# 我用飞桨做了一个无人驾驶智能车

我在观察历届智能车竞赛以及教学实验中发现，采用传统视觉算法的视觉智能车只能在特定赛道中行驶，一旦赛道环境改变，必须修改大量的代码才能运行。算法适应性差是制约智能车场景化适配的重要因素。而“AI 智能车”借助深度学习算法，通过真实数据采集到模型新训练恰恰能够解决这一问题。基于飞桨平台，我们快速研制出了“无人驾驶智能车”，已经实现了道路检测以及交通交通标识识别（红绿灯/限速牌/人行道/停车位）等功能。在本文中，我将为大家揭秘“基于飞桨的无人驾驶智能车”的具体实现过程和效果。

第一步：如图 1 所示，在智能车硬件配置上，高性能处理器是实现深度学习算法运行的必备条件，目前通用流行的高性能处理器如：intel CPU、NVDIA GPU、百度 Edgeboard 系列、NXP i.MX8 系列，在这里我们选择了基于百度 Edgeboard 系列的高性能板卡作为智能车的主处理器。

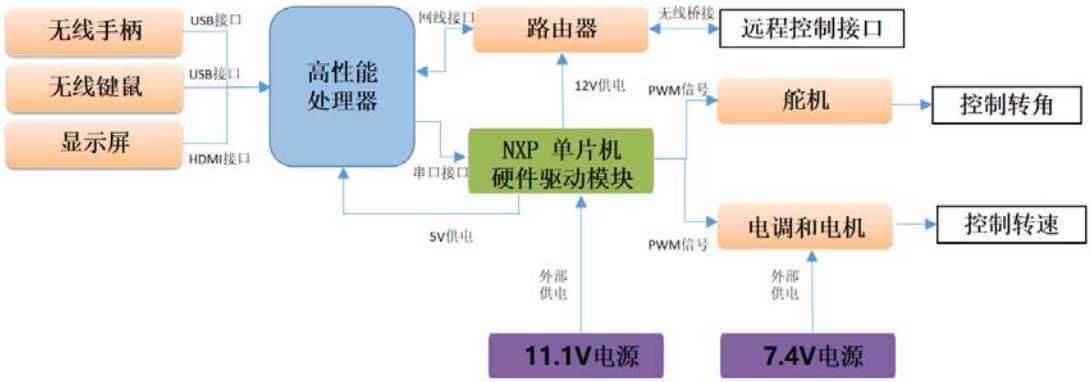
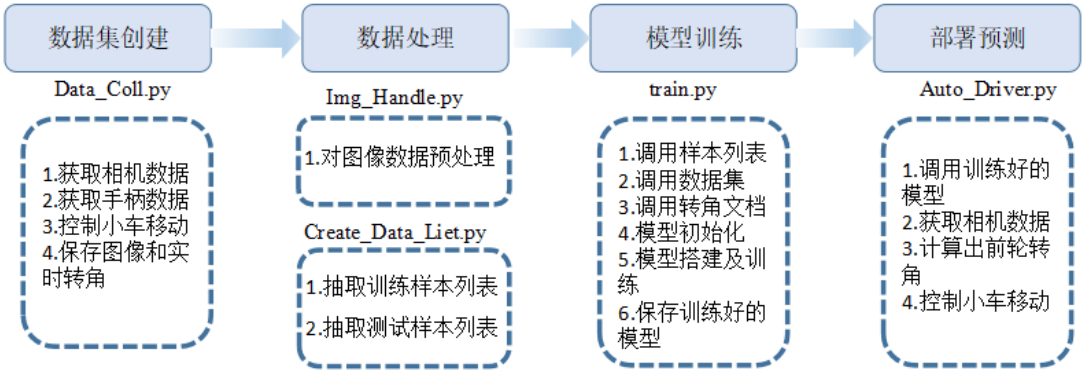



图 1.智能车硬件框架

第二步：在解决了处理器的问题之后，要实现智能车对道路和交通标识的识别就要面临深度学习框架和深度学习算法的选择。目前通用流行的深度学习框架有 Tensorflow、飞桨（PaddlePaddle）、Caffe、PyTorch 等。我们选择了飞桨，飞桨作为国产化的深度学习框架，配合一站式开发平台 AI Studio，为用户提供了优质的开发服务。进一步借助飞桨平台发布的官方支持的工业级模型以及高性能推理引擎 Paddle Lite，可以快速实现自然语言处理、计算机视觉、推荐引擎等多个领域应用的开发和部署。

在车道线识别上，我们采用的是卷积神经网络 CNN。CNN 在图像领域有重要的应用价值，结

合实际测试经验，我们的智能车采用了 5 个卷积层加 2 个全连接层来构成车道线网络模型，智能车整体工作流程可分为数据集创建、数据处理、模型训练和部署预测四步。




## 一、数据采集

通过手柄遥控智能车在赛道内按照适当速度运行，记录过程中的每一帧图像及对应的转弯角度。采集的图像如图 2 所示。

图 2 采集的图像集

代码解析：

运行的程序包括三个进程分别控制：获取手柄数据、保存图像数据以及保存转弯数据；通过创建一个互斥锁，使得图像数据和角度信息一一对应保存下来；最后将转弯数据转成 npy 文档，便于下一步的调用。

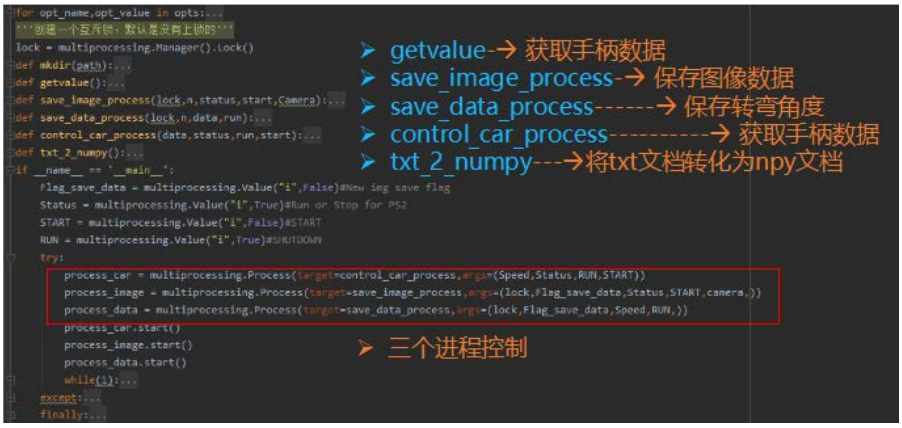



图 3 数据采集代码解析

## 二、图像预处理

对获取的图像信息进行预处理，提取出图像中的赛道，并保存处理后的图像。处理后的图像如图 4 所示。

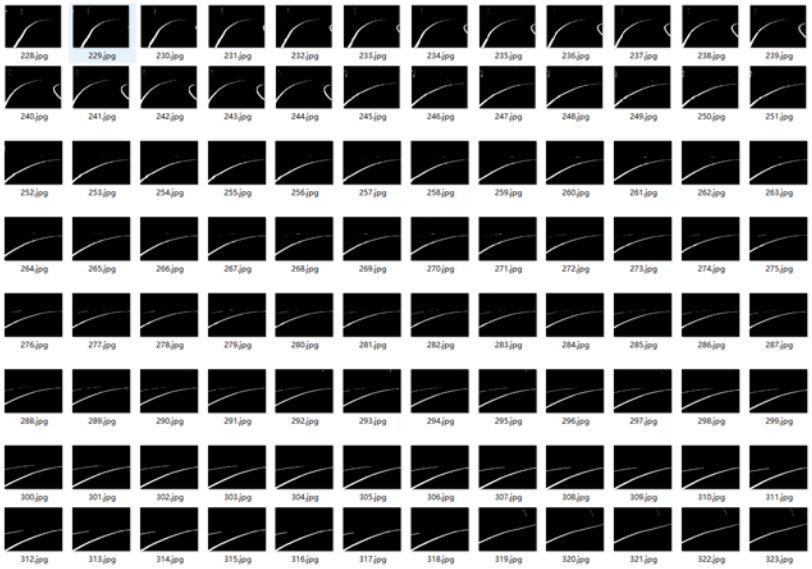


图 4 处理后的数据

代码解析：

依次读取数据集中的图像，根据颜色阈值限定对每一张图像颜色信息进行二值化处理并另存到特定目录下，如图 5 所示。

```
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] |

图 5 数据处理代码解析

## 三、模型训练

以卷积神经网络为主体搭建深度学习网络框架，并将图像及转弯角度信息放入模型中进行训练，最终得到训练后的模型，搭建的模型如图 6 所示。

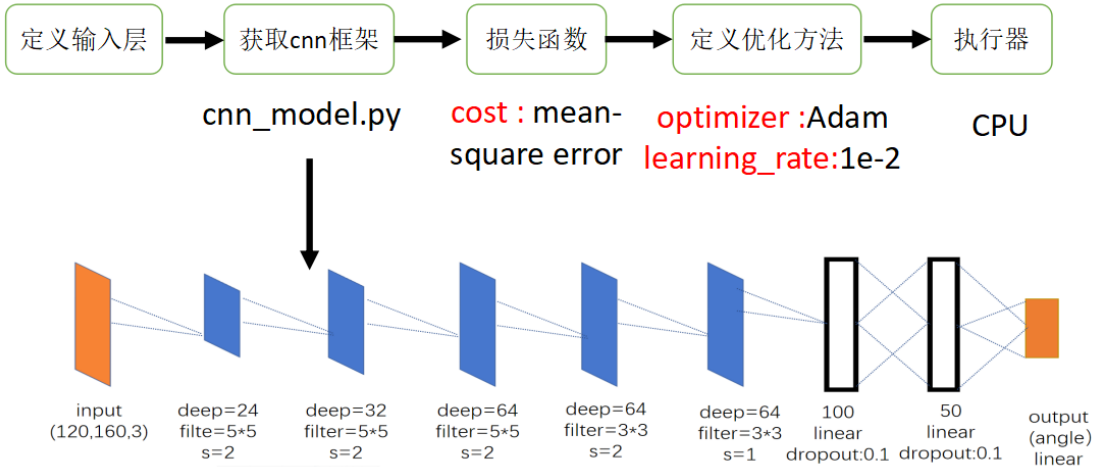



图 6 模型训练

本过程通过飞桨框架，在其中定义损失函数、优化方案、层数、卷积核大小等参数。具体代码如图 7 所示。

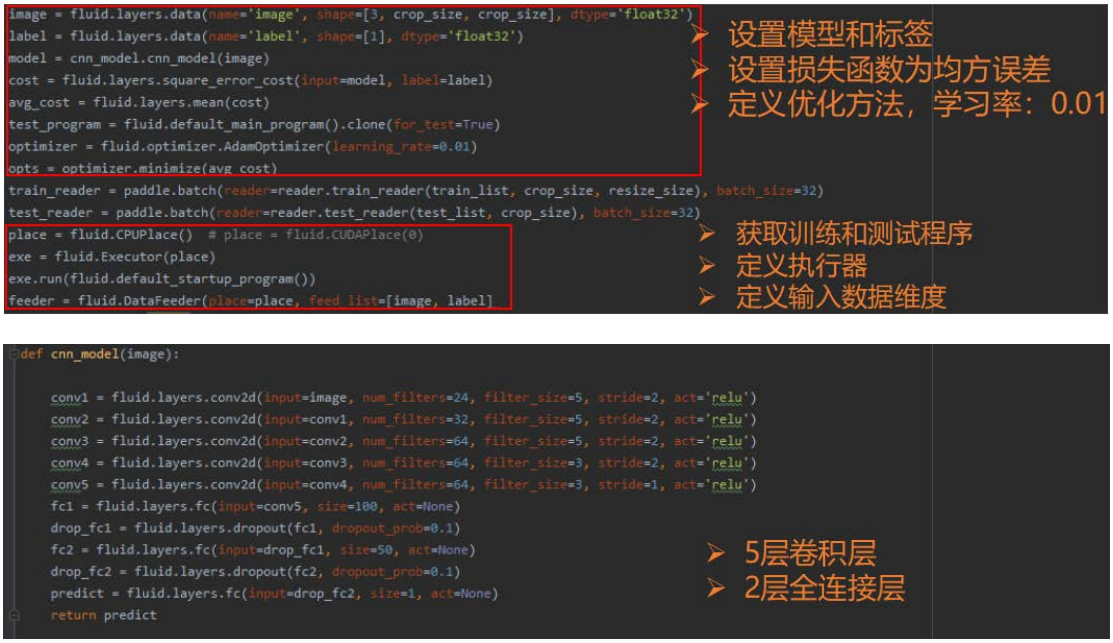



图 7 模型训练代码解析

## 四、部署预测

将 AI Studio 得到的飞桨 CNN 模型下载到终端，并通过局域网传入智能车的主处理器上，在智能车主处理器上利用 Paddle Lite 实现模型调用。

然后把智能车放置在赛道中，智能车通过调用训练好的模型，根据实时采集的图像信息，输出对应的转弯角度，进而实现自主运行，具体实现代码如图 8 所示。

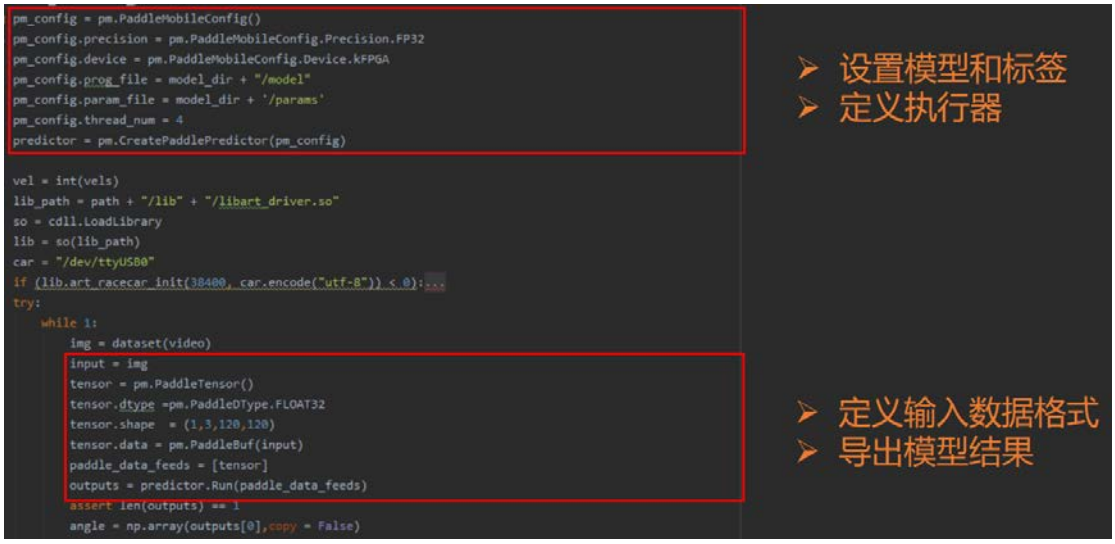



图 8 预测部署代码解析

详情可查看 GitHub 项目地址：

https://github.com/PaddlePaddle/Paddle-Lite/tree/develop/mobile

运行的视频如下：

（GIF2）