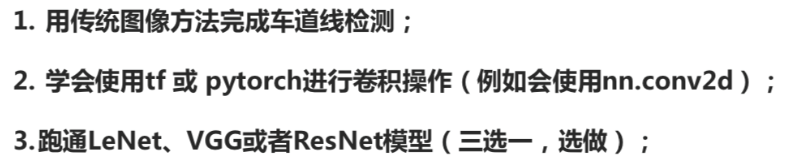
# **名企班项目week2作业**

# 一.作业要求



# 二.作业完成说明

1.用用传统方法完成车道线检测

Week1已完成和提交过。

1. 补充完成作业的文件：

[/home/stu\_15327323676/hub\_dir/CCV4名企项目\_肖模艳](https://train.kaikeba.com/user/stu_15327323676/notebooks/fm_cv1/threshold2binaryImage.ipynb)

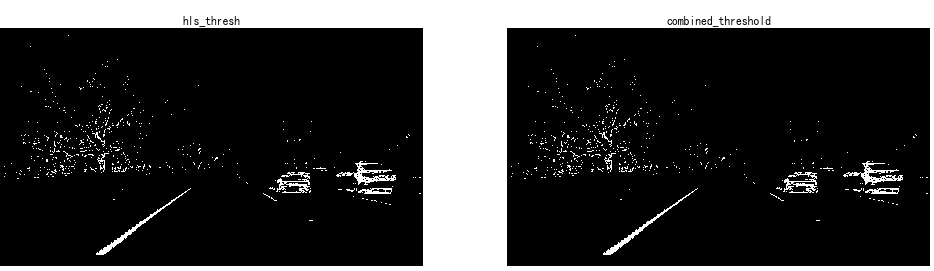
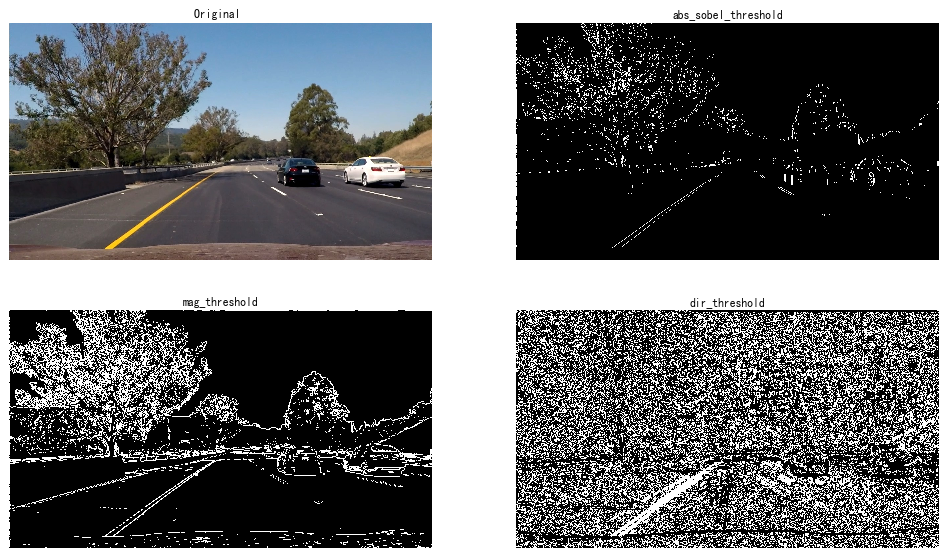
[/threshold2binaryImage.ipynb](https://train.kaikeba.com/user/stu_15327323676/notebooks/fm_cv1/threshold2binaryImage.ipynb)

（2）填空

空较多，见源程序threshold2binaryImage.ipynb

（3）补充完成后的运行结果：

sobel水平或垂直方向，sobel梯度幅值，梯度，HLS空间S饱和度等阈值处理，以及上述多种方法组合的l5种方法边缘检测结果对比：



车道线检测结果：



1. 学会用tf或pythorch进行卷积操作（例如，会用nn.conv2d）

部分关键代码说明：

(1)简化常见模型，定义一个卷积操作类。为显示图像特征，对卷积结果求绝对值。

class test(nn.Module):

def \_\_init\_\_(self):

super(test, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 16, 5)

def forward(self, x):

x = abs(self.conv1(x))

return x

1. 输入准备，使输入信息与模型输入匹配

img = cv2.imread("../testImage/straight\_lines1.jpg")

img=np.transpose(img, (2, 0, 1))

img=torch.tensor(img)

img = torch.unsqueeze(img, dim=0).float()

1. 模型实例化，传入数据，完成卷积操作

model = test()

out\_put=model(img)

1. 卷积后的特征图像显示

for feature\_map in out\_put:

# [N, C, H, W] -> [C, H, W]

im = np.squeeze(feature\_map.detach().numpy())

# [C, H, W] -> [H, W, C]

im = np.transpose(im, [1, 2, 0])

# show top 16 feature maps

plt.figure(figsize = (120, 80))

for i in range(16):

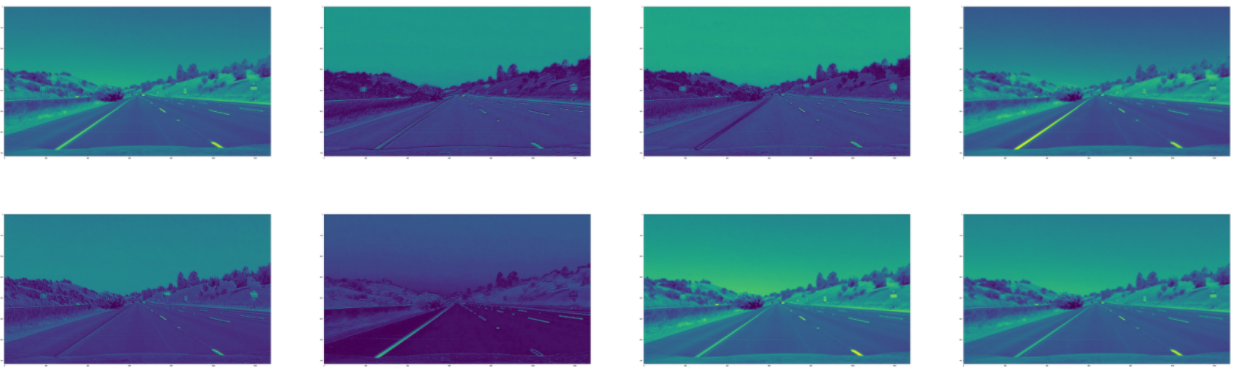
ax = plt.subplot(4, 4, i+1)

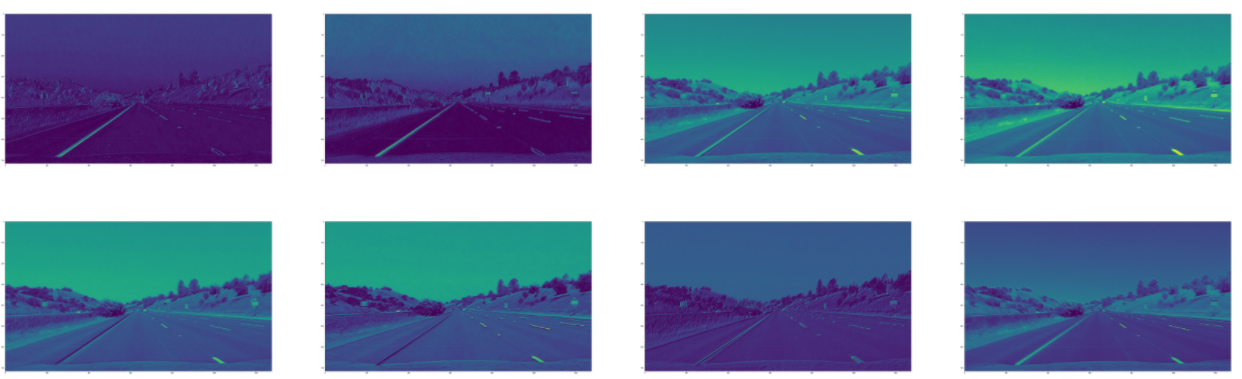
# [H, W, C]

plt.imshow(im[:, :, i])#这里为效果好一点，没设置为灰度显示

plt.show()

特征图：





本作业文件名：**pytorch\_conv2\_test.ipynb**

文件位置：<https://train.kaikeba.com/user/stu_15327323676/notebooks/CCV4名企项目_肖模艳/week2/week2/pytorch_conv2_test.ipynb>

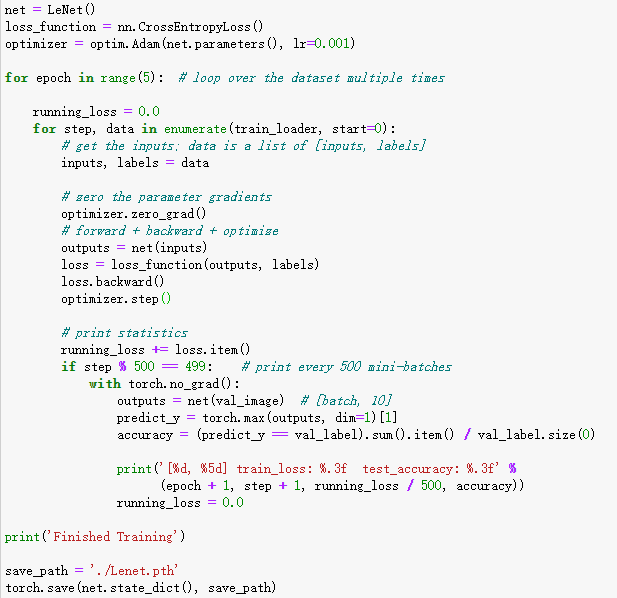


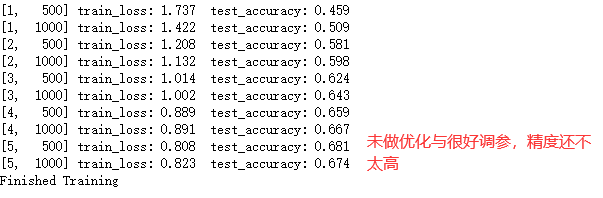
1. 跑通LeNet、VGG或ResNet(选做）

跑通pytorch 官方例子LeNet cifar10图像分类









**跑完后加载模型预测：**

classes = ('plane', 'car', 'bird', 'cat',

'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

net = LeNet()

net.load\_state\_dict(torch.load('Lenet.pth'))

im = Image.open('test.jpg')

plt.figure(figsize = (10, 8))

plt.title('预测的图片')

plt.imshow(im )

plt.axis('off')

im = transform(im) # [C, H, W]

im = torch.unsqueeze(im, dim=0) # [N, C, H, W]

with torch.no\_grad():

outputs = net(im)

predict = torch.max(outputs, dim=1)[1].data.numpy()

print(f'预测的结果：{classes[int(predict)]}')



附：

