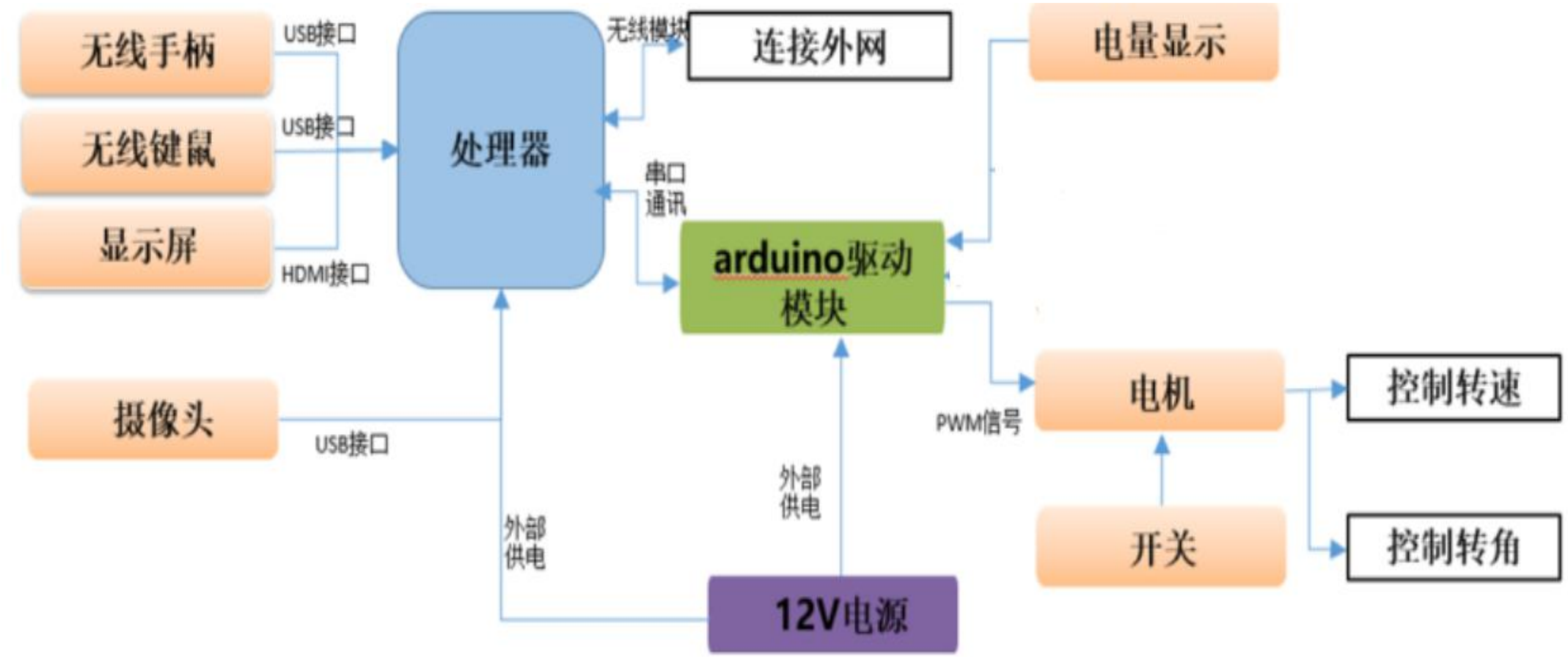
硬件介绍

◆ 硬件

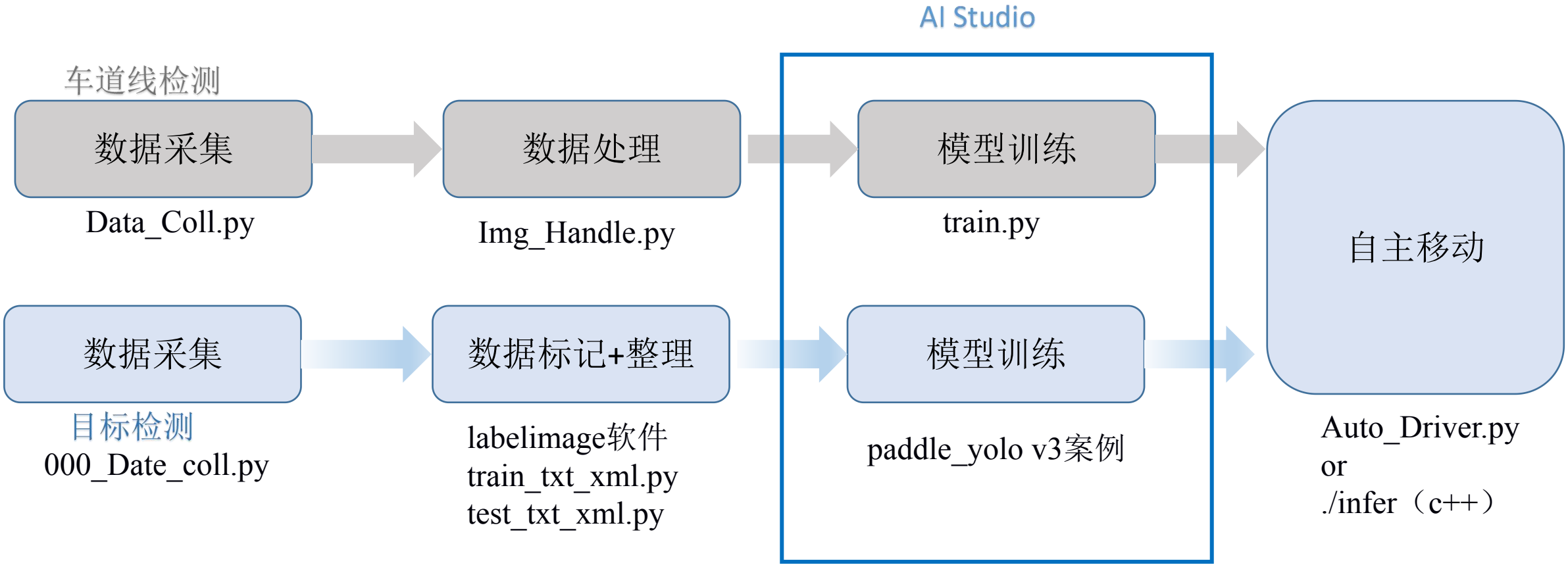
序号	类型	参数
1	底盘	四轮带编码器差速底盘。 尺寸:340mm*270mm*300mm（长宽高）
2	电机	G37-520B 编码器直流电机 12V 空载转速：178rpm
3	处理器	主控：i5 系列内存：4G
4	摄像头（2 个）	像素 720P 对角 70 度 水平 55 度 YUY2/10-15 帧/S

硬件介绍

◆ 硬件连接图


无线手柄
USB接口
无线键鼠
USB接口
显示屏
HDMI接口
处理器
无线模块
连接外网
串口
通讯
电量显示
arduino驱动
模块
摄像头
USB接口
外部
供电
外部
供电
PWM信号
电机
控制转速
开关
控制转角
12V电源

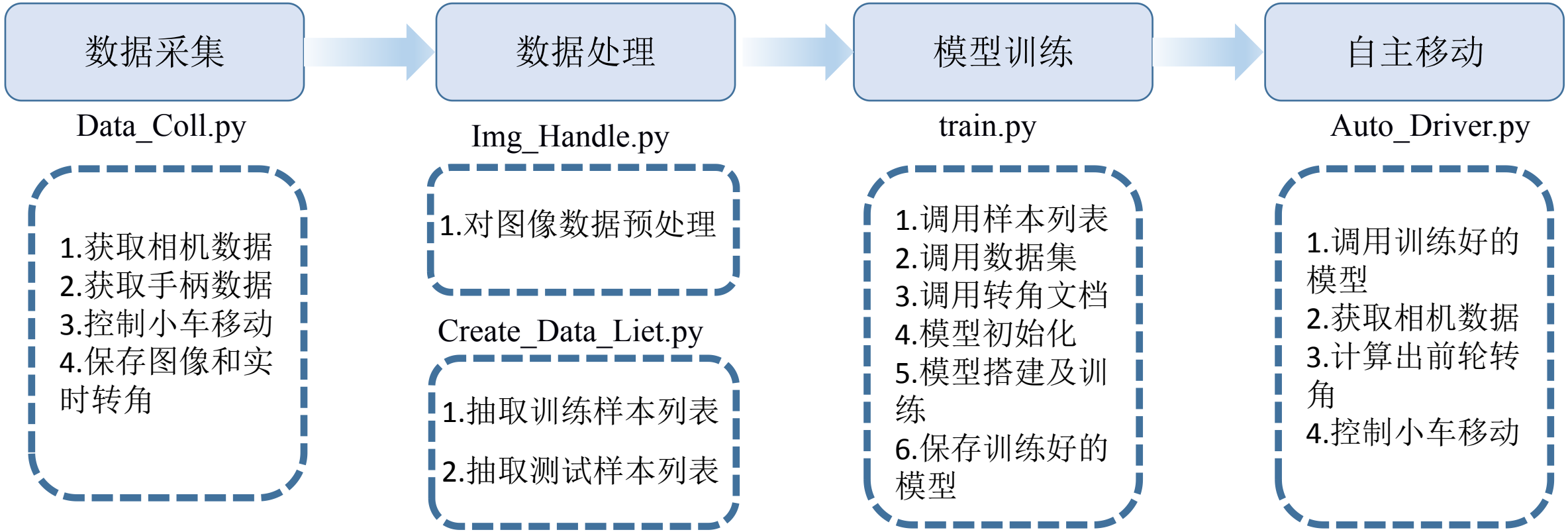
实践流程


AI Studio
车道线检测
数据采集
Data_Coll.py
数据处理
Img_Handle.py
模型训练
train.py
自主移动
Auto_Driver.py
or
./infer（c++）
数据采集
目标检测
000_Date_coll.py
数据标记+整理
labelimage软件
train_txt_xml.py
test_txt_xml.py
模型训练
paddle_yolo v3案例

实践流程

◆ 车道线检测

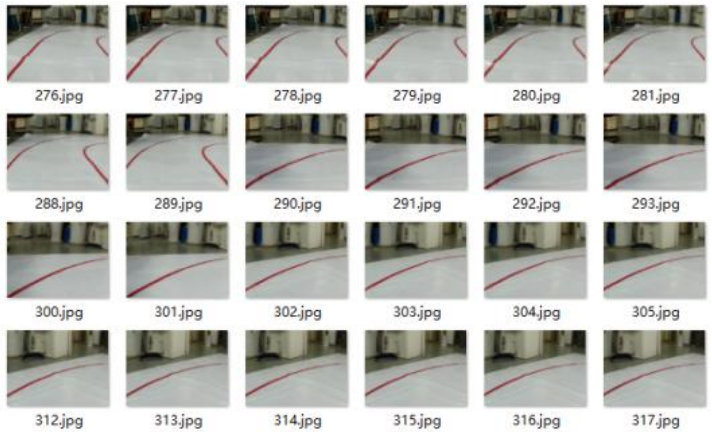
➢ 深度学习智能车主要包括数据采集，数据处理，模型训练和小车自主移动四个部分。



数据集

◆ 数据集

img文件夹

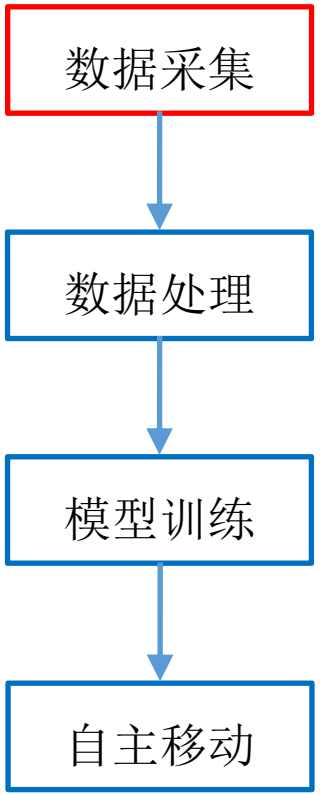


data.npy

- 数据通过手柄控制采集
- 输入为图像，目标值为前轮转角参数
- 图像大小为3通道的1280*720，数据量在8000张左右

✓训练集：7200张用于训练，每批32张图

✓测试集：800张用于测试，每批32张图



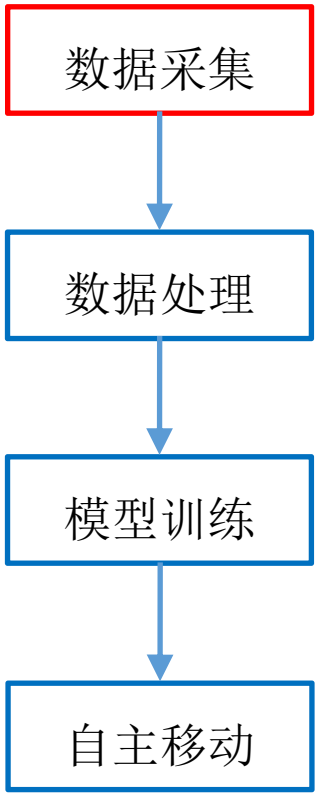
实践流程

◆ 源码解析

```
for opt_name,opt_value in opts:...
'''创建一个互斥锁，默认是没有上锁的'''
lock = multiprocessing.Manager().Lock()
def mkdir(path):...
def getvalue():...
def save_image_process(lock,n,status,start,Camera):...
def save_data_process(lock,n,data,run):...
def control_car_process(data,status,run,start):...
def txt_2_numpy():...
if __name__ == '__main__':
    Flag_save_data = multiprocessing.Value("i",False)#New img save flag
    Status = multiprocessing.Value("i",True)#Run or Stop for PS2
    START = multiprocessing.Value("i",False)#START
    RUN = multiprocessing.Value("i",True)#SHUTDOWN
    try:
        process_car = multiprocessing.Process(target=control_car_process,args=(Speed,Status,RUN,START))
        process_image = multiprocessing.Process(target=save_image_process,args=(lock,Flag_save_data,Status,START,camera,))
        process_data = multiprocessing.Process(target=save_data_process,args=(lock,Flag_save_data,Speed,RUN,))
        process_car.start()
        process_image.start()
        process_data.start()
        while(1):...
    except:...
    finally:...
```

- getvalue-→ 获取手柄数据
- save_image_process-→ 保存图像数据
- save_data_process------→ 保存转弯角度
- control_car_process----------→ 获取手柄数据
- txt_2_numpy---→将txt文档转化为npy文档

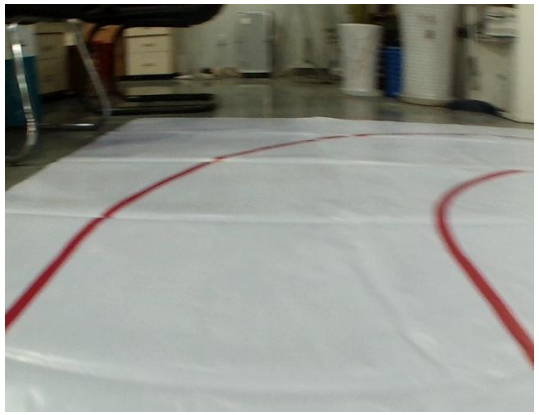
- 三个行程控制



实践流程

◆ 实践流程

- **准备数据**
  - ✓对数据进行预处理；
  - ✓将数据分割为训练集和测试集
- **配置模型**
- **训练模型**
- **模型评估**
- **模型预测**



train.list (~/ART_Deeplearning_car/data) - gedit

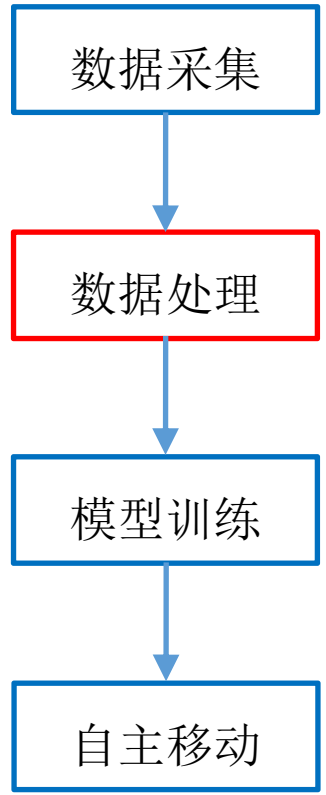
打开(O) 保存(S)

```
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/169.jpg     1054
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/182.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/358.jpg     700
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/239.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/69.jpg      700
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/103.jpg     2100
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/252.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/102.jpg     2100
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/184.jpg     1186
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/389.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/305.jpg     1912
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/132.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/255.jpg     2100
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/140.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/399.jpg     1500
```

test.list (~/ART_Deeplearning_car/data) - gedit

打开(O) 保存(S)

```
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/404.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/134.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/121.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/227.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/316.jpg     700
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/379.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/183.jpg     1219
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/333.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/271.jpg     2100
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/63.jpg      1491
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/346.jpg     2100
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/139.jpg     1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/91.jpg      1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/89.jpg      1500
/home/deep/ART_Deeplearning_car/src/../data/hsv_img/311.jpg     1500
```



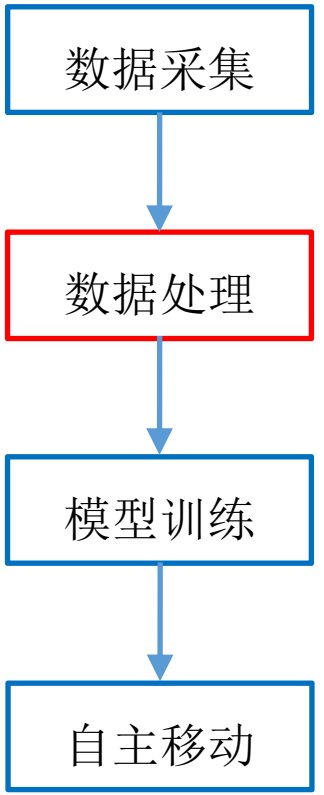
实践流程

◆ 源码解析

```python
def mkdir(path):...
def img_extract(img_path, save_path):
    img_name = os.listdir(img_path)
    lower_hsv = np.array([156, 43, 46])
    upper_hsv = np.array([180, 255, 255])
    lower_hsv1 = np.array([0, 43, 46])
    upper_hsv1 = np.array([10, 255, 255])
    for img in img_name:
        print(img)
        image = os.path.join(img_path, img)
        src = cv.imread(image)
        hsv = cv.cvtColor(src, cv.COLOR_BGR2HSV)
        mask0 = cv.inRange(hsv, lowerb=lower_hsv, upperb=upper_hsv)
        mask1 = cv.inRange(hsv, lowerb=lower_hsv1, upperb=upper_hsv1)
        mask = mask0 + mask1
        ind = int(re.findall('.+(?=.jpg)', img)[0])
        new_name = str(ind) + '.jpg'
        cv.imwrite(os.path.join(save_path, new_name), mask)
if __name__ == "__main__":
    img_path = path + "/data/"+ img_path
    save_path = path + "/data/"+ save_path
    mkdir(save_path)
    if not os.path.isdir(save_path):
        os.makedirs(save_path)
    img_extract(img_path, save_path)
```

- 依次调用数据集中的图像
- 根据hsv参数处理获取的图像
- 保存图像

颜色	Lower_hsv	Upper_hsv
黑色	[0,0,0]	[180,255,46]
红色	[156,43,46]	[180,255,255]
	[0,43,46]	[10,255,255]
橙色	[11,43,46]	[25,255,255]
黄色	[26,43,46]	[34,255,255]
绿色	[35,43,46]	[77,255,255]
蓝色	[100,43,46]	[124,255,255]
紫色	[125,43,46]	[155,255,255]



实践流程

◆ 基本架构

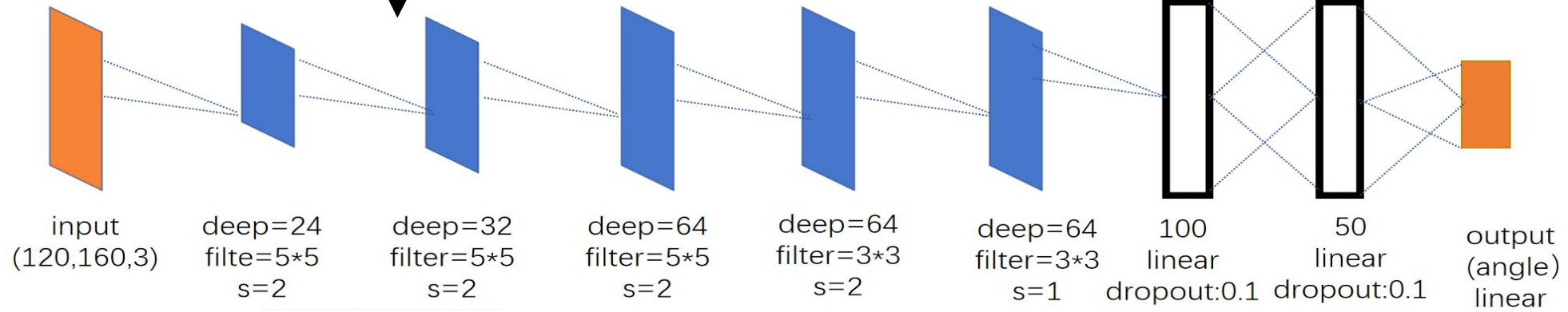
定义输入层 → 获取cnn框架 → 损失函数 → 定义优化方法 → 执行器

cnn_model.py

cost : mean-square error

optimizer :Adam
learning_rate:1e-2

CPU



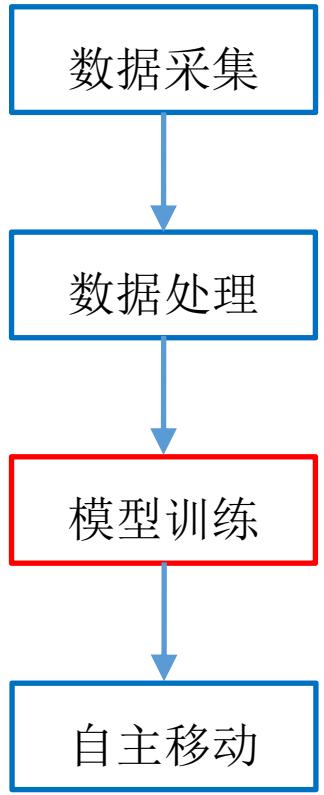
数据采集 → 数据处理 → 模型训练 → 自主移动

实践流程

◆ 源码解析

```
image = fluid.layers.data(name='image', shape=[3, crop_size, crop_size], dtype='float32')
label = fluid.layers.data(name='label', shape=[1], dtype='float32')
model = cnn_model.cnn_model(image)
cost = fluid.layers.square_error_cost(input=model, label=label)
avg_cost = fluid.layers.mean(cost)
test_program = fluid.default_main_program().clone(for_test=True)
optimizer = fluid.optimizer.AdamOptimizer(learning_rate=0.01)
opts = optimizer.minimize(avg_cost)
train_reader = paddle.batch(reader=reader.train_reader(train_list, crop_size, resize_size), batch_size=32)
test_reader = paddle.batch(reader=reader.test_reader(test_list, crop_size), batch_size=32)
place = fluid.CPUPlace()  # place = fluid.CUDAPlace(0)
exe = fluid.Executor(place)
exe.run(fluid.default_startup_program())
feeder = fluid.DataFeeder(place=place, feed_list=[image, label])
```

- 设置模型和标签
- 设置损失函数为均方误差
- 定义优化方法，学习率：0.01
- 获取训练和测试程序
- 定义执行器
- 定义输入数据维度



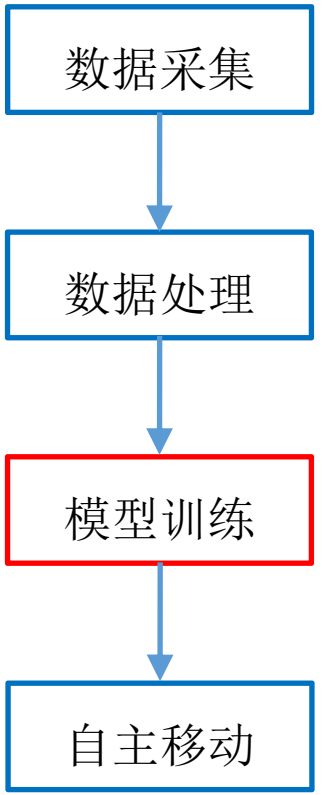
实践流程

◆ 源码解析

```
def cnn_model(image):

    conv1 = fluid.layers.conv2d(input=image, num_filters=24, filter_size=5, stride=2, act='relu')
    conv2 = fluid.layers.conv2d(input=conv1, num_filters=32, filter_size=5, stride=2, act='relu')
    conv3 = fluid.layers.conv2d(input=conv2, num_filters=64, filter_size=5, stride=2, act='relu')
    conv4 = fluid.layers.conv2d(input=conv3, num_filters=64, filter_size=3, stride=2, act='relu')
    conv5 = fluid.layers.conv2d(input=conv4, num_filters=64, filter_size=3, stride=1, act='relu')
    fc1 = fluid.layers.fc(input=conv5, size=100, act=None)
    drop_fc1 = fluid.layers.dropout(fc1, dropout_prob=0.1)
    fc2 = fluid.layers.fc(input=drop_fc1, size=50, act=None)
    drop_fc2 = fluid.layers.dropout(fc2, dropout_prob=0.1)
    predict = fluid.layers.fc(input=drop_fc2, size=1, act=None)
    return predict
```

- 5层卷积层
- 2层全连接层



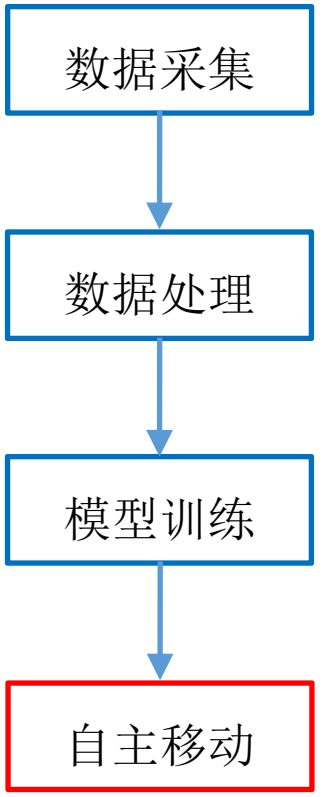
实践流程

```
for opt_name,opt_value in opts:...
def dataset(video):...
if __name__ == "__main__":
    cout = 0
    save_path  = path + "/model/" + save_path
    video = v4l2capture.Video_device(camera)
    video.set_format(1280, 720, fourcc='MJPG')
    video.create_buffers(1)
    video.queue_all_buffers()
    video.start()
    place = fluid.CPUPlace()
    exe = fluid.Executor(place)
    exe.run(fluid.default_startup_program())
    [infer_program, feeded_var_names, target_var] = fluid.io.load_inference_model(dirname=save_path, executor=exe)
    vel = int(vels)
    lib_path = path + "/lib" + "/libart_driver.so"
    so = cdll.LoadLibrary
    lib = so(lib_path)
    car = "/dev/ttyUSB0"
    if (lib.art_racecar_init(38400, car.encode("utf-8")) < 0):
        raise
        pass
    try:
        while 1:
            img = dataset(video)
            result = exe.run(program=infer_program,feed={feeded_var_names[0]: img},fetch_list=target_var)
            angle = result[0][0][0]
            a = int(angle)
            lib.send_cmd(vel, a)
            print(cout)
            cout=cout+1
            print("angle: %d, throttle: %d" % (a, vel))

    except:
        print('error')
    finally:
        lib.send_cmd(1500, 1500)
```

- 设置模型和标签
- 定义执行器

- 喂入数据
- 导出模型结果

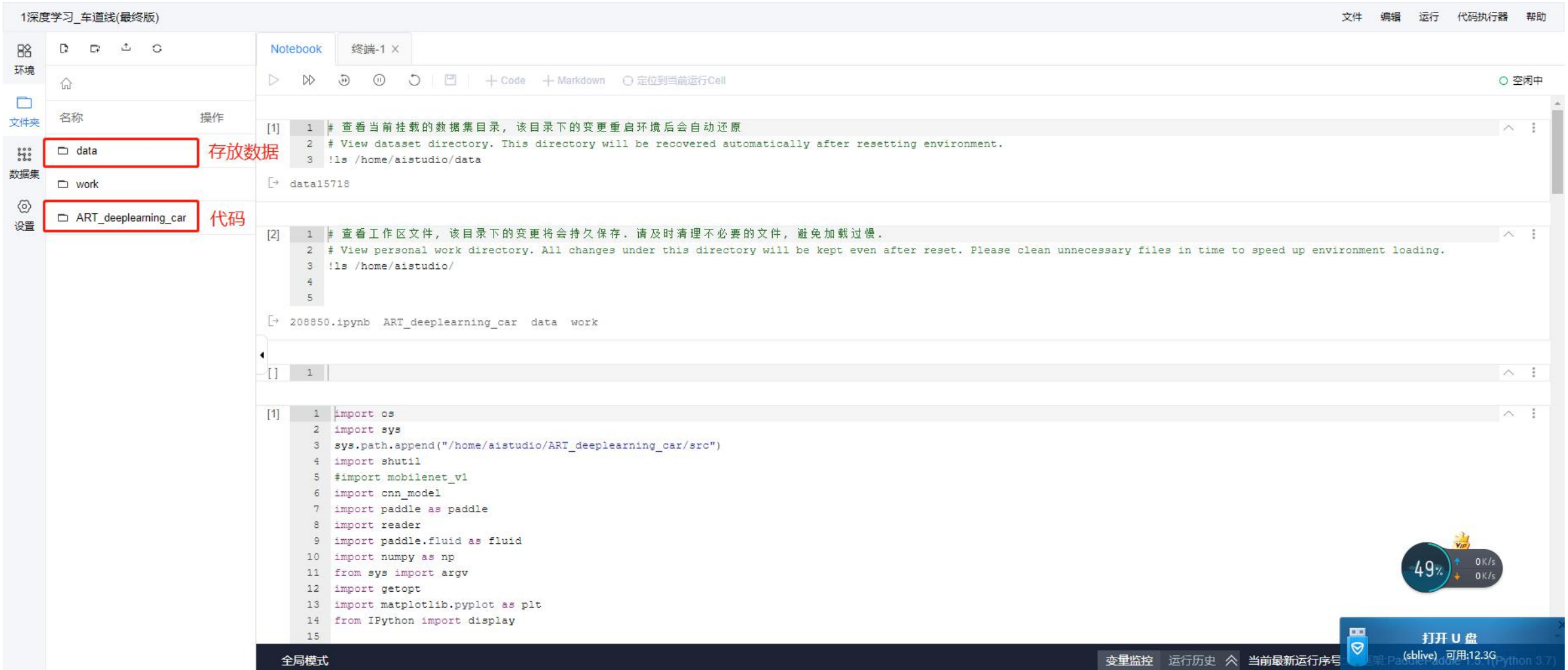


实践流程

◆ 车道线检测—数据结构

实践流程

◆ **车道线检测_AIStudio训练** https://aistudio.baidu.com/aistudio/projectdetail/170328



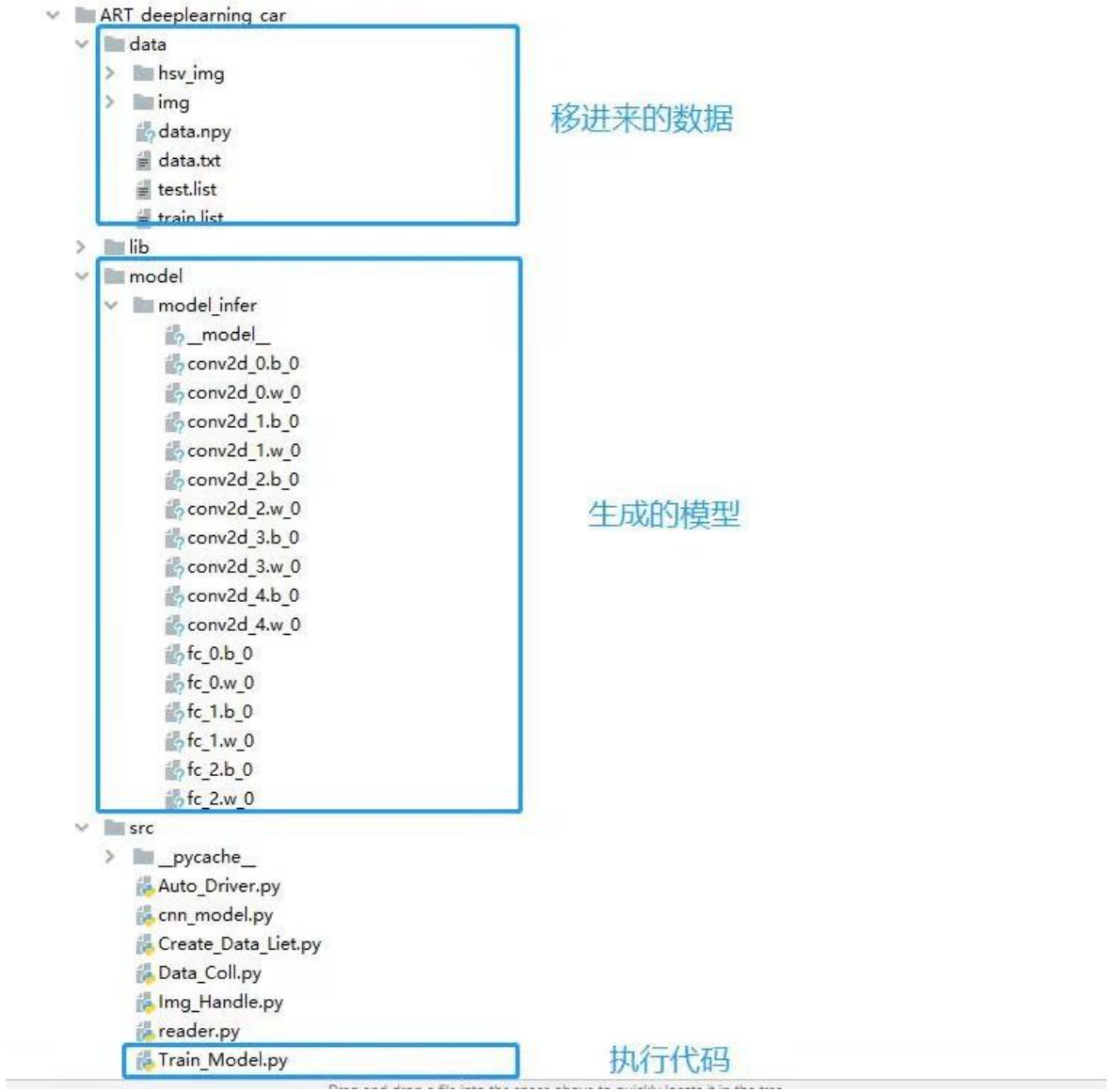
实践流程

◆ 车道线检测_AIStudio训练

1、制作数据集，上传数据集

2、打开自己的AIStudio,加载数据

3、利用cp命令，将数据拷贝到指定目录下



实践流程

车道线检测

数据采集 → 数据处理 → 模型训练 → 自主移动

Data_Coll.py　　Img_Handle.py　　train.py

目标检测

数据采集 → 数据标记+整理 → 模型训练 → 自主移动

000_Date_coll.py

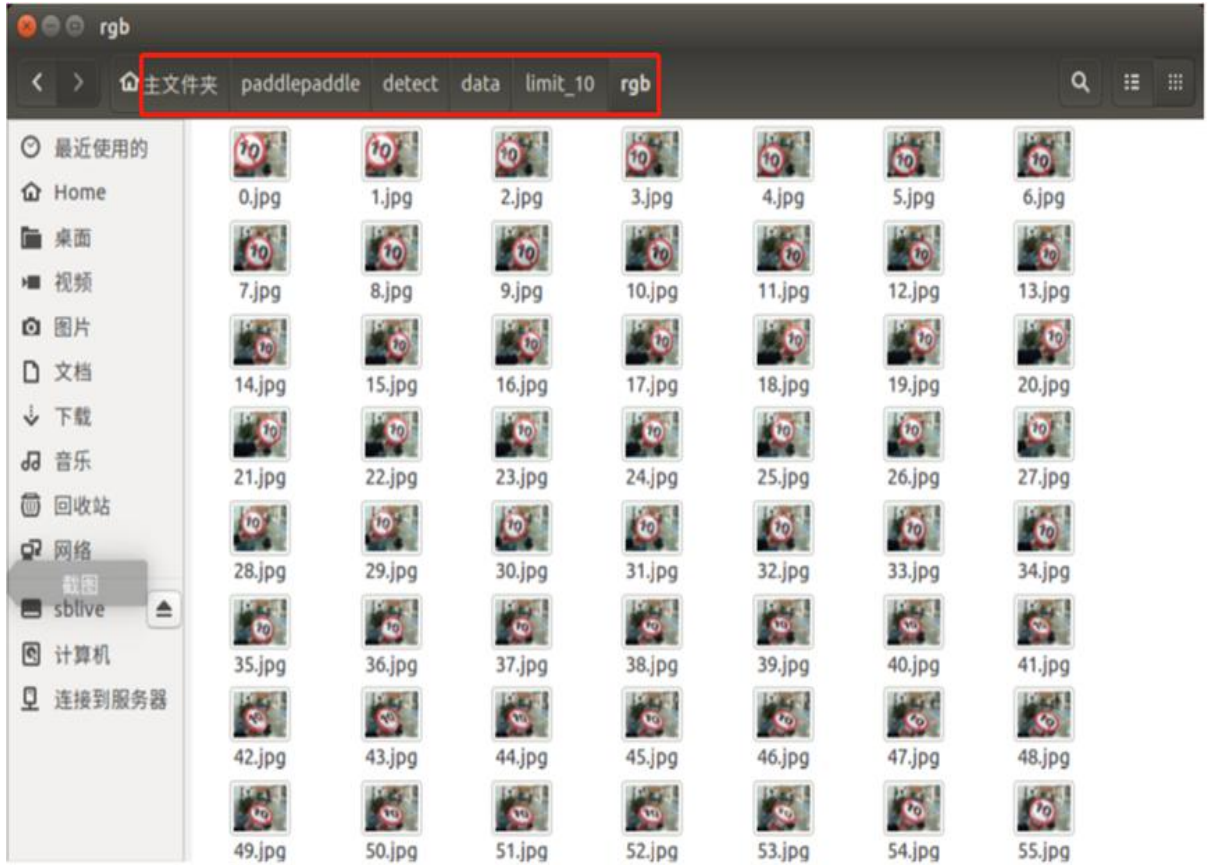
labelimage软件
train_txt_xml.py
test_txt_xml.py

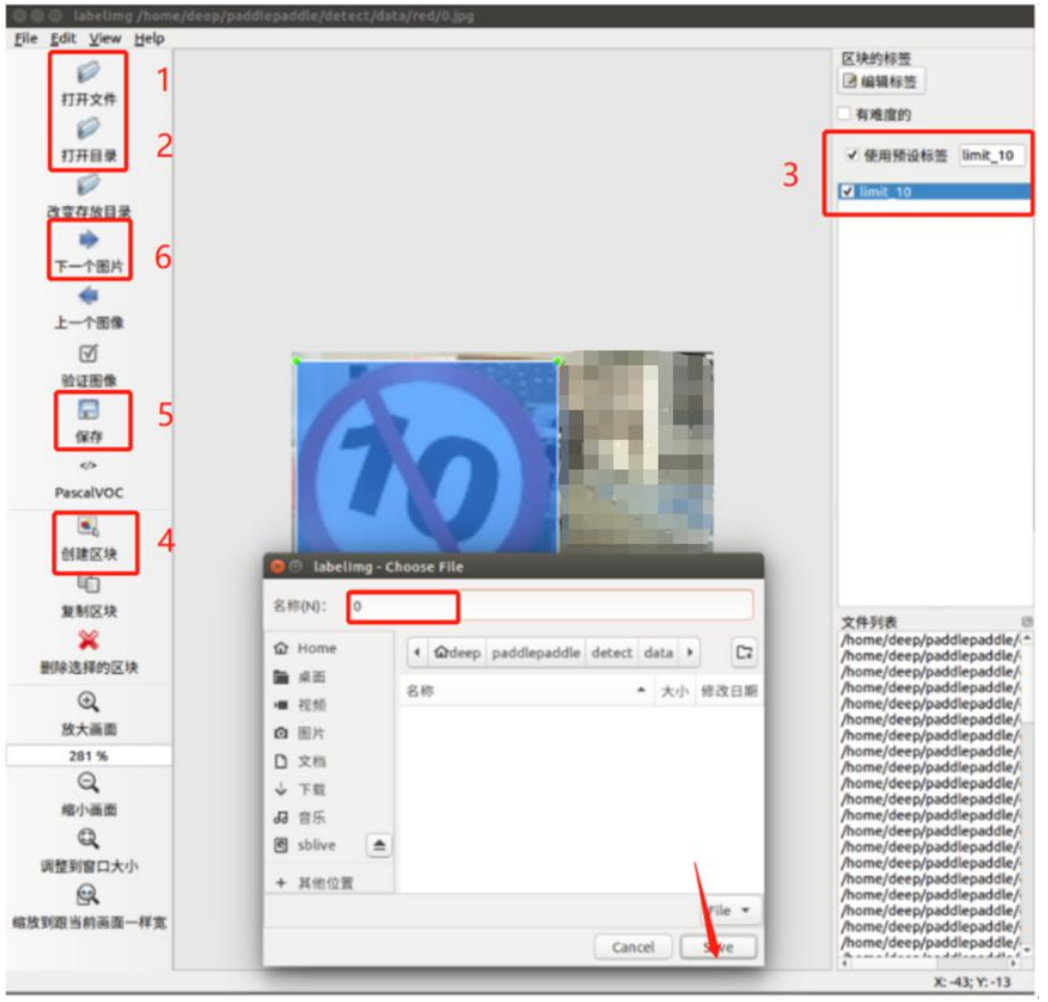
paddle_yolo v3案例

Auto_Driver.py
or
./infer（c++）

实践流程

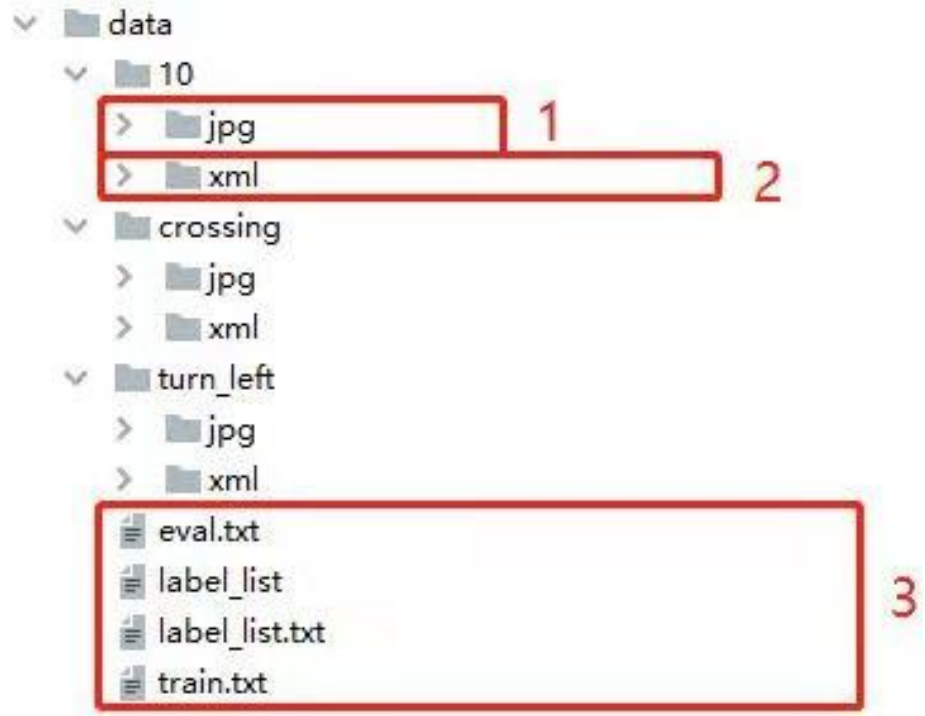
◆ 采集数据+模型标注





实践流程

◆ 目标检测

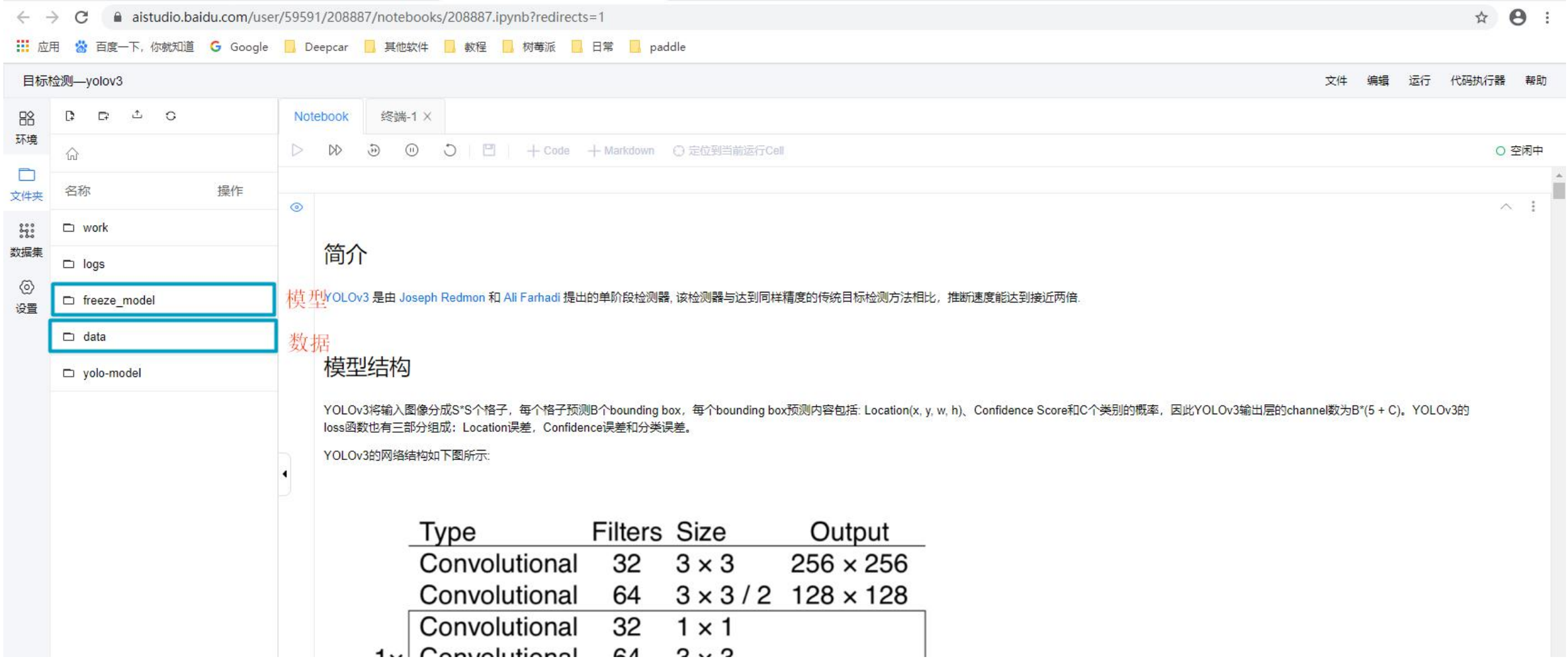


1、采集图像脚本
2、labelimage软件标注
3、脚本生成

实践流程

◆ **目标检测_AIStudio训练**

https://aistudio.baidu.com/aistudio/projectdetail/208887



实践流程

◆ 两模型同时跑——python版