**《计算机视觉》实验报告**

**姓名：刘远航 学号：22121883**

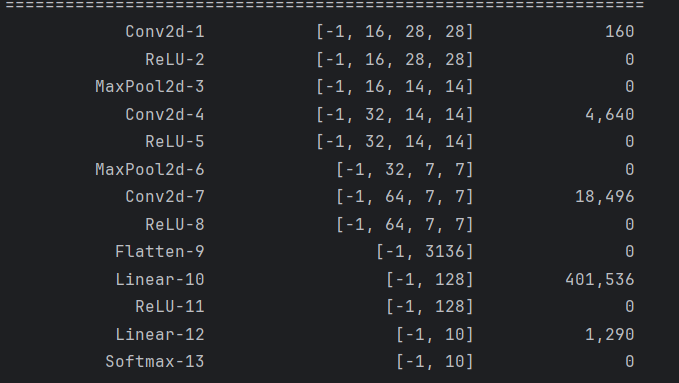
实验10

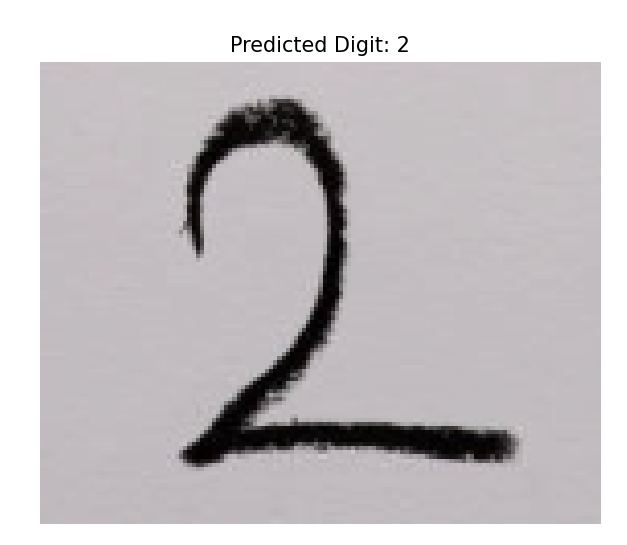
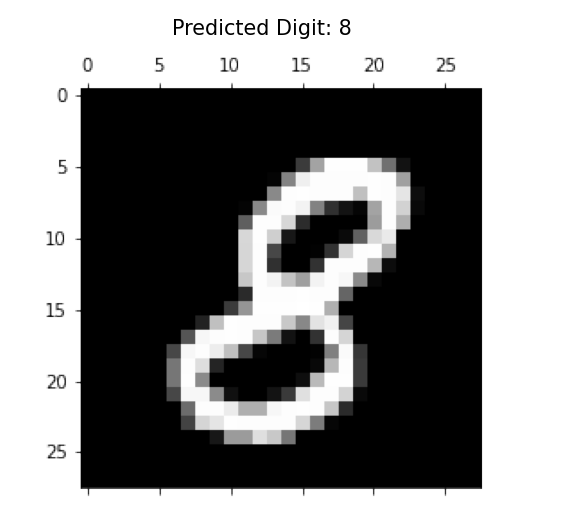
1. **任务1** 
   1. **核心代码：**

# 定义CNN模型  
class Net(torch.nn.Module):  
 def \_\_init\_\_(self):  
 super(Net, self).\_\_init\_\_()  
 self.model = torch.nn.Sequential(  
 # 图像大小为 28x28  
 torch.nn.Conv2d(in\_channels=1, out\_channels=16, kernel\_size=3, stride=1, padding=1),  
 torch.nn.ReLU(),  
 torch.nn.MaxPool2d(kernel\_size=2, stride=2),  
 # 图像大小为 14x14  
 torch.nn.Conv2d(in\_channels=16, out\_channels=32, kernel\_size=3, stride=1, padding=1),  
 torch.nn.ReLU(),  
 torch.nn.MaxPool2d(kernel\_size=2, stride=2),  
 # 图像大小为 7x7  
 torch.nn.Conv2d(in\_channels=32, out\_channels=64, kernel\_size=3, stride=1, padding=1),  
 torch.nn.ReLU(),  
 torch.nn.Flatten(),  
 torch.nn.Linear(in\_features=7 \* 7 \* 64, out\_features=128),  
 torch.nn.ReLU(),  
 torch.nn.Linear(in\_features=128, out\_features=10),  
 torch.nn.Softmax(dim=1)  
 )  
 def forward(self, input):  
 output = self.model(input)  
 return output  
  
lossF = torch.nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(net.parameters())  
history = {'Test Loss': [], 'Test Accuracy': []}  
for epoch in range(1, EPOCHS + 1):  
 processBar = tqdm(trainDataLoader, unit='step')  
 net.train(True)  
 for step, (trainImgs, labels) in enumerate(processBar):  
 trainImgs = trainImgs.to(device)  
 labels = labels.to(device)  
  
 net.zero\_grad()  
 outputs = net(trainImgs)  
 loss = lossF(outputs, labels)  
 predictions = torch.argmax(outputs, dim=1)  
 accuracy = torch.sum(predictions == labels).item() / labels.shape[0]  
 loss.backward()  
 optimizer.step()  
 processBar.set\_description(f"[{epoch}/{EPOCHS}] Loss: {loss.item():.4f}, Acc: {accuracy:.4f}")  
 if step == len(processBar) - 1:  
 correct, totalLoss = 0, 0  
 net.train(False)  
 with torch.no\_grad():  
 for testImgs, labels in testDataLoader:  
 testImgs = testImgs.to(device)  
 labels = labels.to(device)  
 outputs = net(testImgs)  
 loss = lossF(outputs, labels)  
 predictions = torch.argmax(outputs, dim=1)  
 totalLos += loss  
 correct += torch.sum(predictions == labels)  
 testAccuracy = correct.item() / (BATCH\_SIZE \* len(testDataLoader))  
 testLoss = totalLoss / len(testDataLoader)  
 history['Test Loss'].append(testLoss.item())  
 history['Test Accuracy'].append(testAccuracy)  
  
 processBar.set\_description(f"[{epoch}/{EPOCHS}] Loss: {loss.item():.4f}, Acc: {accuracy:.4f}, Test Loss: {testLoss.item():.4f}, Test Acc: {testAccuracy:.4f}"  
torch.save(net.state\_dict(), './model.pth')

* 1. **实验结果截图**

网络结构示意图

****

****



* 1. **实验小结**

通过这次实验，我学习到了卷积神经网络的简单应用，这次实验构建了一个简单的网络进行训练，一进行了5轮训练，最后准确率都高达98%，很好的完成了实验目的，对深度学习有了直观的感受。