zhejiang-yige

智能软件综合研究

实训报告



学期 2023-2024-1

班级 计算机科学与技术21(4)班

姓名 陈昊天

学号 2021329600006

教师 马明泽、吴婷婷

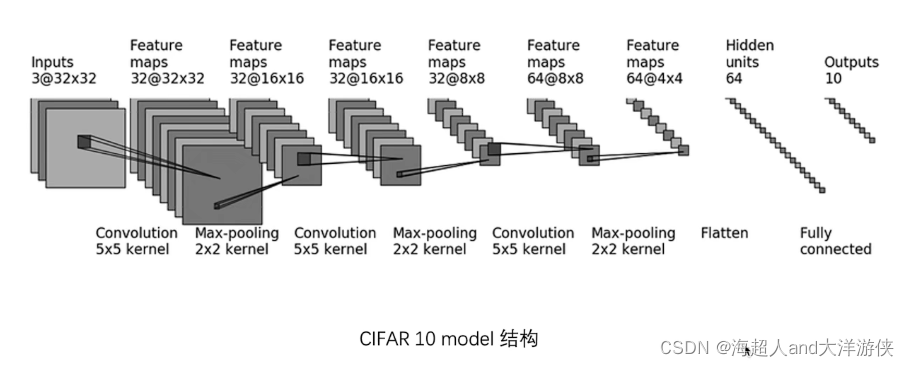
|  |  |  |
| --- | --- | --- |
| 学号： 2021329600006 |  | 姓名： 陈昊天 |

1. **实验目的和要求**
2. 掌握数据预处理相关操作；
3. 掌握深度学习模型构建、训练、评估、优化等过程；
4. **实验内容**

CIFAR-10数据集包含10个类别32×32的真实物体RGB图像，如飞机、汽车、鸟、猫、鹿、狗、蛙、马、船和卡车。每个类别有6000个图像，数据集中一共有50000 张训练图片和10000 张测试图片。通过数据预处理、模型构建、模型训练、模型评估、超参优化等步骤，完成CIFAR-10的预测任务和模型优化分析任务。

1. **实验任务**
2. **数据加载及预处理。**
3. **模型构建、训练、评估**

* 实现以下网络结构。



* 模型可视化。利用TensorBoard生成该卷积网络的图像。
* 模型训练。
* 模型评估。除了准确度分析外，利用ROC曲线对模型性能进行评估。

1. **实验过程和结果分析**
2. **数据加载及预处理。**

import torch

import torchvision

import torchvision.transforms as transforms

# 转换为Tensor，归一化

transform = transforms.Compose(

[transforms.ToTensor(), transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]

)

# 下载训练集，应用转换

trainset = torchvision.datasets.CIFAR10(

root="./data", train=True, download=True, transform=transform

)

trainloader = torch.utils.data.DataLoader(

trainset, batch\_size=4, shuffle=True, num\_workers=8

)

# 下载测试集，应用转换

testset = torchvision.datasets.CIFAR10(

root="./data", train=False, download=True, transform=transform

)

testloader = torch.utils.data.DataLoader(

testset, batch\_size=4, shuffle=False, num\_workers=8

)

# CIFAR-10的类别

classes = (

"plane",

"car",

"bird",

"cat",

"deer",

"dog",

"frog",

"horse",

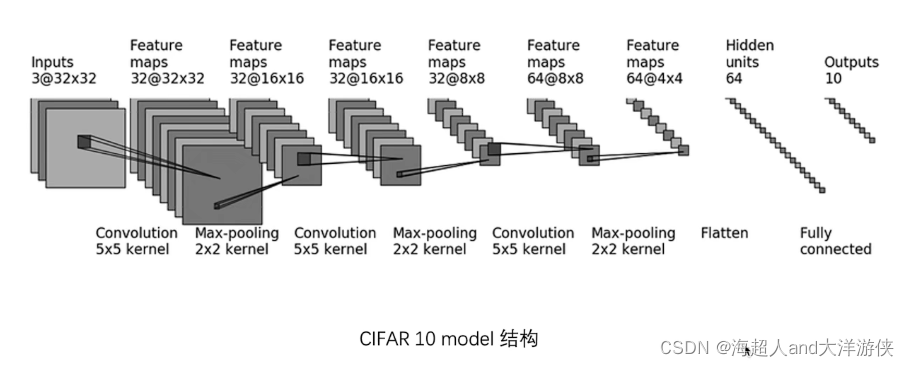
"ship",

"truck",

)

**2. 模型构建、训练、评估**

* 实现以下网络结构。



import torch.nn as nn

import torch.nn.functional as F

class Net(nn.Module):

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

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

self.model = nn.Sequential(

nn.Conv2d(3, 32, kernel\_size=5, padding=2),

nn.MaxPool2d(2),

nn.Conv2d(32, 32, kernel\_size=5, padding=2),

nn.MaxPool2d(2),

nn.Conv2d(32, 64, kernel\_size=5, padding=2),

nn.MaxPool2d(2),

nn.Flatten(),

nn.Linear(1024, 64),

nn.Linear(64, 10),

)

def forward(self, x):

return self.model(x)

net = Net()

* 模型可视化。利用TensorBoard生成该卷积网络的图像。

from torch.utils.tensorboard import SummaryWriter

writer = SummaryWriter("runs/cifar10\_experiment\_1")

# 获取随机训练数据

dataiter = iter(trainloader)

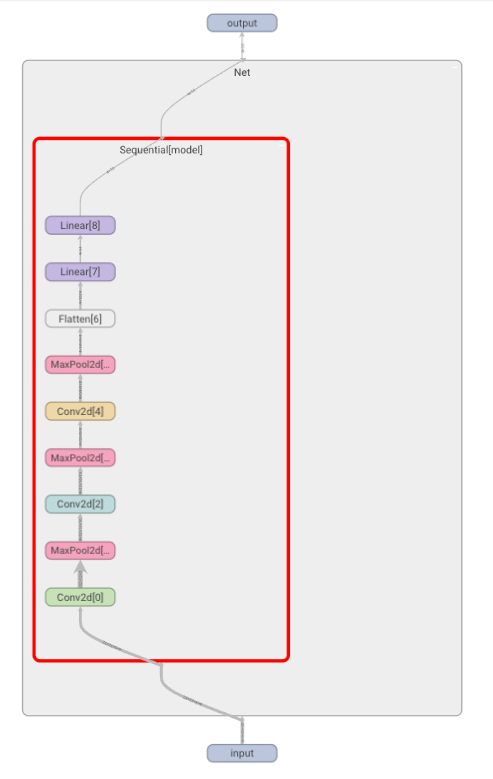
images, labels = next(dataiter)

# 写入模型图

writer.add\_graph(net, images)

writer.close()

模型图如下：



* 模型训练。

import torch.optim as optim

# 损失函数、优化器

criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)

# 训练网络

for epoch in range(10):

running\_loss = 0.0

for i, data in enumerate(trainloader, 0):

# 获取输入数据

inputs, labels = data

# 梯度置零

optimizer.zero\_grad()

# 前向传播、反向传播、优化

outputs = net(inputs)

loss = criterion(outputs, labels)

loss.backward()

optimizer.step()

# 统计信息

running\_loss += loss.item()

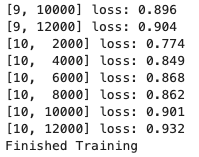
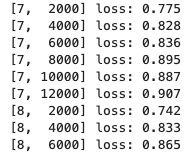
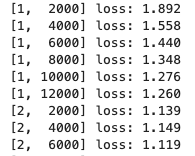
if i % 2000 == 1999:

print("[%d, %5d] loss: %.3f" % (epoch + 1, i + 1, running\_loss / 2000))

running\_loss = 0.0

print("Finished Training")

训练过程如图所示：



* 模型评估。除了准确度分析外，利用ROC曲线对模型性能进行评估。

correct = 0

total = 0

with torch.no\_grad():

for data in testloader:

images, labels = data

outputs = net(images)

\_, predicted = torch.max(outputs.data, 1)

total += labels.size(0)

correct += (predicted == labels).sum().item()

print("Accuracy of the network on the test images: %d %%" % (100 \* correct / total))

import numpy as np

from sklearn.metrics import roc\_curve, auc

from sklearn.preprocessing import label\_binarize

import matplotlib.pyplot as plt

# 标签二值化

y\_test = label\_binarize(testset.targets, classes=list(range(10)))

n\_classes = y\_test.shape[1]

# 获取预测概率

y\_score = np.zeros((len(testset), n\_classes))

# 禁用梯度计算

with torch.no\_grad():

for i, data in enumerate(testloader, 0):

images, labels = data

outputs = net(images)

probabilities = F.softmax(outputs, dim=1)

y\_score[i \* testloader.batch\_size : (i+1) \* testloader.batch\_size, :] = probabilities.numpy()

# 计算ROC和AUC

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(n\_classes):

fpr[i], tpr[i], \_ = roc\_curve(y\_test[:, i], y\_score[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

# 绘制ROC

for i in range(n\_classes):

plt.figure()

plt.plot(fpr[i], tpr[i], label='ROC curve (area = %0.2f)' % roc\_auc[i])

plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

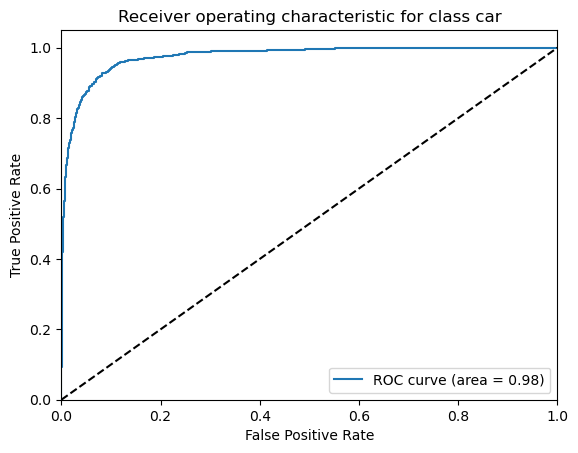
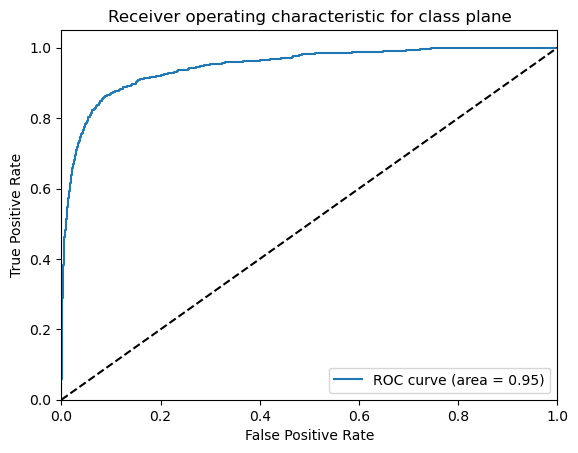
plt.title('Receiver operating characteristic for class {}'.format(classes[i]))

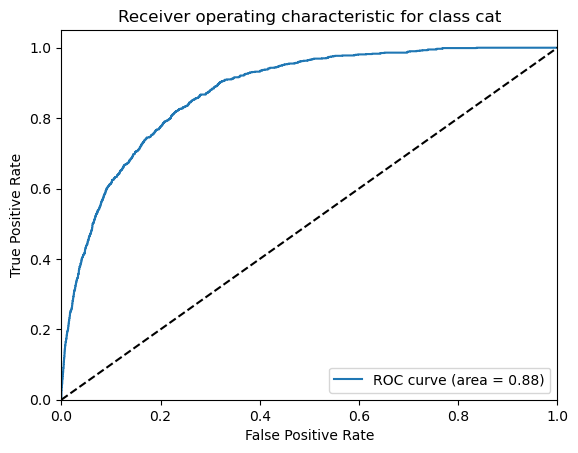
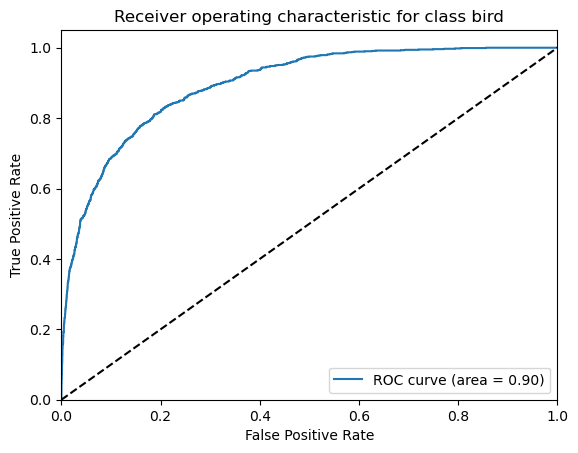
plt.legend(loc="lower right")

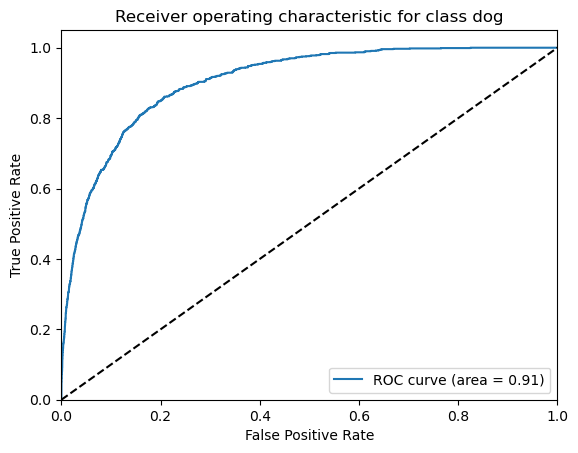
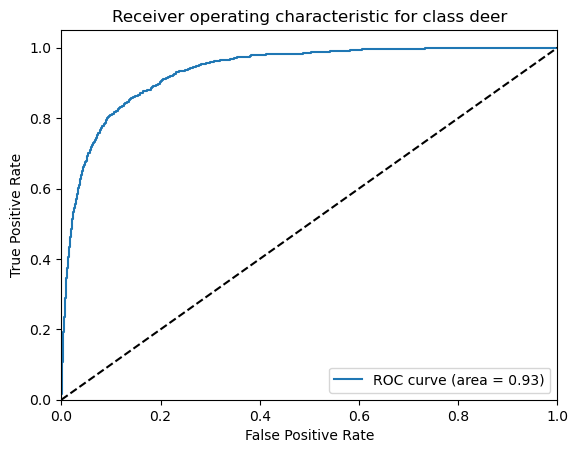
plt.show()

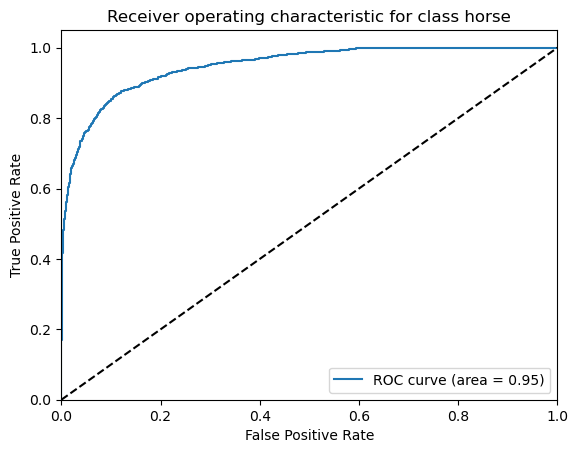
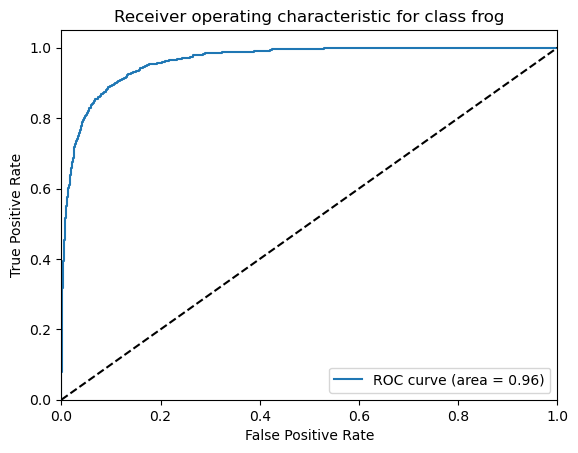
结果如图所示：

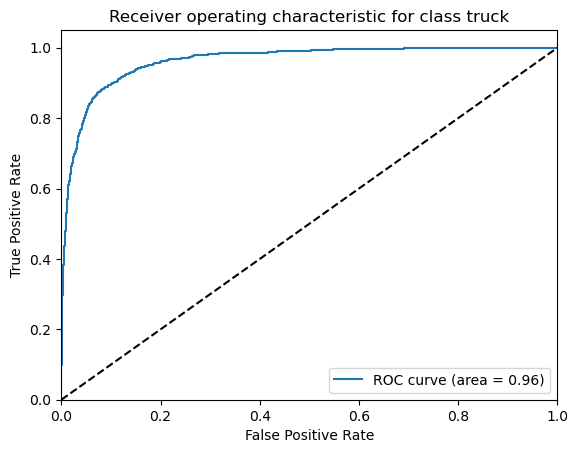
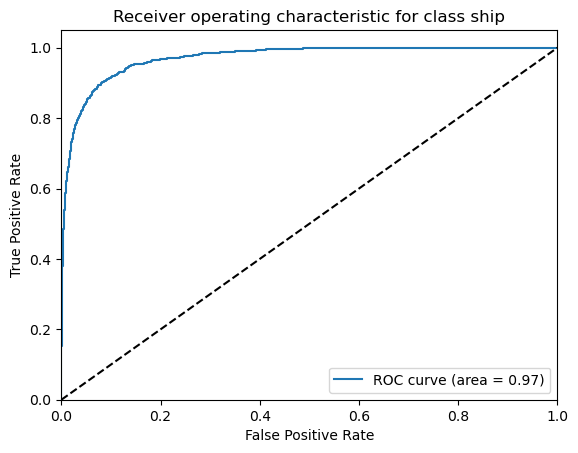












1. **实验总结**
   1. **总结**

本次实验我们掌握了数据预处理的相关操作，包括加载CIFAR-10数据集、转换为Tensor格式并进行归一化处理。通过这些步骤，数据被准备好输入到深度学习模型中。

实验构建了一个卷积神经网络模型，使用最大池化层来减少参数数量和计算量，同时也有助于提取特征。模型构建完成后，利用TensorBoard工具对网络结构进行了可视化，有助于更直观地理解模型的结构。

在模型训练阶段，使用交叉熵损失函数和带动量的随机梯度下降优化器。通过迭代训练，模型的损失逐渐降低，表明模型正在学习数据集中的特征。在每个epoch结束时记录了训练损失以监控训练过程。

模型评估方面不仅计算了模型在测试集上的准确率，还使用ROC曲线和AUC值对模型的性能进行了评估。通过分析每个类别的ROC曲线和AUC值，可以更详细地了解模型在不同类别上的表现。

* 1. **心得体会**

通过这次实验，我更加深刻地体会到了深度学习在图像识别任务中的强大能力。实验过程中，我不仅学习了如何对数据进行预处理，还亲自搭建了一个卷积神经网络，并成功训练和评估了模型。这个过程中，我对神经网络的结构和工作原理有了更加清晰的认识。

我也体会到了超参数调整对于模型性能的重要性。在实验中，调整学习率和动量等参数对训练结果有显著影响。使用TensorBoard可视化工具能够帮助我更直观地理解模型结构和训练过程。

在评估模型性能时，我学会了使用ROC曲线和AUC值这些更为复杂的指标，它们提供了比简单准确率更丰富的信息。通过这些指标，我能够对模型在各个类别上的表现进行细致分析。

这次实验不仅增强了我的实践技能，也加深了我对深度学习理论知识的理解。我相信这些经验将在未来的学习和研究中发挥重要作用。