Lab 5: 基于深度学习的CIFAR-10图像分类

231880101 孙俊晖

1 引言

1.1 背景

CIFAR-10数据集是一个经典的图像分类数据集，包含了10个类别的彩色图片，这些类别分别是飞机（plane）、汽车（car）、鸟类（bird）、猫（cat）、鹿（deer）、狗（dog）、蛙类（frog）、马（horse）、船（ship）和卡车（truck）。每个类别有6000张图像，整个数据集共有60000张图片，其中50000张用于训练，10000张用于测试。由于其规模适中、分类任务比较复杂，因此CIFAR-10数据集成为了机器学习领域中一个广泛使用的基准数据集。

1.2 目标

复现PyTorch官方教程中的代码，实现利用卷积神经网络对CIFAR-10数据集进行分类，并尝试采用不同的网络结构、优化算法、超参数等对算法性能进行优化。

2 方法

2.1 技术栈

开发环境：Visual Studio Code 1.93.0

编程语言：Python 3.12.3

2.2 代码复现

2.2.1 导入必要的库

首先，我导入了PyTorch的核心库torch、PyTorch的神经网络库torch.nn、PyTorch的优化算法库torch.optim等必要的库。这些库将用于后续构建、训练和评估卷积神经网络。

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np
```

2.2.2 数据预处理与加载

接着，我将图像数据转换为PyTorch张量，并进行标准化处理，以便后续的网络训练。同时，我还加载了CIFAR-10数据集的训练集和测试集，并设置了数据加载器。

```python
#将图像转换为张量并标准化
transform = transforms.Compose(
    [transforms.ToTensor(),  #转换为张量
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])  #标准化
```

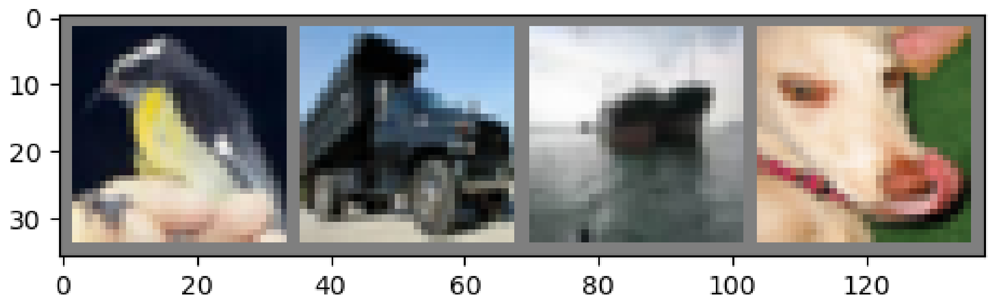
```
#加载训练数据集
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4,
                                          shuffle=True, num_workers=2)
#加载测试数据集
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=2)
#CIFAR-10数据集的类别
classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',
'ship', 'truck')
```

2.2.3 数据可视化

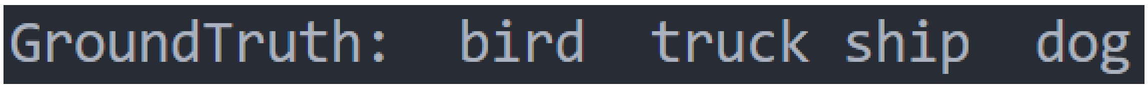
为了更好地理解CIFAR-10数据集，我选择从训练集中随机挑选4张图像进行可视化，并打印图像对应的类别。

```
#显示图像的函数
def imshow(img):
    img = img / 2 + 0.5     #反标准化
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))  #转换通道顺序以匹配matplotlib
    plt.show()
#显示一些随机训练图像
dataiter = iter(trainloader)
images, labels = next(dataiter)
#显示图像
imshow(torchvision.utils.make_grid(images))
#打印类别
print('GroundTruth: ',' '.join(f'{classes[labels[j]]:5s}' for j in
range(4)))
```

随机挑选的图像如下图所示：



图像对应的类别为：



```
GroundTruth:  bird  truck ship  dog
```

2.2.4 定义卷积神经网络

然后，我定义一个简单的卷积神经网络（CNN）。这个网络将包含两个卷积层，每个卷积层后面跟着一个ReLU激活函数和一个最大池化层，最后是全连接层。

```python
#定义卷积神经网络（CNN）
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        #第一个卷积层，输入通道3，输出通道6，卷积核大小5
        self.conv1 = nn.Conv2d(3, 6, 5)
        #最大池化层，池化窗口2x2
        self.pool = nn.MaxPool2d(2, 2)
        #第二个卷积层，输入通道6，输出通道16，卷积核大小5
        self.conv2 = nn.Conv2d(6, 16, 5)
        #第一个全连接层，输入特征数量16*5*5，输出特征数量120
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        #第二个全连接层，输入特征数量120，输出特征数量84
        self.fc2 = nn.Linear(120, 84)
        #第三个全连接层（输出层），输入特征数量84，输出特征数量10（类别数）
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        #通过第一个卷积层和池化层
        x = self.pool(F.relu(self.conv1(x)))
        #通过第二个卷积层和池化层
        x = self.pool(F.relu(self.conv2(x)))
        #将卷积层输出展平为一维向量
        x = x.view(-1, 16 * 5 * 5)
        #通过第一个全连接层和ReLU激活函数
        x = F.relu(self.fc1(x))
        #通过第二个全连接层和ReLU激活函数
        x = F.relu(self.fc2(x))
        #通过第三个全连接层（输出层）
        x = self.fc3(x)
        return x

net = Net()
```

2.2.5 定义损失函数和优化器

我选择使用交叉熵损失函数作为损失函数，使用SGD优化器作为优化器。

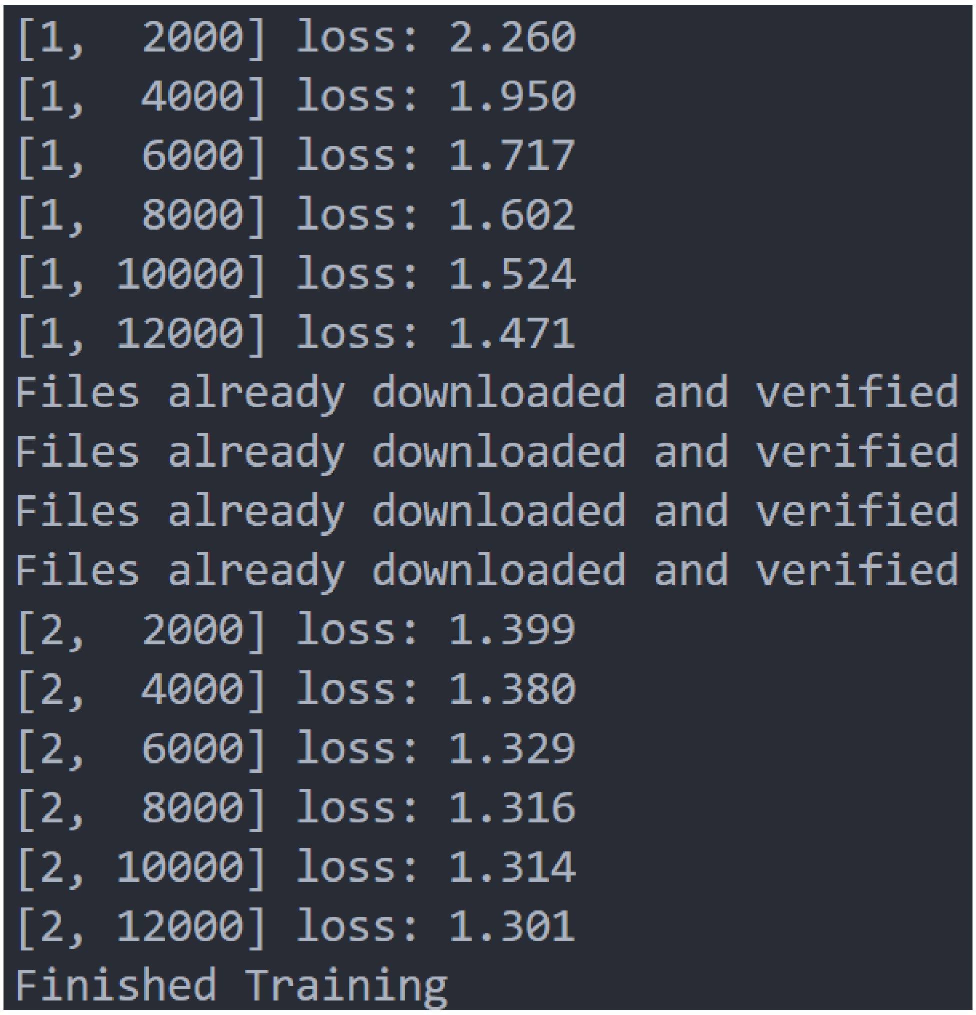
```
#定义损失函数和优化器
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

2.2.6 训练网络

现在就可以训练构建好的卷积神经网络了。为了提高训练效率，我选择在GPU上训练。与在CPU上直接训练相比，在GPU上训练需要将模型和数据移动到GPU上。在训练过程中，会打印当前训练周期内过去2000个小批量数据的平均损失值。在训练过后，将模型保存到指定路径中。

```
#设置设备
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
#训练循环
for epoch in range(2):  #训练2次
        running_loss = 0.0
        for i, data in enumerate(trainloader, 0):
            #获取输入和类别，并移动到GPU上
            inputs, labels = data[0].to(device), data[1].to(device)
            #清零参数梯度
            optimizer.zero_grad()
            #前向传播、反向传播和优化
            outputs = net(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            #打印平均损失值
            running_loss += loss.item()
            if i % 2000 == 1999:  # 每2000个mini-batch打印一次
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /
2000:.3f}')
                running_loss = 0.0
#保存训练好的模型参数
torch.save(net.state_dict(), PATH)
print('Finished Training')
PATH = './cifar_net.pth'
torch.save(net.state_dict(), PATH)
```

在训练过程中，数据的平均损失值变化如下：



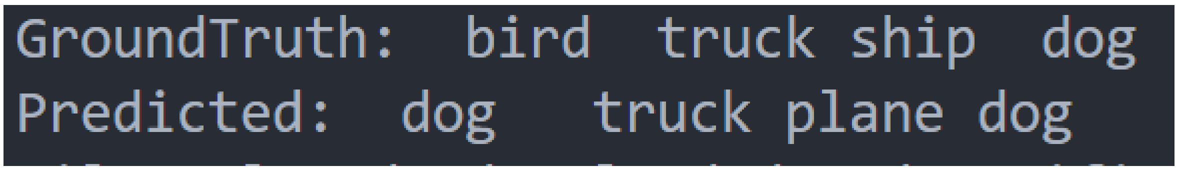
```
[1,  2000] loss: 2.260
[1,  4000] loss: 1.950
[1,  6000] loss: 1.717
[1,  8000] loss: 1.602
[1, 10000] loss: 1.524
[1, 12000] loss: 1.471
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
[2,  2000] loss: 1.399
[2,  4000] loss: 1.380
[2,  6000] loss: 1.329
[2,  8000] loss: 1.316
[2, 10000] loss: 1.314
[2, 12000] loss: 1.301
Finished Training
```

2.2.7 测试网络

在测试阶段，首先利用训练好的模型预测之前随机选取的4张图像的类别：

```
#显示一些随机训练图像和预测结果
dataiter = iter(trainloader)
images, labels = next(dataiter)
#显示图像
imshow(torchvision.utils.make_grid(images))
#打印类别
print('GroundTruth: ',' '.join(f'{classes[labels[j]]:5s}' for j in range(4)))
images = images.to(device)
outputs = net(images)
_, predicted = torch.max(outputs, 1)
print('Predicted: ', ' '.join(f'{classes[predicted[j]]:5s}' for j in range(4)))
```

预测的结果如下：



```
GroundTruth:  bird  truck ship  dog
Predicted:  dog   truck plane dog
```

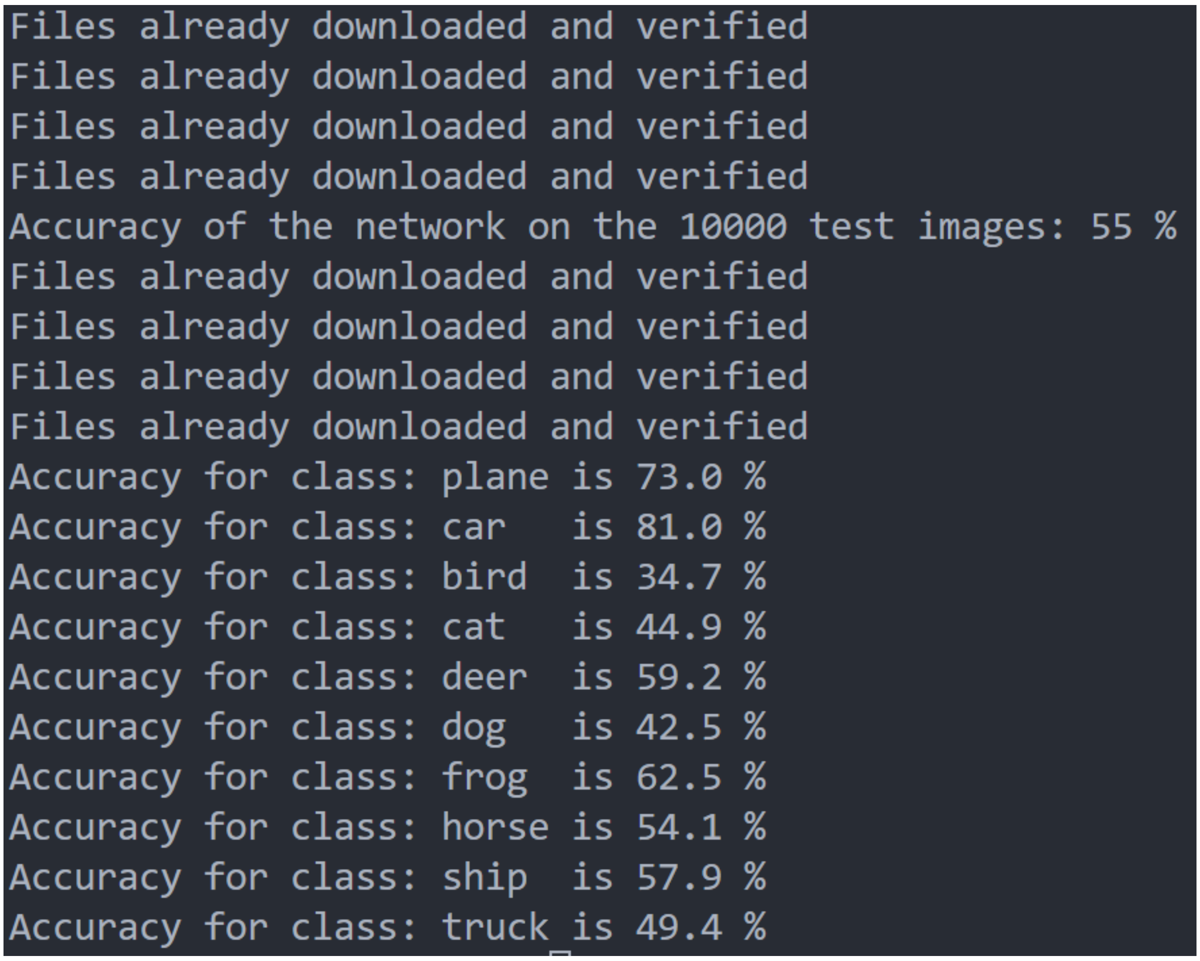
将预测的结果与图像真实的类比进行对比，发现训练的模型预测随机挑选的这四个图像的类别的准确率为50%。

最后，计算网络在测试数据集上的整体准确率和各个类别的准确率。

```python
#计算测试集上的准确率
correct = 0
total = 0
with torch.no_grad():  #测试时不计算梯度
    for data in testloader:
       images, labels = data[0].to(device), data[1].to(device)
       #通过网络计算输出
       outputs = net(images)
       #获取预测结果
       _, predicted = torch.max(outputs.data, 1)
       total += labels.size(0)
       correct += (predicted == labels).sum().item()
#打印整体准确率
print(f'Accuracy of the network on the 10000 test images: {100 * correct // total} %')

#准备计算每个类别的预测准确率
correct_pred = {classname: 0 for classname in classes}
total_pred = {classname: 0 for classname in classes}
#测试时不计算梯度
with torch.no_grad():
    for data in testloader:
       images, labels = data[0].to(device), data[1].to(device)
       outputs = net(images)
       _, predictions = torch.max(outputs, 1)
       #收集每个类别的正确预测数量
       for label, prediction in zip(labels, predictions):
          if label == prediction:
            correct_pred[classes[label]] += 1
            total_pred[classes[label]] += 1
#打印每个类别的准确率
for classname, correct_count in correct_pred.items():
   accuracy = 100 * float(correct_count) / total_pred[classname]
   print(f'Accuracy for class: {classname:5s} is {accuracy:.1f} %')
```

卷积神经网络在测试集上的评估结果如下：

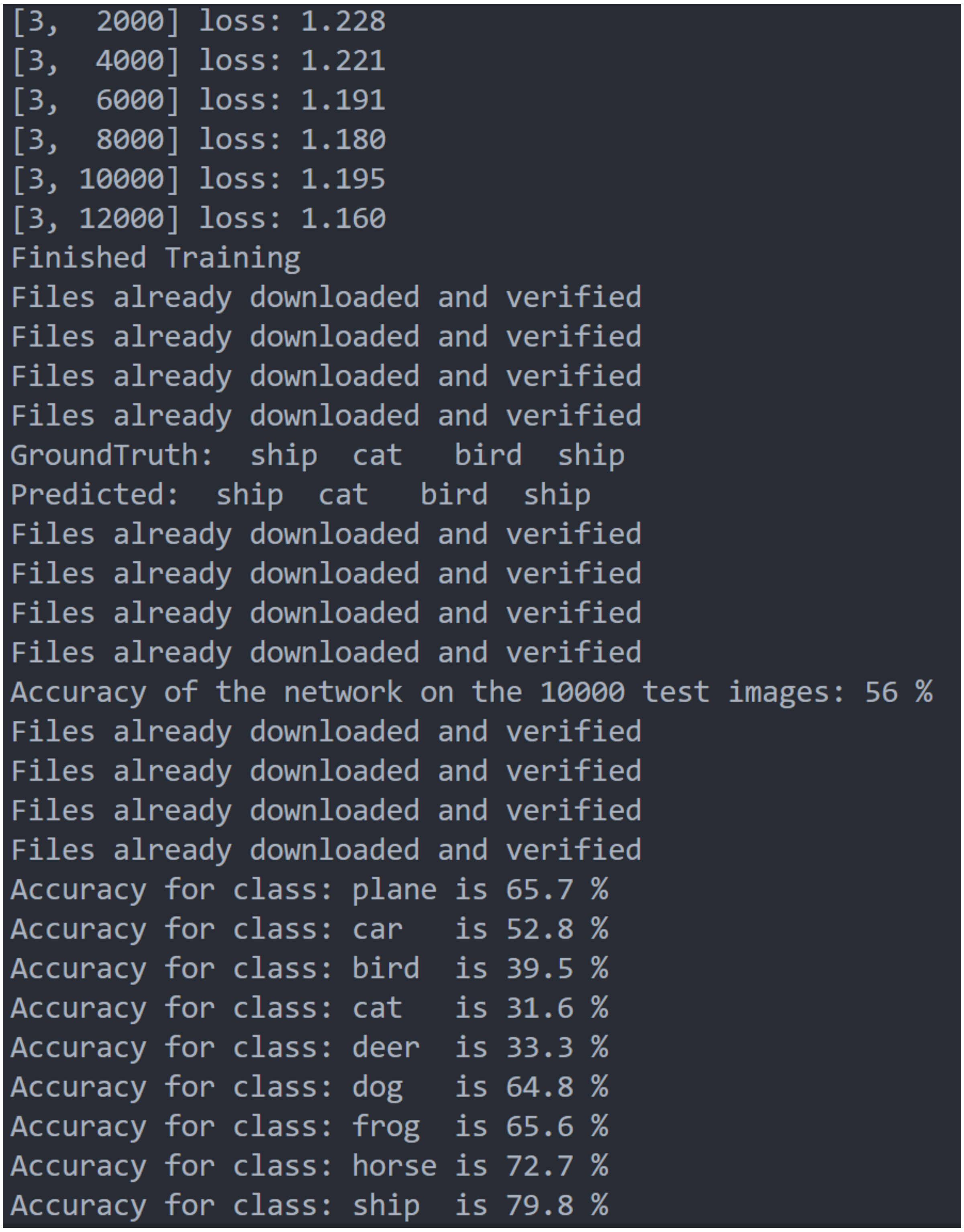

```
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 55 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 73.0 %
Accuracy for class: car   is 81.0 %
Accuracy for class: bird  is 34.7 %
Accuracy for class: cat   is 44.9 %
Accuracy for class: deer  is 59.2 %
Accuracy for class: dog   is 42.5 %
Accuracy for class: frog  is 62.5 %
Accuracy for class: horse is 54.1 %
Accuracy for class: ship  is 57.9 %
Accuracy for class: truck is 49.4 %
```

在训练数据集上训练了 2 次的网络在测试集上的整体准确率为55%，近似等于PyTorch教程中的整体准确率(54%)，代码复现完成。

2.2.8 训练次数对卷积神经网络的影响

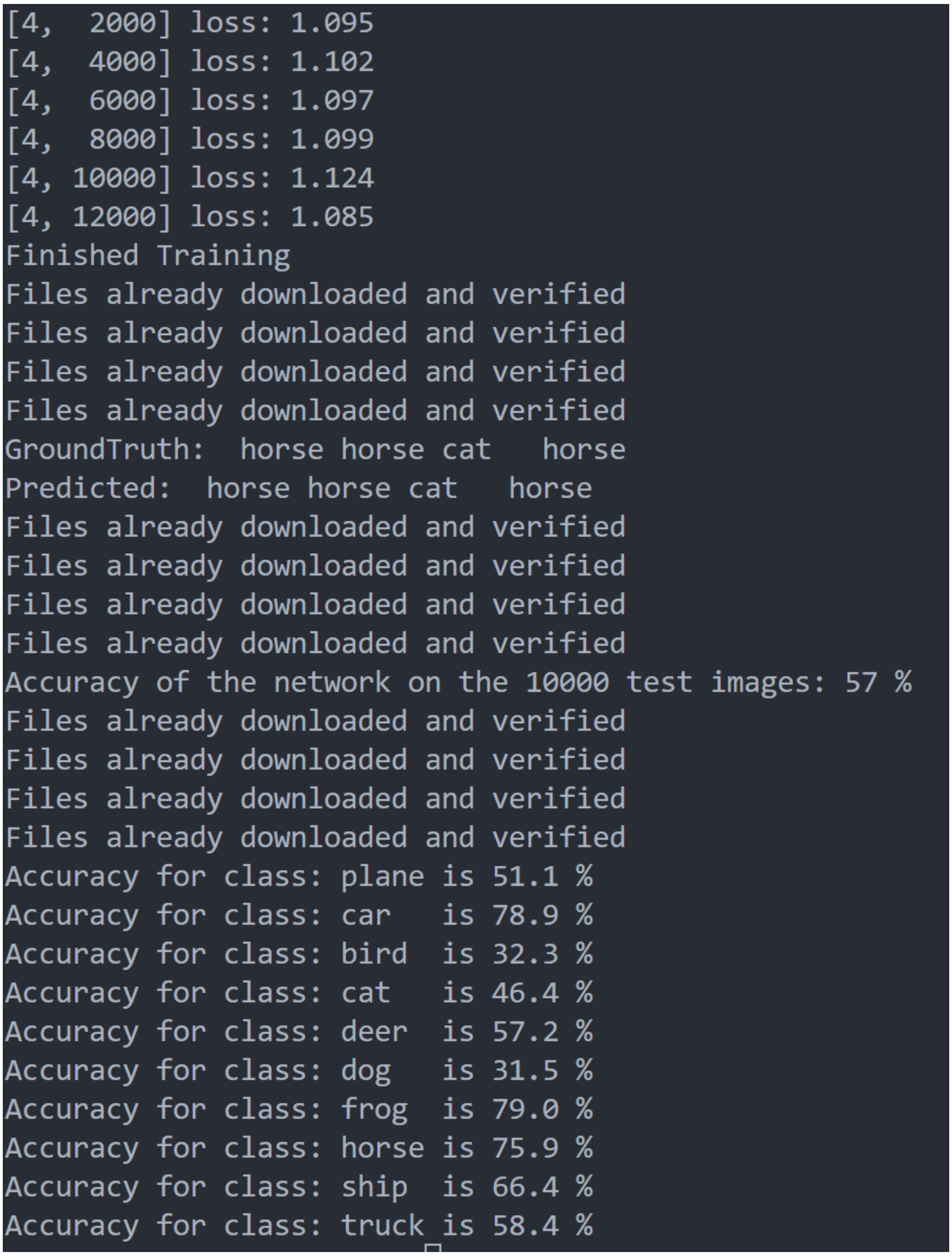
在复现完PyTorch中的代码后，我尝试在训练过程中改变训练循环的次数，结果发现随着训练次数的增加，卷积神经网络的平均损失值逐渐降低直至一个下界，在测试集上的整体准确率逐渐提高直至一个上界，而各个类别的准确率在不断浮动。下面展示了在训练集上训练次数不同的网络在最后一次训练周期中的平均值损失以及在测试集上的准确率。

在训练数据集上训练3次的网络：



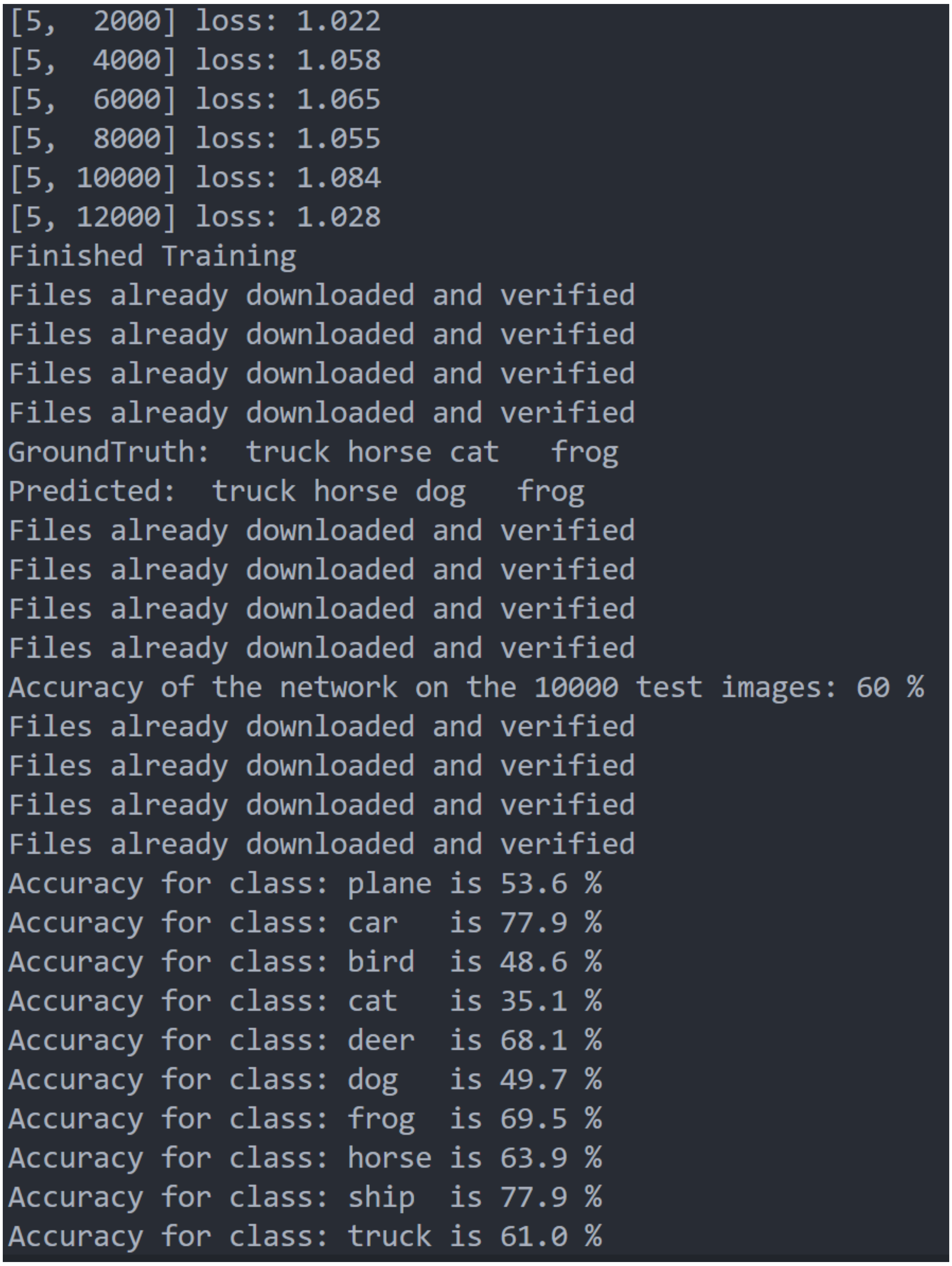
```
[3,  2000] loss: 1.228
[3,  4000] loss: 1.221
[3,  6000] loss: 1.191
[3,  8000] loss: 1.180
[3, 10000] loss: 1.195
[3, 12000] loss: 1.160
Finished Training
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
GroundTruth:  ship  cat   bird  ship
Predicted:  ship  cat   bird  ship
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 56 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 65.7 %
Accuracy for class: car   is 52.8 %
Accuracy for class: bird  is 39.5 %
Accuracy for class: cat   is 31.6 %
Accuracy for class: deer  is 33.3 %
Accuracy for class: dog   is 64.8 %
Accuracy for class: frog  is 65.6 %
Accuracy for class: horse is 72.7 %
Accuracy for class: ship  is 79.8 %
```

在训练集上训练4次的网络：

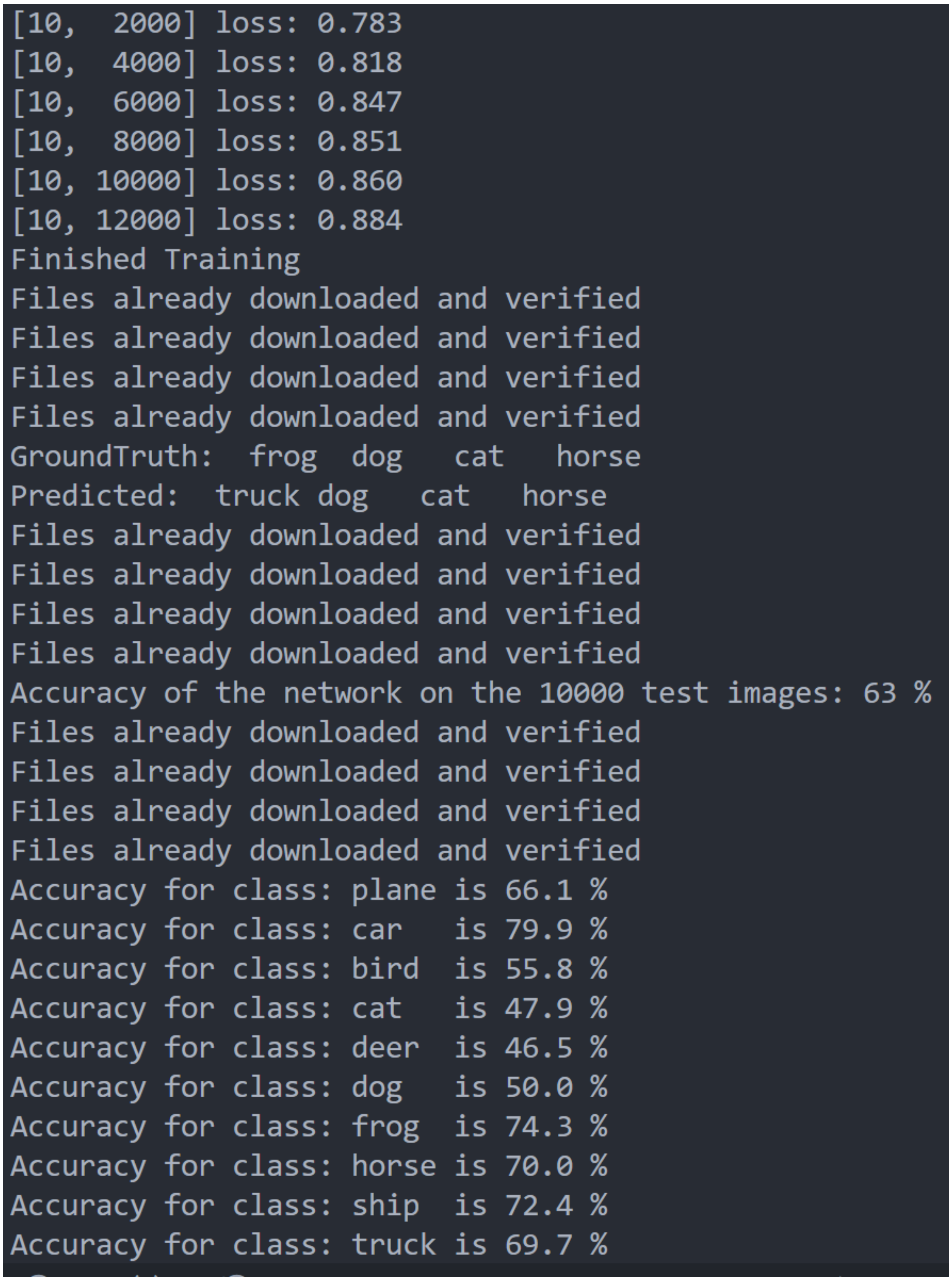


```
[4,  2000] loss: 1.095
[4,  4000] loss: 1.102
[4,  6000] loss: 1.097
[4,  8000] loss: 1.099
[4, 10000] loss: 1.124
[4, 12000] loss: 1.085
Finished Training
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
GroundTruth:  horse horse cat   horse
Predicted:  horse horse cat   horse
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 57 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 51.1 %
Accuracy for class: car   is 78.9 %
Accuracy for class: bird  is 32.3 %
Accuracy for class: cat   is 46.4 %
Accuracy for class: deer  is 57.2 %
Accuracy for class: dog   is 31.5 %
Accuracy for class: frog  is 79.0 %
Accuracy for class: horse is 75.9 %
Accuracy for class: ship  is 66.4 %
Accuracy for class: truck is 58.4 %
```

在训练集上训练5次的网络：

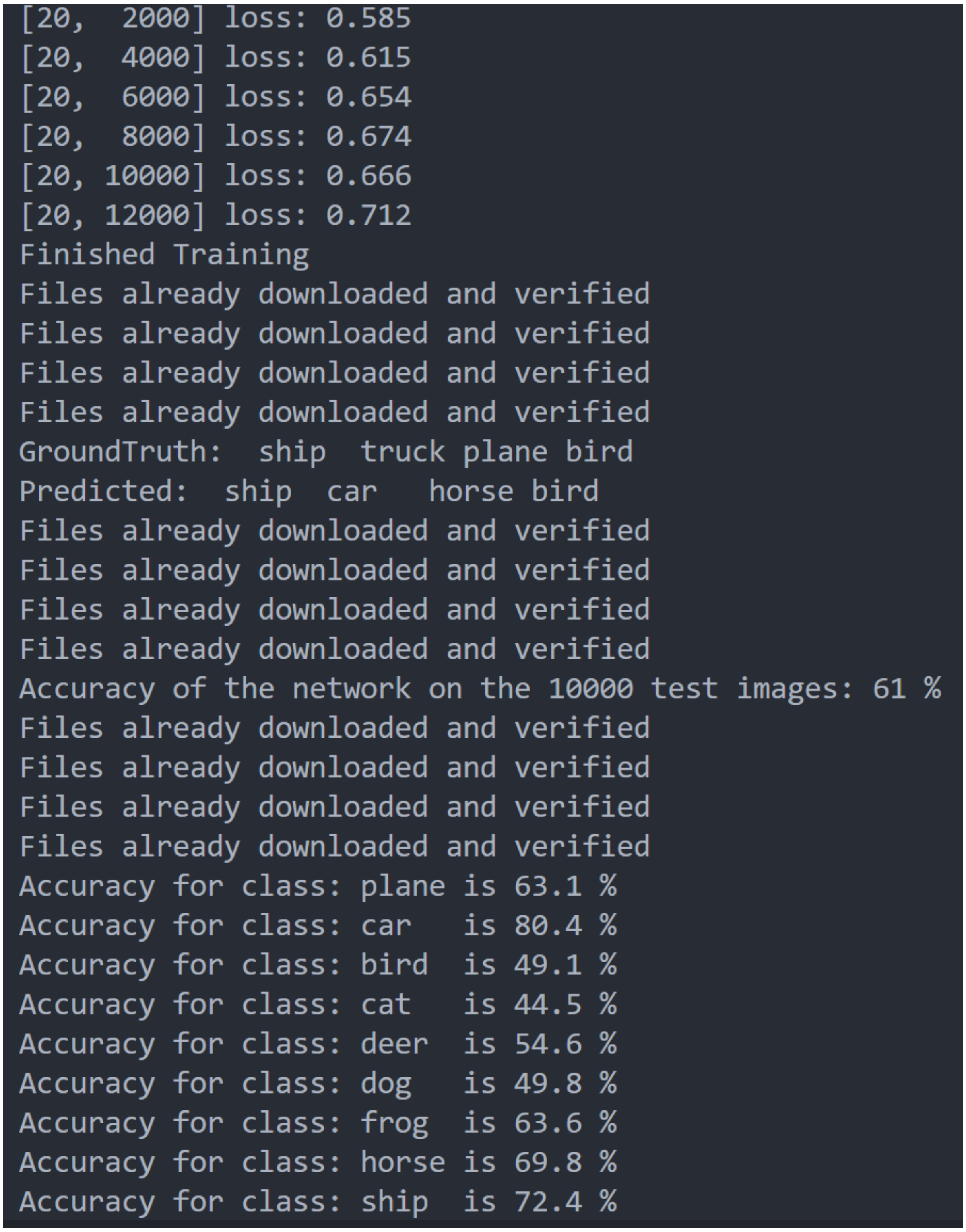

```
[5,  2000] loss: 1.022
[5,  4000] loss: 1.058
[5,  6000] loss: 1.065
[5,  8000] loss: 1.055
[5, 10000] loss: 1.084
[5, 12000] loss: 1.028
Finished Training
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
GroundTruth:  truck horse cat   frog
Predicted:  truck horse dog   frog
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 60 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 53.6 %
Accuracy for class: car   is 77.9 %
Accuracy for class: bird  is 48.6 %
Accuracy for class: cat   is 35.1 %
Accuracy for class: deer  is 68.1 %
Accuracy for class: dog   is 49.7 %
Accuracy for class: frog  is 69.5 %
Accuracy for class: horse is 63.9 %
Accuracy for class: ship  is 77.9 %
Accuracy for class: truck is 61.0 %
```

在训练集上训练10次的网络：



```
[10,  2000] loss: 0.783
[10,  4000] loss: 0.818
[10,  6000] loss: 0.847
[10,  8000] loss: 0.851
[10, 10000] loss: 0.860
[10, 12000] loss: 0.884
Finished Training
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
GroundTruth:  frog  dog   cat   horse
Predicted:  truck dog   cat   horse
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 63 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 66.1 %
Accuracy for class: car   is 79.9 %
Accuracy for class: bird  is 55.8 %
Accuracy for class: cat   is 47.9 %
Accuracy for class: deer  is 46.5 %
Accuracy for class: dog   is 50.0 %
Accuracy for class: frog  is 74.3 %
Accuracy for class: horse is 70.0 %
Accuracy for class: ship  is 72.4 %
Accuracy for class: truck is 69.7 %
```

在训练集上训练20次的网络：



```
[20,  2000] loss: 0.585
[20,  4000] loss: 0.615
[20,  6000] loss: 0.654
[20,  8000] loss: 0.674
[20, 10000] loss: 0.666
[20, 12000] loss: 0.712
Finished Training
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
GroundTruth:  ship  truck plane bird
Predicted:  ship  car   horse bird
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 61 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 63.1 %
Accuracy for class: car   is 80.4 %
Accuracy for class: bird  is 49.1 %
Accuracy for class: cat   is 44.5 %
Accuracy for class: deer  is 54.6 %
Accuracy for class: dog   is 49.8 %
Accuracy for class: frog  is 63.6 %
Accuracy for class: horse is 69.8 %
Accuracy for class: ship  is 72.4 %
```

最终我发现只通过改变训练次数，最多能将卷积神经网络在测试集上的整体准确率提高到**63%**。

2.3 优化算法性能

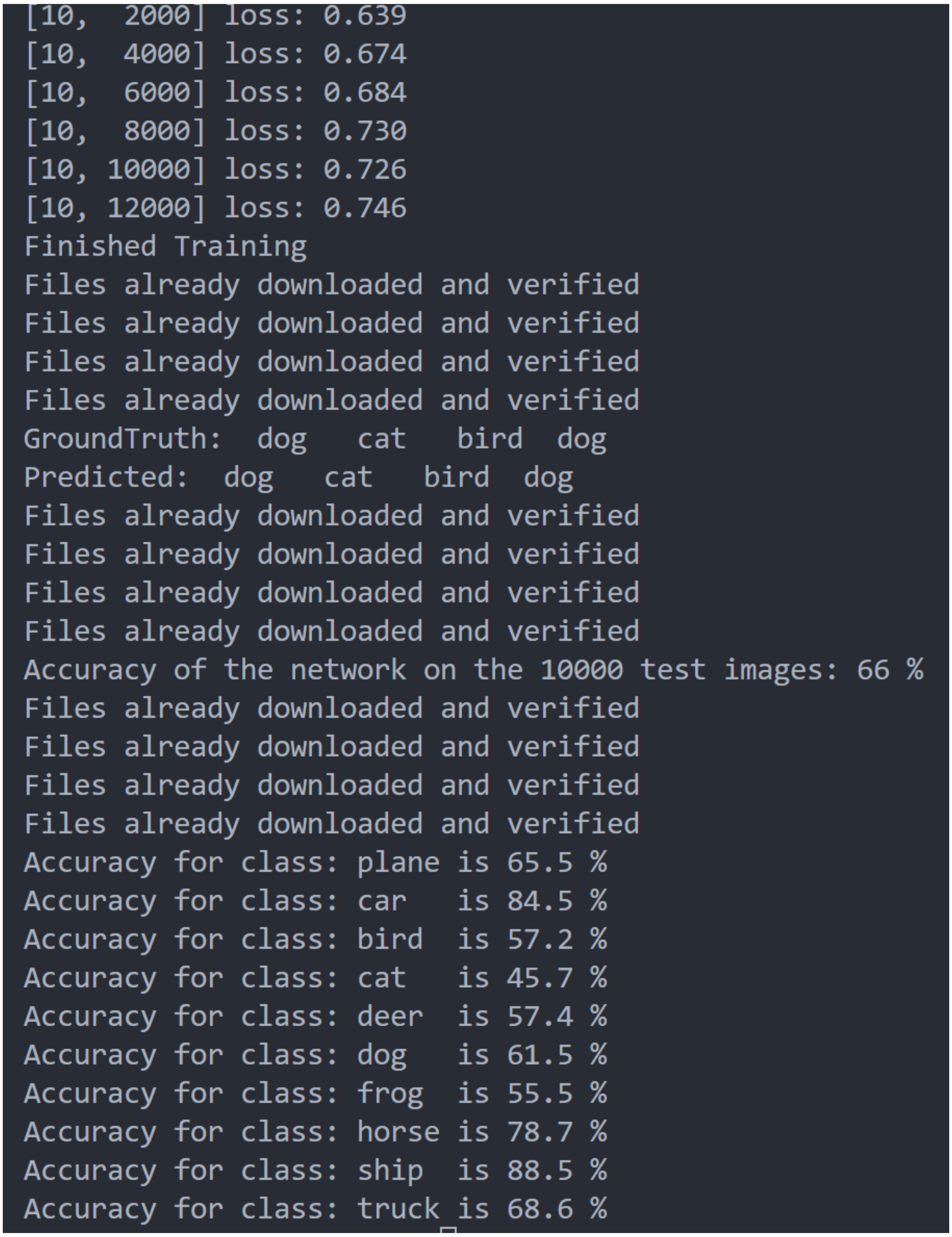
2.3.1 采用不同的网络结构

我尝试通过添加更多的卷积层来增加卷积神经网络的深度，以此来优化算法性能。

```
#定义卷积神经网络（CNN）
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5, padding=2)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5, padding=2)
        #新添加的卷积层
        self.conv3 = nn.Conv2d(16, 32, 5, padding=2)
        #计算全连接层的输入尺寸
        self.fc1 = nn.Linear(32 * 4 * 4, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = x.view(-1, 32 * 4 * 4)  #展平为一维向量，匹配fc1的输入尺寸
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

使用新的网络结构后，**在训练集上训练网络10次**，结果如下：



```
[10,  2000] loss: 0.639
[10,  4000] loss: 0.674
[10,  6000] loss: 0.684
[10,  8000] loss: 0.730
[10, 10000] loss: 0.726
[10, 12000] loss: 0.746
Finished Training
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
GroundTruth:  dog   cat   bird  dog
Predicted:  dog   cat   bird  dog
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 66 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 65.5 %
Accuracy for class: car   is 84.5 %
Accuracy for class: bird  is 57.2 %
Accuracy for class: cat   is 45.7 %
Accuracy for class: deer  is 57.4 %
Accuracy for class: dog   is 61.5 %
Accuracy for class: frog  is 55.5 %
Accuracy for class: horse is 78.7 %
Accuracy for class: ship  is 88.5 %
Accuracy for class: truck is 68.6 %
```

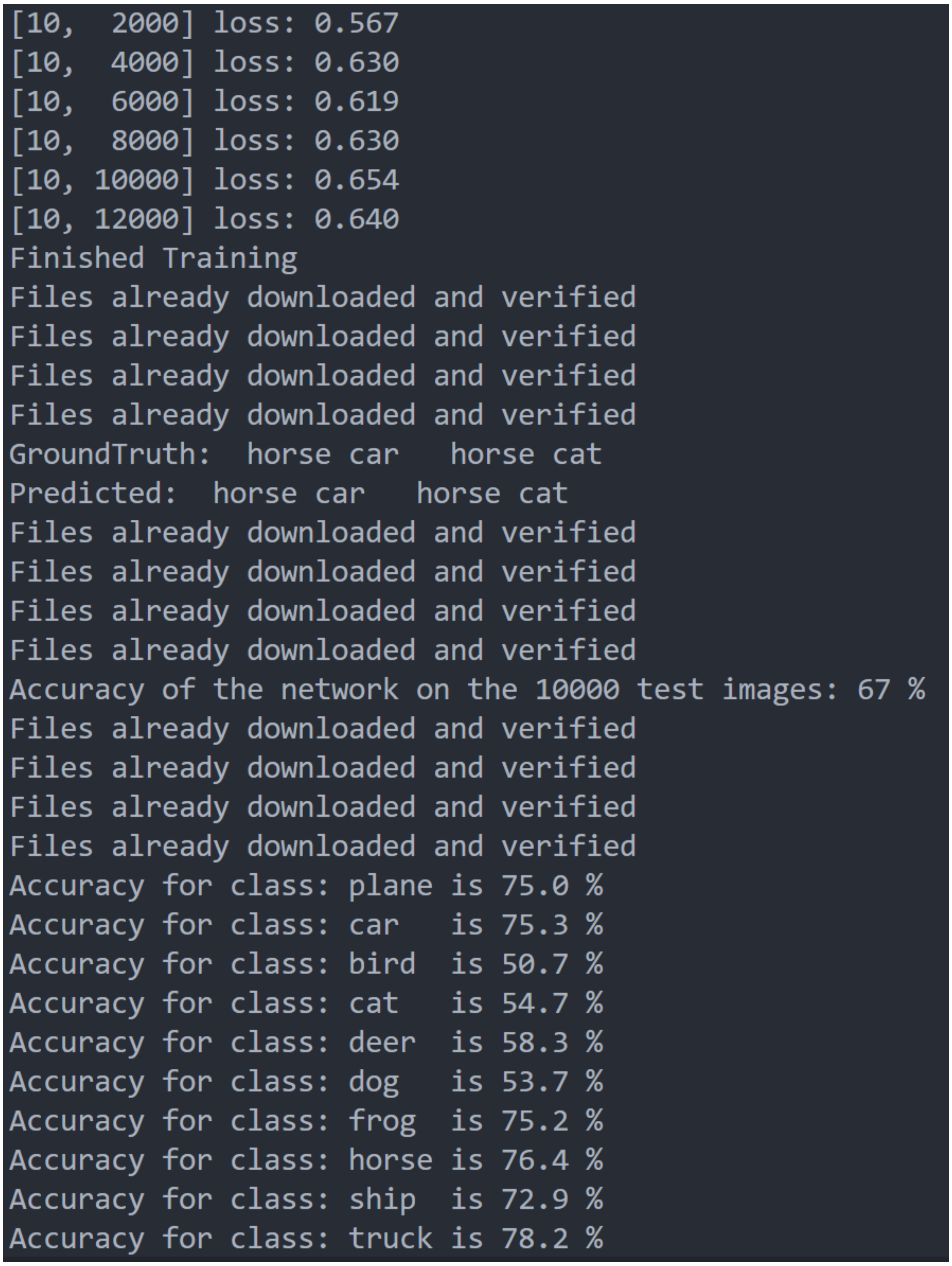
采用了新的网络结构后网络在测试集上的整体准确率从**63%提高到了66%**，平均损失值降低，各个类别的准确率亦有提升。

2.3.2 采用不同的优化算法

我尝试使用**Adam优化**器来替代**SGD优化**器。

```
optimizer = optim.Adam(net.parameters(), lr=0.0005)
```

使用Adam优化器后，**在训练集上训练网络10次**，结果如下：



```
[10,  2000] loss: 0.567
[10,  4000] loss: 0.630
[10,  6000] loss: 0.619
[10,  8000] loss: 0.630
[10, 10000] loss: 0.654
[10, 12000] loss: 0.640
Finished Training
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
GroundTruth:  horse car   horse cat
Predicted:  horse car   horse cat
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 67 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 75.0 %
Accuracy for class: car   is 75.3 %
Accuracy for class: bird  is 50.7 %
Accuracy for class: cat   is 54.7 %
Accuracy for class: deer  is 58.3 %
Accuracy for class: dog   is 53.7 %
Accuracy for class: frog  is 75.2 %
Accuracy for class: horse is 76.4 %
Accuracy for class: ship  is 72.9 %
Accuracy for class: truck is 78.2 %
```

更换优化器之后，卷积神经网络在测试集上的整体准确率从**66%提高到了67%**，平均损失值降低，各个类别的准确率亦有提升。

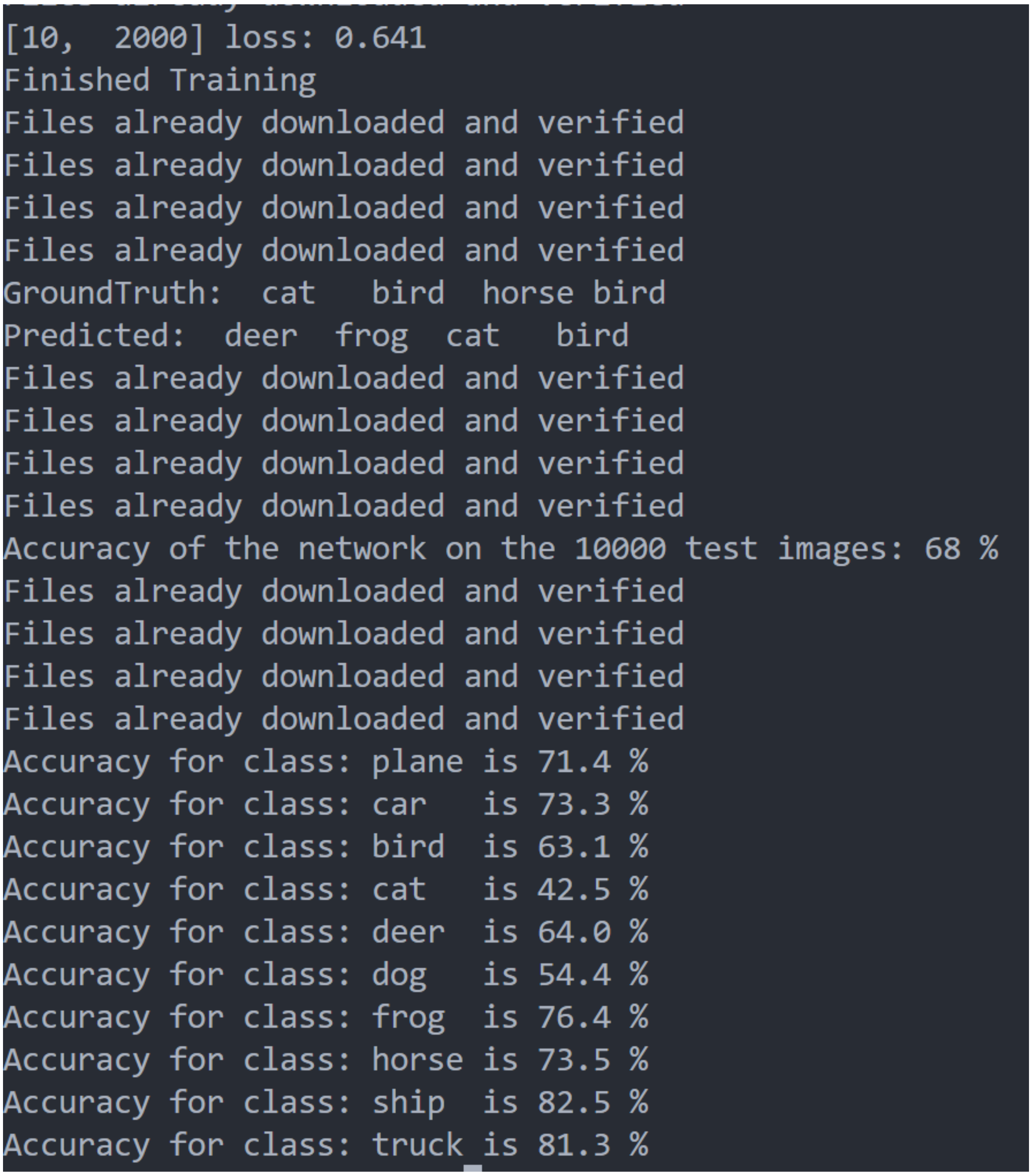
2.3.3 采用不同的超参数

首先，我为了优化网络的收敛情况，将训练次数由PyTorch教程中的2增加到10。除此之外，我尝试增大批量大小，以此来增加网络的训练稳定性。我将批量大小batch_size由原先的4改为了16。

```
#加载训练数据集
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=16,
                                          shuffle=True, num_workers=2)

#加载测试数据集
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=16,
                                         shuffle=False, num_workers=2)
```

增大批量大小后，**在训练集上训练网络10次**，结果如下：



```
[10,  2000] loss: 0.641
Finished Training
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
GroundTruth:   cat   bird  horse bird
Predicted:  deer  frog  cat   bird
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 test images: 68 %
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Accuracy for class: plane is 71.4 %
Accuracy for class: car   is 73.3 %
Accuracy for class: bird  is 63.1 %
Accuracy for class: cat   is 42.5 %
Accuracy for class: deer  is 64.0 %
Accuracy for class: dog   is 54.4 %
Accuracy for class: frog  is 76.4 %
Accuracy for class: horse is 73.5 %
Accuracy for class: ship  is 82.5 %
Accuracy for class: truck is 81.3 %
```

改变批量大小后，卷积神经网络在测试集上的整体准确率从**67%提高到了68%。**

2.3.4 算法性能的变化

最终，我通过采用不同的网络结构、优化算法、超参数，有效地优化了算法性能，使得网络在测试集上的整体准确率**由PyTorch教程中的54%提升到了68%**，同时网络在训练过程中的平均损失值明显降低，各个类别的准确率亦有明显提升。

3 总结

通过本次实验，我成功地复现了PyTorch教程中的代码，在 CIFAR-10 数据集上实现了一个基于卷积神经网络的图像分类模型。通过采用不同的网络结构、优化算法、超参数，我进一步提升了模型的性能。这个过程不仅加深了我对深度学习原理的理解，还锻炼了我解决实际问题的能力。

4 参考文献

1.https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

2.https://blog.csdn.net/Little_Carter/article/details/135934842

3.https://blog.csdn.net/Ever______/article/details/136596213?ops_request_misc=%7B%22request%5Fid%22%3A%22483d777d47d5ae4bbeca81299dbae95e%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=483d777d47d5ae4bbeca81299dbae95e&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-6-136596213-null-null.142