In [23]:

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
import torch.optim as optim
import time
import os

/*
CIFAR-10数据集共有有 50000 张训练图片以及10000 张测试图片，
图片总共 10 个类别，分别是：
 ‘airplane’, ‘automobile’, ‘bird’, ‘cat’, ‘deer’, ‘dog’, ‘frog’, ‘horse’
CIFAR-10 图片的尺寸是 32x32x3
*/


# 加载数据集
transform = transforms.Compose(
    [
     transforms.RandomHorizontalFlip()  # 随机水平翻转
     transforms.RandomGrayscale()  # 随机转为灰度图
     # 通过以上两种方法增加训练时数据集的容量
     transforms.ToTensor()  # 数据集加载的图片默认是numpy，通过transforms转化为Tensor
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])  # 对输入图片进行标准化

transform1 = transforms.Compose(
    [
     transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

# 加载数据集中的训练集
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=100,
                                          shuffle=True, num_workers=1)
# 加载数据集中的测试集
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True, transform=transform1)
testloader = torch.utils.data.DataLoader(testset, batch_size=50,
                                         shuffle=False, num_workers=1)

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')



class Net(nn.Module):


    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(3,64,3,padding=1)
        self.conv2 = nn.Conv2d(64,64,3,padding=1)
        self.pool1 = nn.MaxPool2d(2, 2)
        self.bn1 = nn.BatchNorm2d(64)
        self.relu1 = nn.ReLU()

        self.conv3 = nn.Conv2d(64,128,3,padding=1)
        self.conv4 = nn.Conv2d(128, 128, 3,padding=1)
        self.pool2 = nn.MaxPool2d(2, 2, padding=1)
        self.bn2 = nn.BatchNorm2d(128)
        self.relu2 = nn.ReLU()
```

```
        self.conv5 = nn.Conv2d(128,128, 3,padding=1)
        self.conv6 = nn.Conv2d(128, 128, 3,padding=1)
        self.conv7 = nn.Conv2d(128, 128, 1,padding=1)
        self.pool3 = nn.MaxPool2d(2, 2, padding=1)
        self.bn3 = nn.BatchNorm2d(128)
        self.relu3 = nn.ReLU()

        self.conv8 = nn.Conv2d(128, 256, 3,padding=1)
        self.conv9 = nn.Conv2d(256, 256, 3, padding=1)
        self.conv10 = nn.Conv2d(256, 256, 1, padding=1)
        self.pool4 = nn.MaxPool2d(2, 2, padding=1)
        self.bn4 = nn.BatchNorm2d(256)
        self.relu4 = nn.ReLU()

        self.conv11 = nn.Conv2d(256, 512, 3, padding=1)
        self.conv12 = nn.Conv2d(512, 512, 3, padding=1)
        self.conv13 = nn.Conv2d(512, 512, 1, padding=1)
        self.pool5 = nn.MaxPool2d(2, 2, padding=1)
        self.bn5 = nn.BatchNorm2d(512)
        self.relu5 = nn.ReLU()

        self.fc14 = nn.Linear(512*4*4,1024)
        self.drop1 = nn.Dropout2d()
        self.fc15 = nn.Linear(1024,1024)
        self.drop2 = nn.Dropout2d()
        self.fc16 = nn.Linear(1024,10)


    def forward(self,x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.pool1(x)
        x = self.bn1(x)
        x = self.relu1(x)


        x = self.conv3(x)
        x = self.conv4(x)
        x = self.pool2(x)
        x = self.bn2(x)
        x = self.relu2(x)

        x = self.conv5(x)
        x = self.conv6(x)
        x = self.conv7(x)
        x = self.pool3(x)
        x = self.bn3(x)
        x = self.relu3(x)

        x = self.conv8(x)
        x = self.conv9(x)
        x = self.conv10(x)
        x = self.pool4(x)
        x = self.bn4(x)
        x = self.relu4(x)

        x = self.conv11(x)
        x = self.conv12(x)
        x = self.conv13(x)
        x = self.pool5(x)
        x = self.bn5(x)
        x = self.relu5(x)
        # print(" x shape ",x.size())
        x = x.view(-1,512*4*4)
```

```
        x = F.relu(self.fc14(x))
        x = self.drop1(x)
        x = F.relu(self.fc15(x))
        x = self.drop2(x)
        x = self.fc16(x)

        return x

    def train_sgd(self,device):
        optimizer = optim.Adam(self.parameters(), lr=0.0001)

        path = 'weights.tar'
        initepoch = 0

        if os.path.exists(path) is not True:
            loss = nn.CrossEntropyLoss()
            # optimizer = optim.SGD(self.parameters(),lr=0.01)

        else:
            checkpoint = torch.load(path)
            self.load_state_dict(checkpoint['model_state_dict'])
            optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
            initepoch = checkpoint['epoch']
            loss = checkpoint['loss']




        for epoch in range(initepoch,100):  # loop over the dataset multiple times
            timestart = time.time()

            running_loss = 0.0
            total = 0
            correct = 0
            for i, data in enumerate(trainloader, 0):
                # get the inputs
                inputs, labels = data
                inputs, labels = inputs.to(device),labels.to(device)

                # zero the parameter gradients
                optimizer.zero_grad()

                # forward + backward + optimize
                outputs = self(inputs)
                l = loss(outputs, labels)
                l.backward()
                optimizer.step()

                # print statistics
                running_loss += l.item()
                # print("i ",i)
                if i % 500 == 499:  # print every 500 mini-batches
                    print('[%d, %5d] loss: %.4f' %
                          (epoch, i, running_loss / 500))
                    running_loss = 0.0
                    _, predicted = torch.max(outputs.data, 1)
                    total += labels.size(0)
                    correct += (predicted == labels).sum().item()
                    print('Accuracy of the network on the %d tran images: %.3f %%' %
                            100.0 * correct / total))
                    total = 0
                    correct = 0
                    torch.save({'epoch':epoch,
                                'model_state_dict':net.state_dict(),
                                'optimizer_state_dict':optimizer.state_dict(),
```

```
                                'loss':loss
                                },path)

            print('epoch %d cost %3f sec' %(epoch,time.time()-timestart))

        print('Finished Training')

    def test(self,device):
        correct = 0
        total = 0
        with torch.no_grad():
            for data in testloader:
                images, labels = data
                images, labels = images.to(device), labels.to(device)
                outputs = self(images)
                _, predicted = torch.max(outputs.data, 1)
                total += labels.size(0)
                correct += (predicted == labels).sum().item()

        print('Accuracy of the network on the 10000 test images: %.3f %%' % (
                100.0 * correct / total))

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
net = Net()
net = net.to(device)
net.train_sgd(device)
net.test(device)
```

```
Files already downloaded and verified
Files already downloaded and verified
[0,   499] loss: 1.4157
Accuracy of the network on the 100 tran images: 61.000 %
epoch 0 cost 508.527639 sec
[1,   499] loss: 0.9196
Accuracy of the network on the 100 tran images: 73.000 %
epoch 1 cost 488.130352 sec
[2,   499] loss: 0.7435
Accuracy of the network on the 100 tran images: 74.000 %
epoch 2 cost 488.855711 sec
[3,   499] loss: 0.6326
Accuracy of the network on the 100 tran images: 82.000 %
epoch 3 cost 496.843026 sec
[4,   499] loss: 0.5642
Accuracy of the network on the 100 tran images: 84.000 %
epoch 4 cost 434.165527 sec
[5,   499] loss: 0.5075
Accuracy of the network on the 100 tran images: 84.000 %
epoch 5 cost 495.289160 sec
[6,   499] loss: 0.4548
Accuracy of the network on the 100 tran images: 81.000 %
epoch 6 cost 493.505541 sec
[7,   499] loss: 0.4188
Accuracy of the network on the 100 tran images: 81.000 %
epoch 7 cost 496.580862 sec
[8,   499] loss: 0.3802
Accuracy of the network on the 100 tran images: 87.000 %
epoch 8 cost 497.154572 sec
[9,   499] loss: 0.3482
Accuracy of the network on the 100 tran images: 87.000 %
epoch 9 cost 496.897977 sec
[10,   499] loss: 0.3200
Accuracy of the network on the 100 tran images: 88.000 %
epoch 10 cost 496.664461 sec
[11,   499] loss: 0.3012
Accuracy of the network on the 100 tran images: 92.000 %
```

```
epoch 11 cost 496.789545 sec
[12,   499] loss: 0.2751
Accuracy of the network on the 100 tran images: 84.000 %
epoch 12 cost 496.483903 sec
[13,   499] loss: 0.2548
Accuracy of the network on the 100 tran images: 91.000 %
epoch 13 cost 496.686521 sec
[14,   499] loss: 0.2346
Accuracy of the network on the 100 tran images: 96.000 %
epoch 14 cost 496.295659 sec
[15,   499] loss: 0.2180
Accuracy of the network on the 100 tran images: 91.000 %
epoch 15 cost 496.728284 sec
[16,   499] loss: 0.2179
Accuracy of the network on the 100 tran images: 93.000 %
epoch 16 cost 496.354608 sec
[17,   499] loss: 0.1984
Accuracy of the network on the 100 tran images: 95.000 %
epoch 17 cost 496.523661 sec
[18,   499] loss: 0.1829
Accuracy of the network on the 100 tran images: 96.000 %
epoch 18 cost 496.590387 sec
[19,   499] loss: 0.1696
Accuracy of the network on the 100 tran images: 94.000 %
epoch 19 cost 496.542952 sec
[20,   499] loss: 0.1619
Accuracy of the network on the 100 tran images: 95.000 %
epoch 20 cost 493.248300 sec
[21,   499] loss: 0.1556
Accuracy of the network on the 100 tran images: 95.000 %
epoch 21 cost 493.421164 sec
[22,   499] loss: 0.1517
Accuracy of the network on the 100 tran images: 99.000 %
epoch 22 cost 496.833132 sec
[23,   499] loss: 0.1407
Accuracy of the network on the 100 tran images: 97.000 %
epoch 23 cost 496.807348 sec
[24,   499] loss: 0.1377
Accuracy of the network on the 100 tran images: 91.000 %
epoch 24 cost 496.619593 sec
[25,   499] loss: 0.1242
Accuracy of the network on the 100 tran images: 94.000 %
epoch 25 cost 496.553643 sec
[26,   499] loss: 0.1240
Accuracy of the network on the 100 tran images: 97.000 %
epoch 26 cost 493.298579 sec
[27,   499] loss: 0.1193
Accuracy of the network on the 100 tran images: 95.000 %
epoch 27 cost 496.704650 sec
[28,   499] loss: 0.1136
Accuracy of the network on the 100 tran images: 96.000 %
epoch 28 cost 496.690127 sec
[29,   499] loss: 0.1166
Accuracy of the network on the 100 tran images: 97.000 %
epoch 29 cost 496.410660 sec
[30,   499] loss: 0.1079
Accuracy of the network on the 100 tran images: 95.000 %
epoch 30 cost 496.432688 sec
[31,   499] loss: 0.1000
Accuracy of the network on the 100 tran images: 95.000 %
epoch 31 cost 493.111736 sec
[32,   499] loss: 0.1033
Accuracy of the network on the 100 tran images: 95.000 %
epoch 32 cost 493.459229 sec
[33,   499] loss: 0.0974
```

```
Accuracy of the network on the 100 tran images: 96.000 %
epoch 33 cost 492.985878 sec
[34,   499] loss: 0.0958
Accuracy of the network on the 100 tran images: 97.000 %
epoch 34 cost 492.716762 sec
[35,   499] loss: 0.0961
Accuracy of the network on the 100 tran images: 96.000 %
epoch 35 cost 496.507544 sec
[36,   499] loss: 0.0872
Accuracy of the network on the 100 tran images: 98.000 %
epoch 36 cost 496.378100 sec
[37,   499] loss: 0.0895
Accuracy of the network on the 100 tran images: 94.000 %
epoch 37 cost 495.662259 sec
[38,   499] loss: 0.0844
Accuracy of the network on the 100 tran images: 98.000 %
epoch 38 cost 496.240882 sec
[39,   499] loss: 0.0850
Accuracy of the network on the 100 tran images: 99.000 %
epoch 39 cost 493.372185 sec
[40,   499] loss: 0.0828
Accuracy of the network on the 100 tran images: 96.000 %
epoch 40 cost 496.023248 sec
[41,   499] loss: 0.0797
Accuracy of the network on the 100 tran images: 98.000 %
epoch 41 cost 496.006652 sec
[42,   499] loss: 0.0776
Accuracy of the network on the 100 tran images: 98.000 %
epoch 42 cost 492.537566 sec
[43,   499] loss: 0.0771
Accuracy of the network on the 100 tran images: 92.000 %
epoch 43 cost 496.084206 sec
[44,   499] loss: 0.0754
Accuracy of the network on the 100 tran images: 98.000 %
epoch 44 cost 496.134086 sec
[45,   499] loss: 0.0740
Accuracy of the network on the 100 tran images: 97.000 %
epoch 45 cost 495.972228 sec
[46,   499] loss: 0.0717
Accuracy of the network on the 100 tran images: 97.000 %
epoch 46 cost 493.708189 sec
[47,   499] loss: 0.0734
Accuracy of the network on the 100 tran images: 100.000 %
epoch 47 cost 496.219372 sec
[48,   499] loss: 0.0684
Accuracy of the network on the 100 tran images: 100.000 %
epoch 48 cost 496.067663 sec
[49,   499] loss: 0.0682
Accuracy of the network on the 100 tran images: 95.000 %
epoch 49 cost 496.334620 sec
[50,   499] loss: 0.0656
Accuracy of the network on the 100 tran images: 100.000 %
epoch 50 cost 496.581956 sec
[51,   499] loss: 0.0721
Accuracy of the network on the 100 tran images: 100.000 %
epoch 51 cost 496.530360 sec
[52,   499] loss: 0.0650
Accuracy of the network on the 100 tran images: 99.000 %
epoch 52 cost 493.358120 sec
[53,   499] loss: 0.0645
Accuracy of the network on the 100 tran images: 97.000 %
epoch 53 cost 496.248608 sec
[54,   499] loss: 0.0627
Accuracy of the network on the 100 tran images: 98.000 %
epoch 54 cost 496.462408 sec
```

```
[55,   499] loss: 0.0599
Accuracy of the network on the 100 tran images: 98.000 %
epoch 55 cost 496.298491 sec
[56,   499] loss: 0.0603
Accuracy of the network on the 100 tran images: 100.000 %
epoch 56 cost 496.135206 sec
[57,   499] loss: 0.0589
Accuracy of the network on the 100 tran images: 99.000 %
epoch 57 cost 495.947770 sec
[58,   499] loss: 0.0581
Accuracy of the network on the 100 tran images: 98.000 %
epoch 58 cost 493.328561 sec
[59,   499] loss: 0.0607
Accuracy of the network on the 100 tran images: 97.000 %
epoch 59 cost 496.042449 sec
[60,   499] loss: 0.0572
Accuracy of the network on the 100 tran images: 98.000 %
epoch 60 cost 495.971731 sec
[61,   499] loss: 0.0600
Accuracy of the network on the 100 tran images: 99.000 %
epoch 61 cost 492.826595 sec
[62,   499] loss: 0.0577
Accuracy of the network on the 100 tran images: 99.000 %
epoch 62 cost 492.667359 sec
[63,   499] loss: 0.0566
Accuracy of the network on the 100 tran images: 95.000 %
epoch 63 cost 492.987821 sec
[64,   499] loss: 0.0552
Accuracy of the network on the 100 tran images: 100.000 %
epoch 64 cost 496.562544 sec
[65,   499] loss: 0.0536
Accuracy of the network on the 100 tran images: 99.000 %
epoch 65 cost 493.046238 sec
[66,   499] loss: 0.0538
Accuracy of the network on the 100 tran images: 99.000 %
epoch 66 cost 496.971552 sec
[67,   499] loss: 0.0571
Accuracy of the network on the 100 tran images: 97.000 %
epoch 67 cost 496.294769 sec
[68,   499] loss: 0.0491
Accuracy of the network on the 100 tran images: 100.000 %
epoch 68 cost 271.365613 sec
[69,   499] loss: 0.0528
Accuracy of the network on the 100 tran images: 99.000 %
epoch 69 cost 497.407017 sec
[70,   499] loss: 0.0498
Accuracy of the network on the 100 tran images: 98.000 %
epoch 70 cost 497.945675 sec
[71,   499] loss: 0.0507
Accuracy of the network on the 100 tran images: 100.000 %
epoch 71 cost 493.146346 sec
[72,   499] loss: 0.0502
Accuracy of the network on the 100 tran images: 98.000 %
epoch 72 cost 492.885409 sec
[73,   499] loss: 0.0540
Accuracy of the network on the 100 tran images: 98.000 %
epoch 73 cost 496.854811 sec
[74,   499] loss: 0.0502
Accuracy of the network on the 100 tran images: 98.000 %
epoch 74 cost 497.257533 sec
[75,   499] loss: 0.0499
Accuracy of the network on the 100 tran images: 98.000 %
epoch 75 cost 496.419245 sec
[76,   499] loss: 0.0452
Accuracy of the network on the 100 tran images: 95.000 %
```

```
epoch 76 cost 492.992290 sec
[77,   499] loss: 0.0530
Accuracy of the network on the 100 tran images: 98.000 %
epoch 77 cost 496.988197 sec
[78,   499] loss: 0.0471
Accuracy of the network on the 100 tran images: 100.000 %
epoch 78 cost 497.032666 sec
[79,   499] loss: 0.0416
Accuracy of the network on the 100 tran images: 100.000 %
epoch 79 cost 497.462252 sec
[80,   499] loss: 0.0461
Accuracy of the network on the 100 tran images: 100.000 %
epoch 80 cost 281.016020 sec
[81,   499] loss: 0.0464
Accuracy of the network on the 100 tran images: 96.000 %
epoch 81 cost 260.470206 sec
[82,   499] loss: 0.0481
Accuracy of the network on the 100 tran images: 99.000 %
epoch 82 cost 260.802317 sec
[83,   499] loss: 0.0443
Accuracy of the network on the 100 tran images: 98.000 %
epoch 83 cost 260.716182 sec
[84,   499] loss: 0.0441
Accuracy of the network on the 100 tran images: 99.000 %
epoch 84 cost 260.895376 sec
[85,   499] loss: 0.0467
Accuracy of the network on the 100 tran images: 95.000 %
epoch 85 cost 261.204703 sec
[86,   499] loss: 0.0392
Accuracy of the network on the 100 tran images: 99.000 %
epoch 86 cost 261.087546 sec
[87,   499] loss: 0.0417
Accuracy of the network on the 100 tran images: 100.000 %
epoch 87 cost 261.001372 sec
[88,   499] loss: 0.0423
Accuracy of the network on the 100 tran images: 98.000 %
epoch 88 cost 261.346520 sec
[89,   499] loss: 0.0435
Accuracy of the network on the 100 tran images: 99.000 %
epoch 89 cost 261.271220 sec
[90,   499] loss: 0.0416
Accuracy of the network on the 100 tran images: 99.000 %
epoch 90 cost 261.263486 sec
[91,   499] loss: 0.0411
Accuracy of the network on the 100 tran images: 98.000 %
epoch 91 cost 262.037283 sec
[92,   499] loss: 0.0426
Accuracy of the network on the 100 tran images: 97.000 %
epoch 92 cost 261.453189 sec
[93,   499] loss: 0.0435
Accuracy of the network on the 100 tran images: 98.000 %
epoch 93 cost 261.335392 sec
[94,   499] loss: 0.0392
Accuracy of the network on the 100 tran images: 99.000 %
epoch 94 cost 261.731063 sec
[95,   499] loss: 0.0405
Accuracy of the network on the 100 tran images: 98.000 %
epoch 95 cost 261.374457 sec
[96,   499] loss: 0.0401
Accuracy of the network on the 100 tran images: 97.000 %
epoch 96 cost 261.369956 sec
[97,   499] loss: 0.0417
Accuracy of the network on the 100 tran images: 97.000 %
epoch 97 cost 261.744702 sec
[98,   499] loss: 0.0395
```

```
Accuracy of the network on the 100 tran images: 100.000 %
epoch 98 cost 261.334008 sec
[99,   499] loss: 0.0416
Accuracy of the network on the 100 tran images: 98.000 %
epoch 99 cost 261.583364 sec
Finished Training
Accuracy of the network on the 10000 test images: 84.050 %
```

In [ ]:

In [ ]: