paddlepaddle/detect/data/{标签名}/xml； 3.在右侧“使用预设标签”前打勾并在后面输入标签名； 4.单击“创建区块”按钮给当前显示图像打标签； 5.单击“保存”保存当前标签文件，此时弹出对话框，“名称”处不用改动，单击“save”保存 6.然后单击“下一个图片”切换到下一张图片,循环执行 4、5、6 三步直至此文件夹标记完，然后按照切换到下一个文件夹下进行标注。



## 第八节：标志物检测\_模型训练

### 8.1 构建项目

此次标志物识别的开源项目地址为：  
(<https://aistudio.baidu.com/aistudio/projectdetail/596152>)；进入网址，单击右侧“fork”键，fork到自己的项目里；

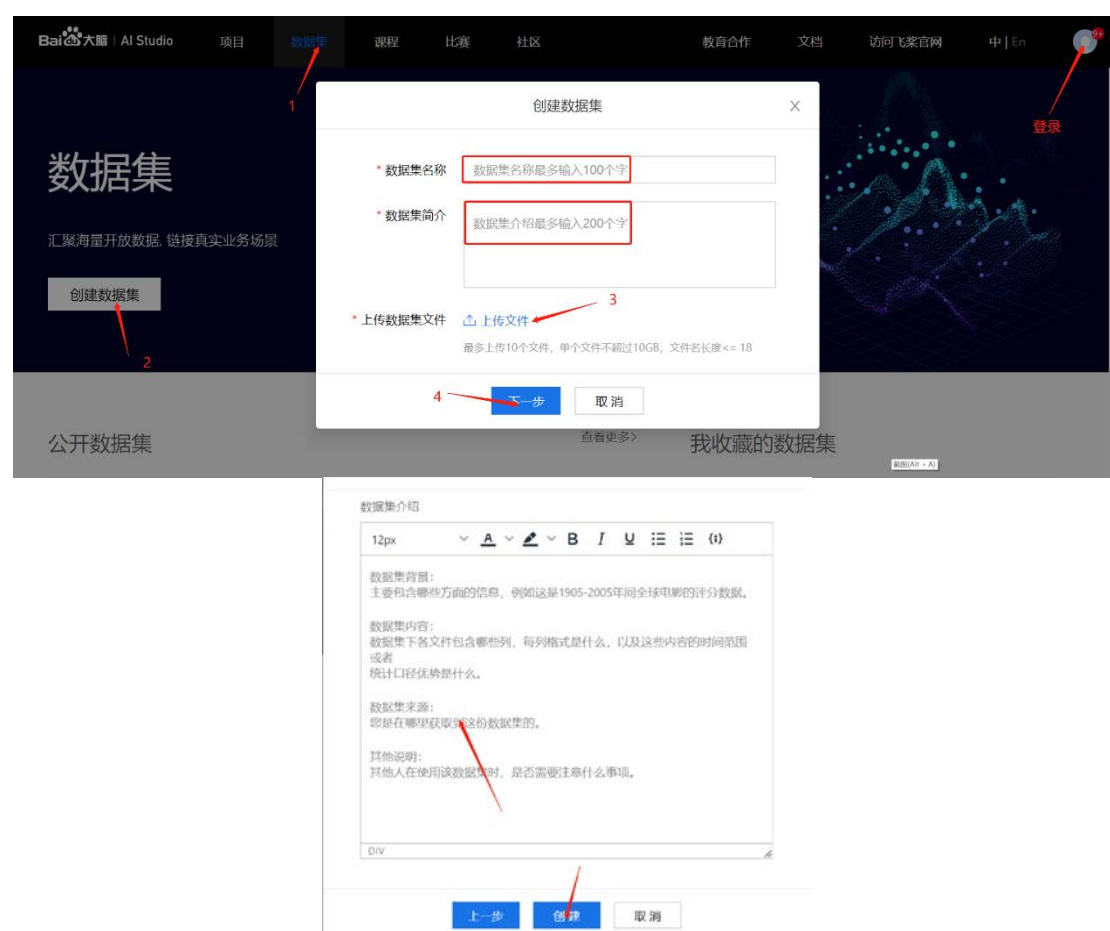


回到 Aistudio 首页，单击标签栏“项目”，然后单击“我的项目”，就可以看到刚才创建的新项目。



### 8.2 上传数据集

在 AIstudio 官网上，首先单击右上角的登录按钮，登录个人账户；接着单击标题栏的“数据集”；接着单击“创建数据集”，此时会弹出对话框，添加数据集名称，数据集简介；接着单击“上传文件”，接着单击“下一步”在弹出对话框下面选择“创建”，该数据集创建完成。

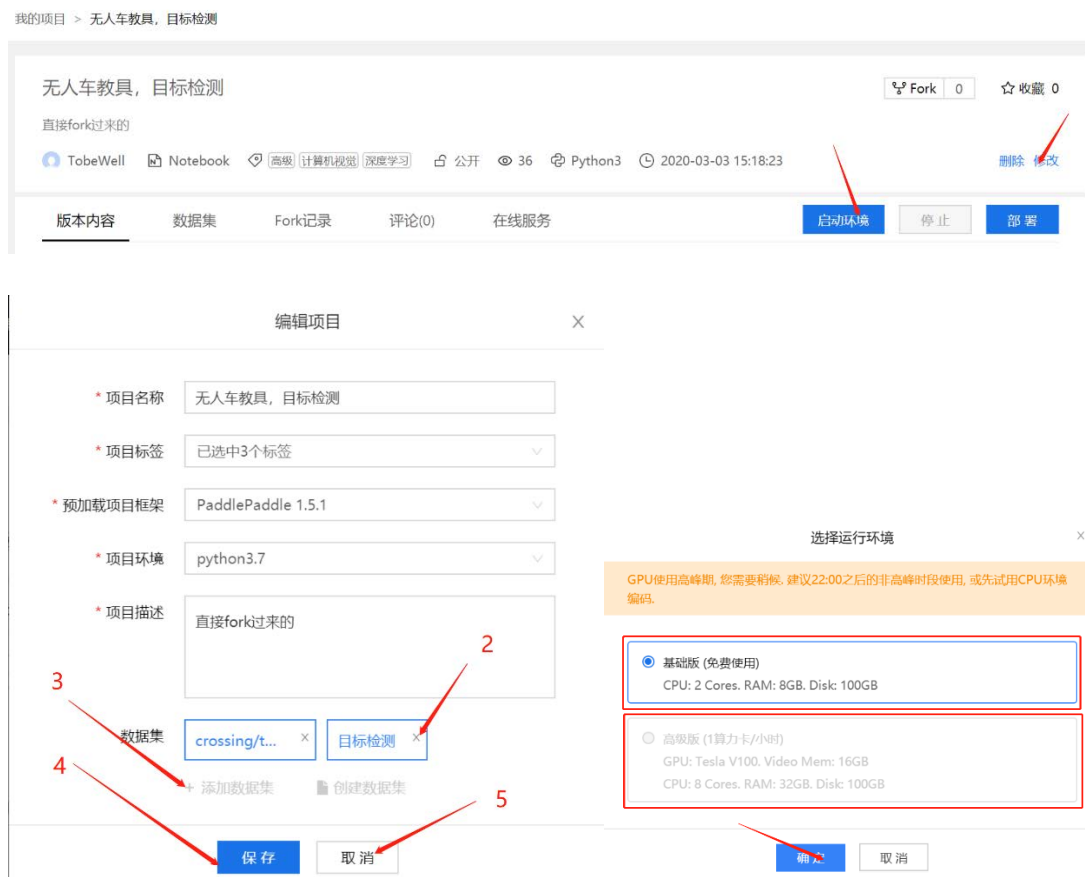


## 8.3 打开项目

在 8.1 中我们创建了项目，单击创建好的项目；进入到项目中；单击“修改”，跳出编辑界面，单击 2 处删除已有的数据集，单击“添加数据集”选择 8.2 创建的数据集，最后单击“保存”“取消”；接着单击“启动环境”，此时跳出另外一个对话框，在



对话框中选择使用 GPU 资源还是 CPU 资源；建议优先使用 GPU 资源，该项目默认代码是在 GPU 环境下运行的；最后单击“确定”。



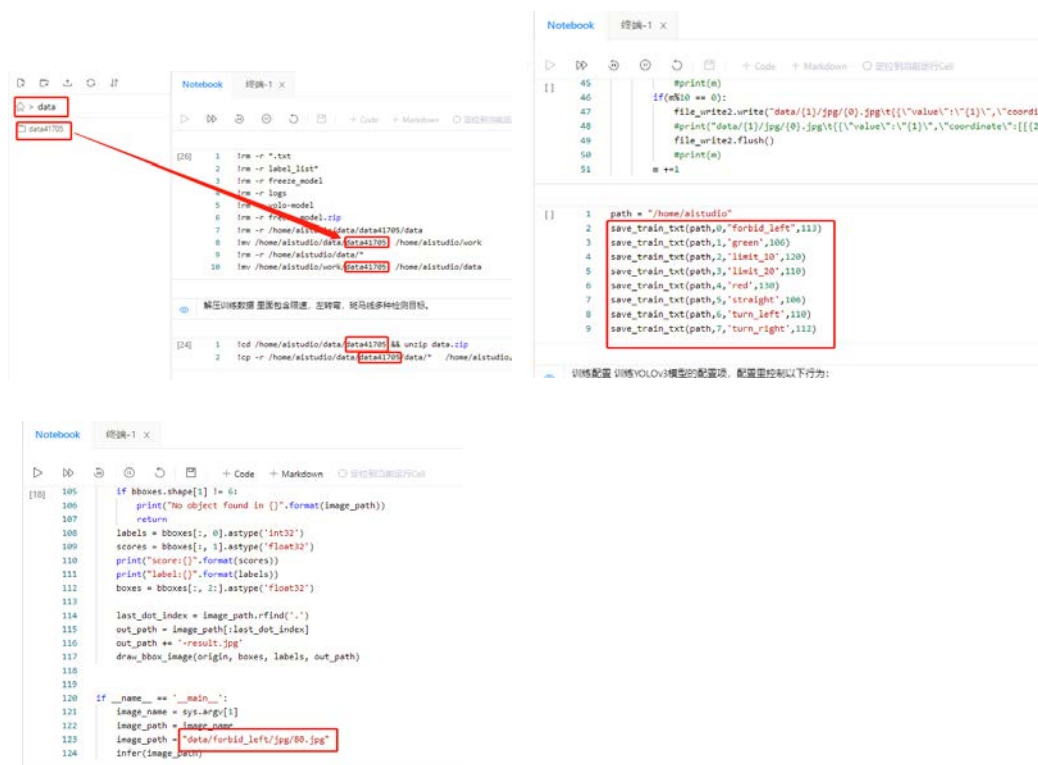
## 8.4 训练模型

打开项目后，首先在左侧看一下 **data** 目录下包含的数据集；然后找到自己上传的数据集，将右侧相应位置进行修改；

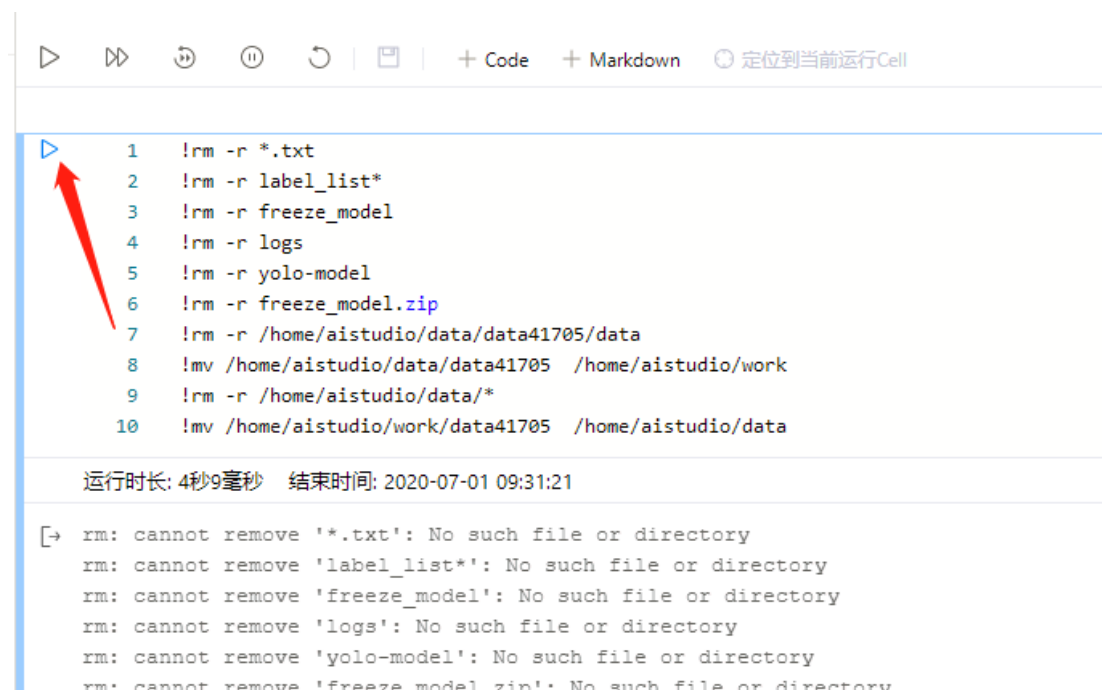
在数据处理的时候；需要根据自己的数据；按照以下格式，对代码块进行修改；`save_train_txt(path,第几个,'标签名称',采集照片数量)`

最后测试效果的时候，改代码块中 `image_path` = “文件目录”，

可显示当前训练效果。

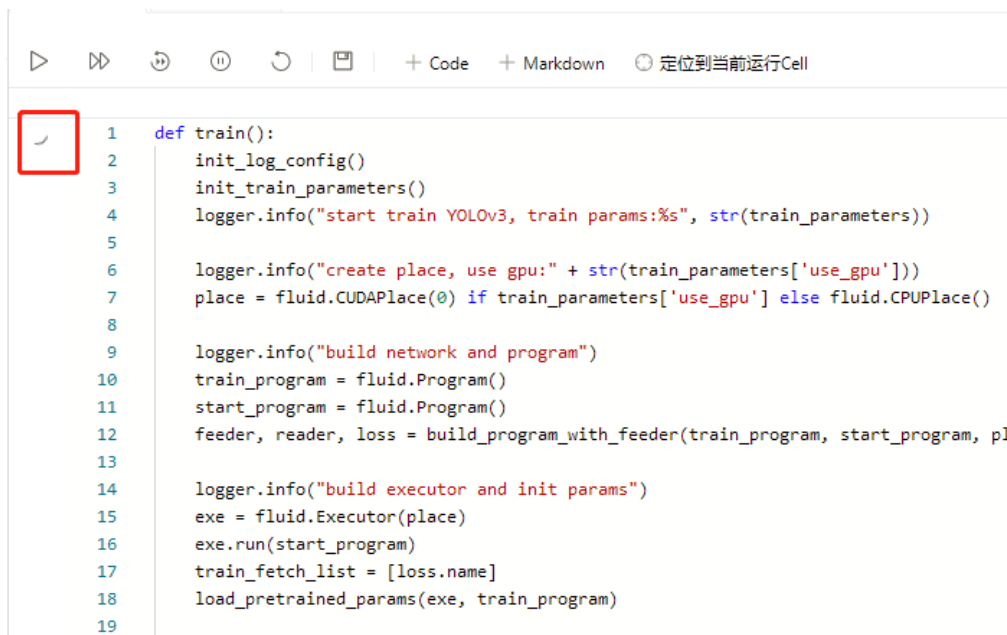


依次单击每个 cell 左上角的三角标志，运行代码；



当代码块左上角显示“旋转标志”时，代表改代码在执行中，

需耐心等待；图中代码块是训练主体部分，耗时比较长，需耐心等待

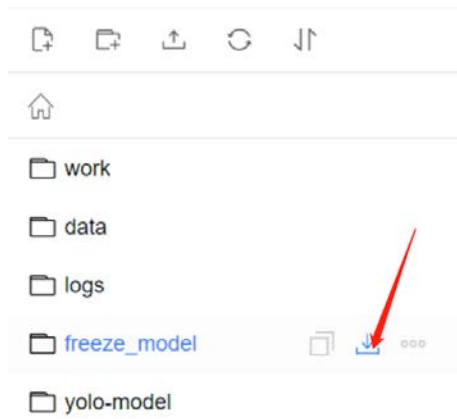


```
1 def train():
2     init_log_config()
3     init_train_parameters()
4     logger.info("start train YOLOv3, train params:%s", str(train_parameters))
5
6     logger.info("create place, use gpu:" + str(train_parameters['use_gpu']))
7     place = fluid.CUDAPlace(0) if train_parameters['use_gpu'] else fluid.CPUPlace()
8
9     logger.info("build network and program")
10    train_program = fluid.Program()
11    start_program = fluid.Program()
12    feeder, reader, loss = build_program_with_feeder(train_program, start_program, place)
13
14    logger.info("build executor and init params")
15    exe = fluid.Executor(place)
16    exe.run(start_program)
17    train_fetch_list = [loss.name]
18    load_pretrained_params(exe, train_program)
19
```

如果中途出现问题，需要单击右上角“代码执行器”中的“重启执行器”重启执行器；然后单击“编辑”中的“清除所有输出”，然后重复以上操作。

## 8.5 下载模型

代码运行后在右侧会生成 `freeze_model` 文件夹；单击“下载按钮”按钮将文件下载到车载电脑上；选中“保存文件”并单击确定；此时保存到本地；通过单击“文件夹”图标可显示当前下载的 `freeze_model.zip` 文件。

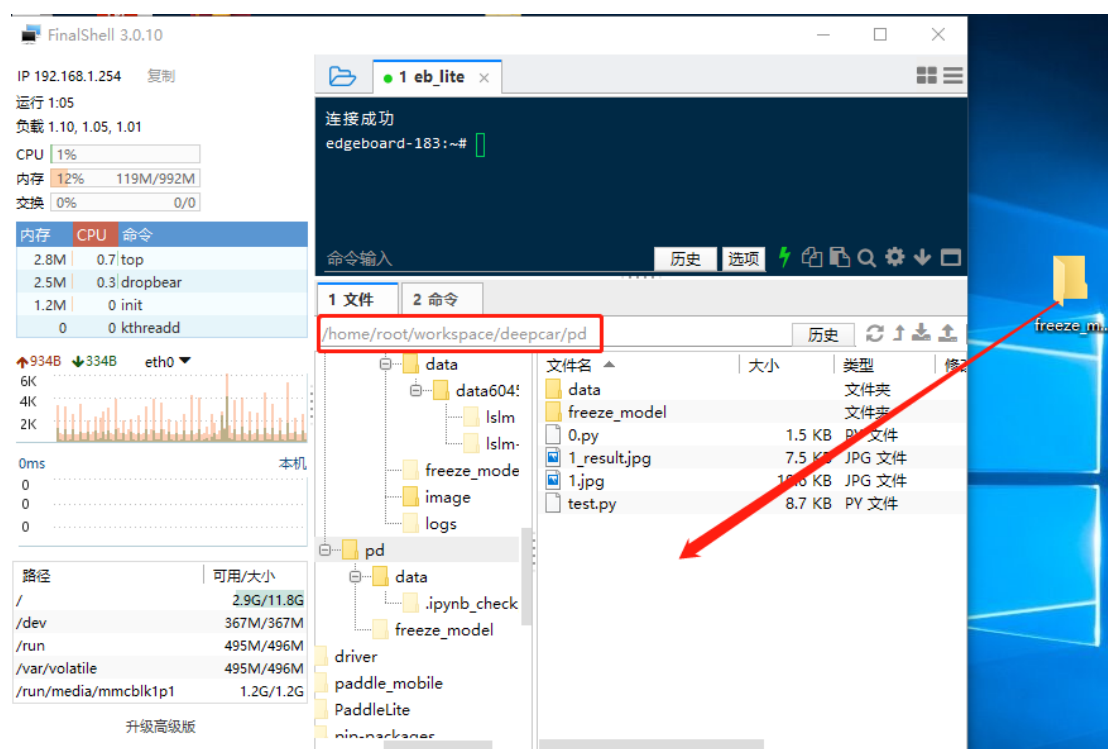


## 第九节：标志物检测\_自主移动

### 9.1 上传模型和 labellist

将从 AIstudio 训练生成的 freeze\_model.zip 文件在笔记本电脑上解压缩,并将解压缩以后的 freeze\_model 文件夹替换 edgeboard 下 /home/work/workspace/deepcar/deeplearning\_python/src 目录下的 freeze\_model 文件夹;

将 AIstudio 生成的 labellist、labellist.txt 文件替换 edgeboard 下 /home/root/workspace/deepcar/deeplearning\_python/src/data/data6045 目录下的 labellist、labellist.txt 文件



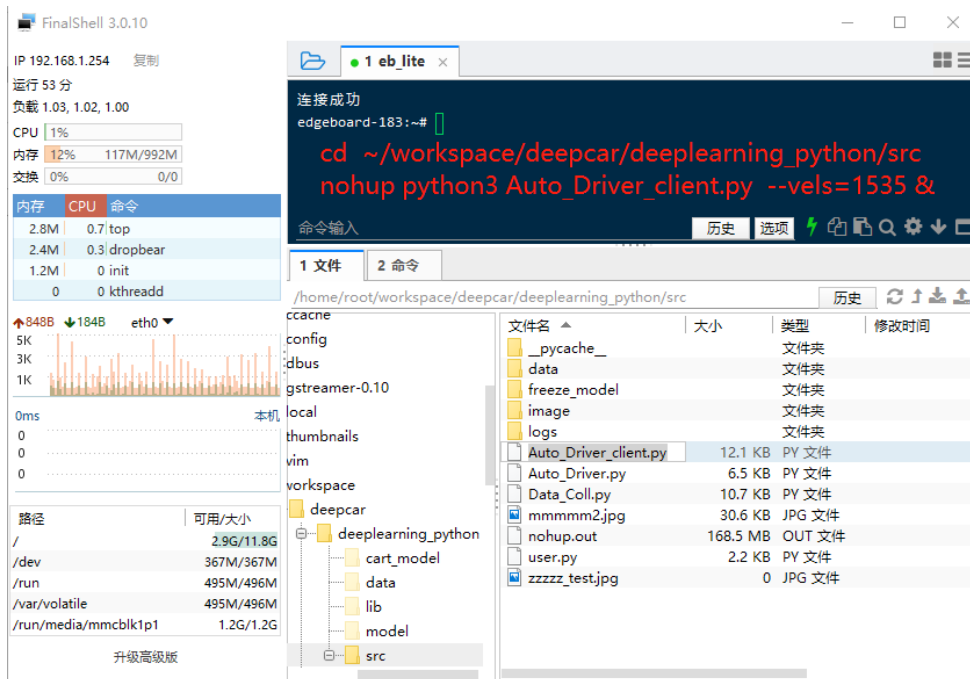
### 9.2 自主运行

确保 Edgeboard 上电, 底盘开关为关的状态下, 按照第二节

的内容连接到小车，在终端输入：

```
cd ~/workspace/deepcar/deeplearning_python/src(回车)
```

```
nohup python3 Auto_Driver_client.py --vels=1535 &
```



拔掉网线，将小车放在赛道上，开启底盘开关，小车开始自主移动；

停止时，关闭底盘开关，将 edgeboard 上的摄像头插拔一下。

Edeboard 里面~/workspace/deepcar/deeplearning\_python/src 目录下包含标志物检测后的相关操作，可以根据需要对其内容进行修改。