上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

搭建一个简单网络识别交通标志牌

2024 年 10 月

任务简介

在本演示中搭建了一个简单的两层卷积网络，用于对交通标志牌的识别

在自建数据集上，这个简单的神经网络可达到99.4%的识别正确率，相较于SVM的95.3%的正确率，卷积网络在视觉任务上更具优势。

SJTU

01

构建数据集

SJTU

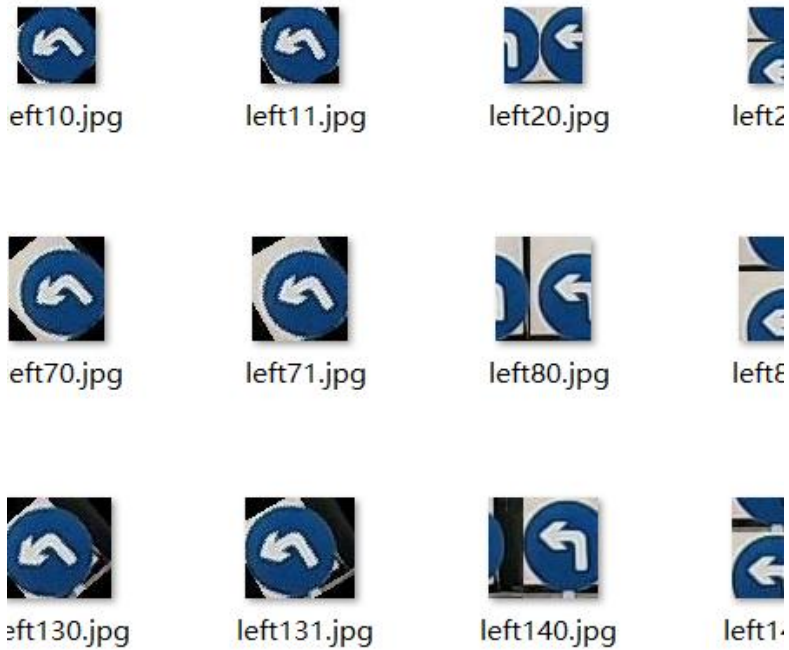
采集的方式

为了可以方便的进行数据的采集，可以采用树莓派摄像头记录视频的形式，再对视频进行抽帧、裁剪以获得标志牌不同光照、不同大小、不同背景下的数据

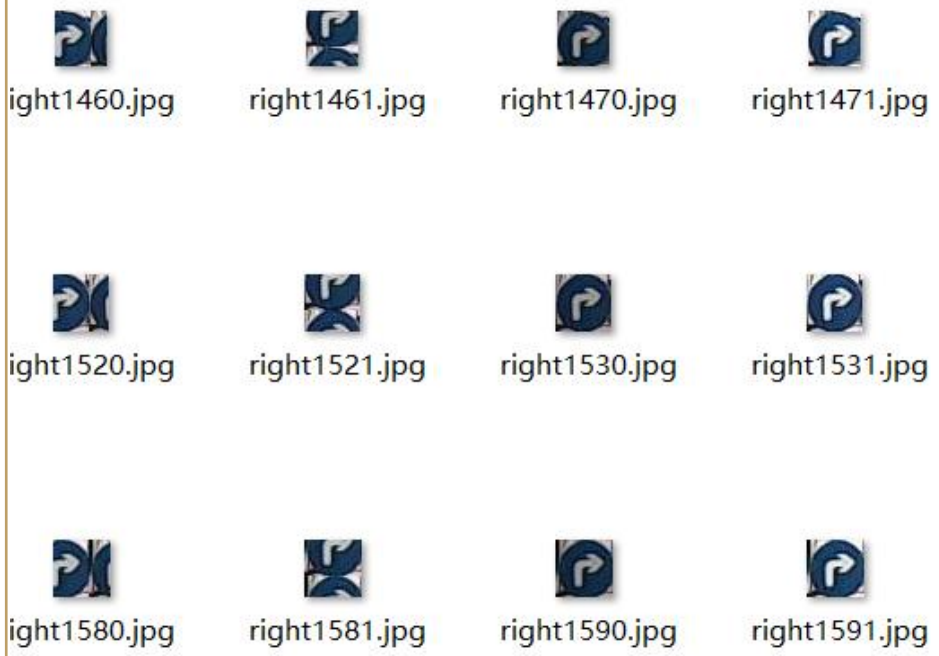
此外还可以对原始收集到的数据进行仿射变换来达到数据增广的目的，确保网络可以学习到更加鲁棒的特征

数据集

左标志牌



右标志牌



SJTU

数据集的组织

可以使用pytorch中内置的数据来进行Dataset的实现，也可以自定义Dataset（详见 https://pytorch.org/tutorials/beginner/basics/intro.html ），如果使用内置的Dataset类需要将文件夹的组织形式和类的说明严格保存一致

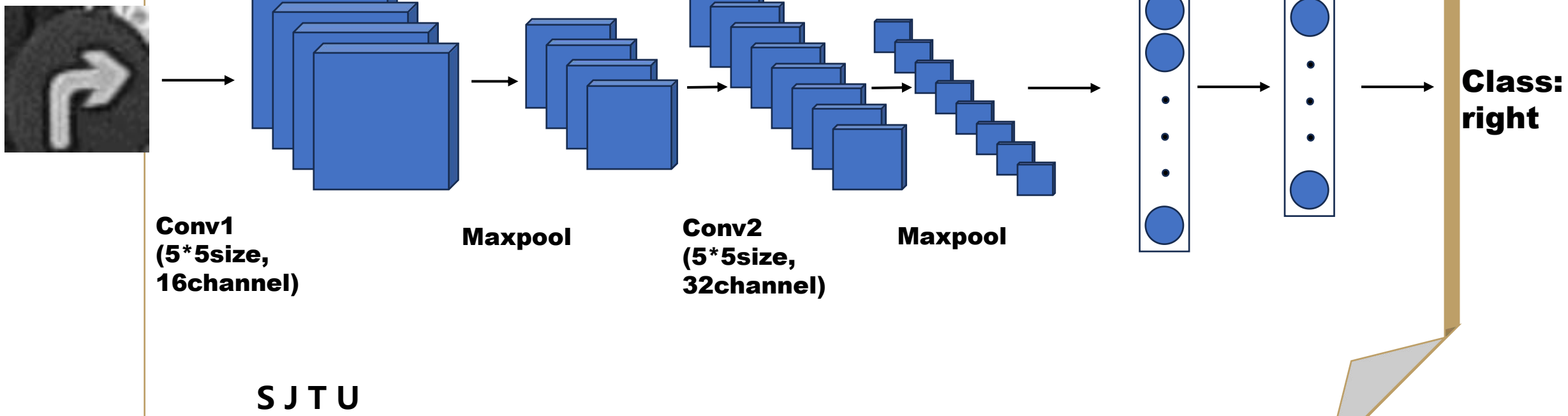
```
# 定义dataset
from torchvision import datasets

train_data=datasets.ImageFolder(root='./raw_data/train',transform=data_transform)
test_data=datasets.ImageFolder(root='./raw_data/val',transform=data_transform)
print(train_data.class_to_idx)  # .class_to_idx属性以了解label和文件夹名的映射关系

# dataloader封装
train_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True, num_workers=num_workers)
test_loader = DataLoader(test_data, batch_size=batch_size, shuffle=False, num_workers=num_workers)
```

02

网络搭建和训练

SJTU

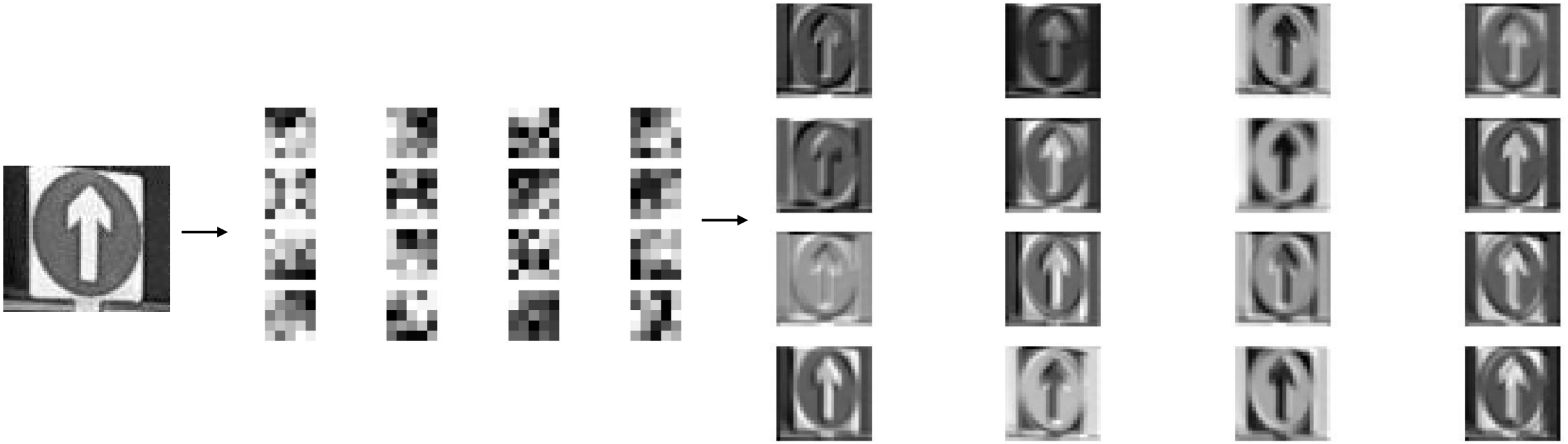
网络结构


28 * 28 * 1
24 * 24 * 16
12 * 12 * 16
8 * 8 * 32
4 * 4 * 32
512
100
Class:
right
Conv1
(5*5size,
16channel)
Maxpool
Conv2
(5*5size,
32channel)
Maxpool

S J T U

可视化

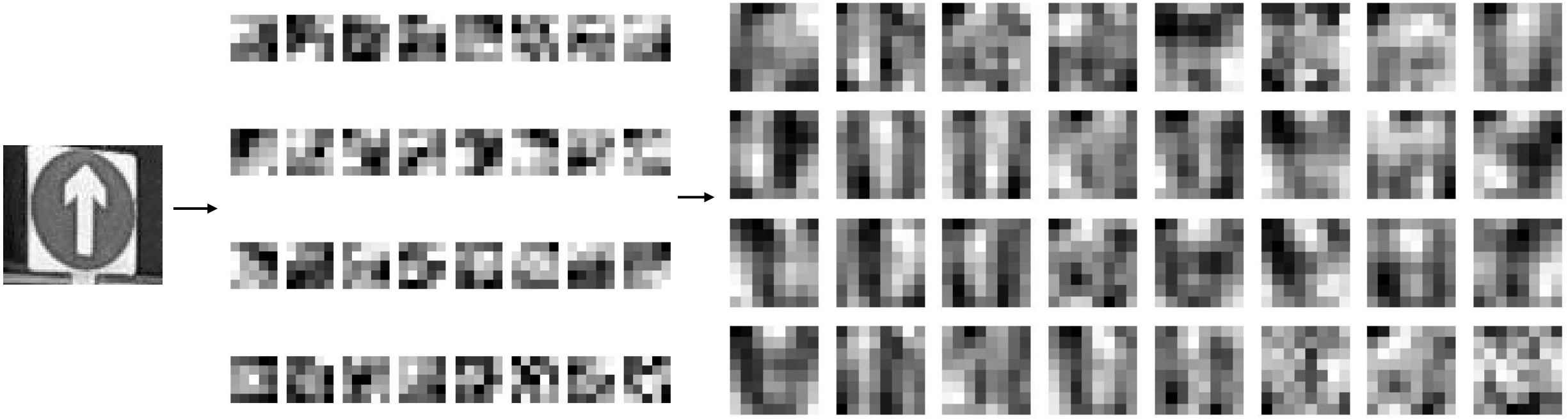
为了可以更直观地感受卷积，将第一层卷积中的卷积核以及卷积后的特征图可视化如下：



输入图像　　16通道的卷积核　　经过卷积操作后的图像

可视化

为了可以更直观地感受卷积，将第一层卷积中的卷积核以及卷积后的特征图可视化如下：



输入图像 **16通道的卷积核** **经过卷积操作后的图像**

神经网络的搭建

```python
# 定义model
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 16, 3, padding=1),
            nn.Conv2d(16, 16, 5),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2),
            nn.Dropout(0.3),
            nn.Conv2d(16, 32, 5),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2),
            nn.Dropout(0.3)
        )
        self.fc = nn.Sequential(
            nn.Linear(32*4*4, 100),
            nn.ReLU(),
            nn.Linear(100, 4)
        )

    def forward(self, x):
        x = self.conv(x)
        x = x.view(-1, 32*4*4)
        x = self.fc(x)
        # x = nn.functional.normalize(x)
        return x
```

使用pytorch中的nn.Module类
可以像搭积木一样轻松的实现
上述提到的网络：
在init方法中定义网络的组件
在forward方法中定义好数据输入到输出流程

神经网络的推理

初始化网络并加载权重

```
net = Net()
net.load_state_dict(torch.load('four.pth',map_location='cpu'))
net.eval()
```

选取概率最大的类别作为最后识别结果

```
with torch.no_grad():
    outputs = net(roi)
    predict = torch.max(outputs, dim=1)[1].numpy()
max_label=int(predict)
```

SJTU

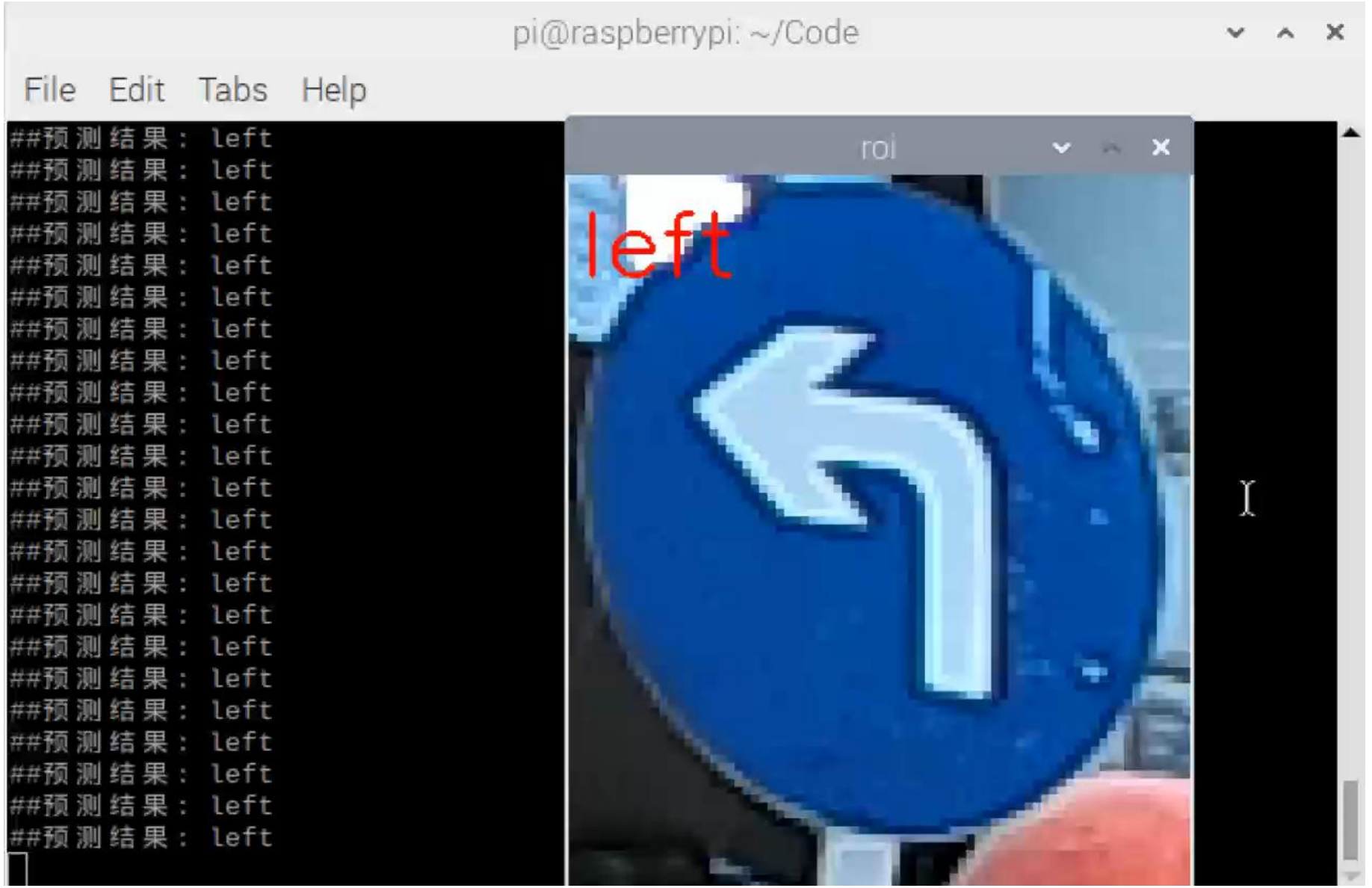
上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

识别结果展示

SJTU

03

识别结果展示



谢谢！