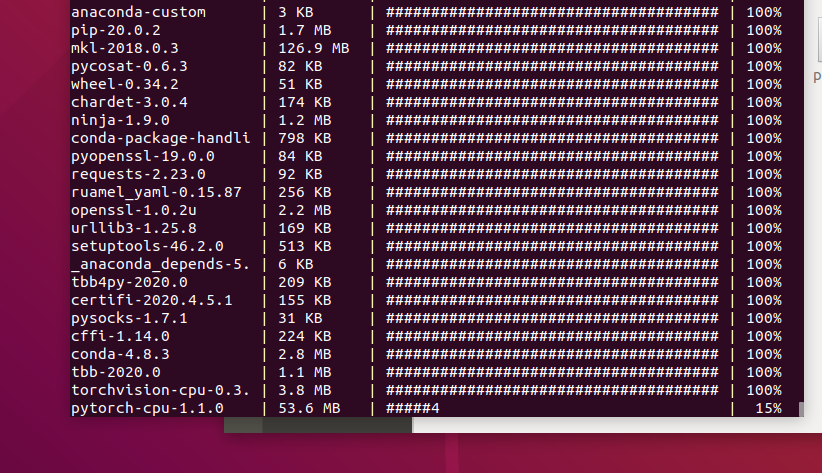
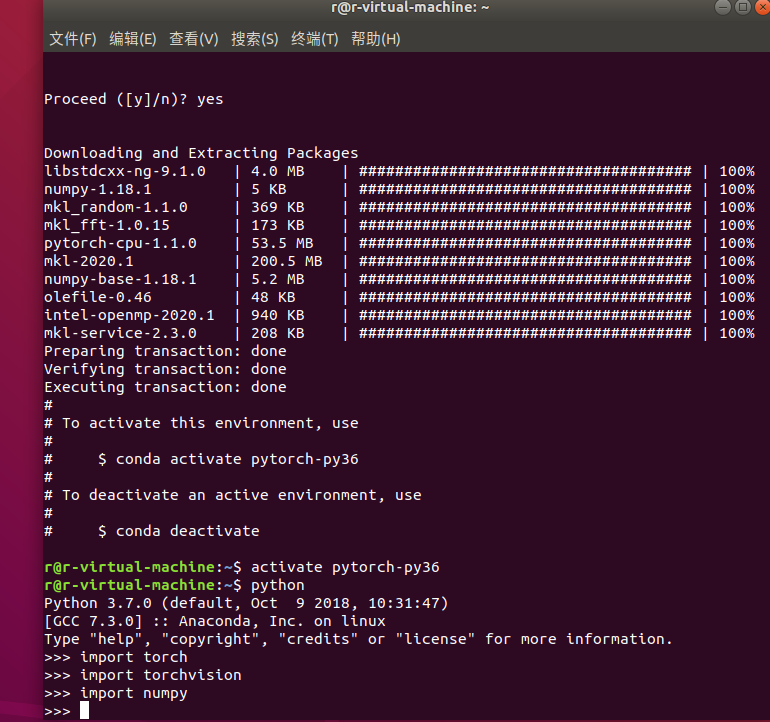
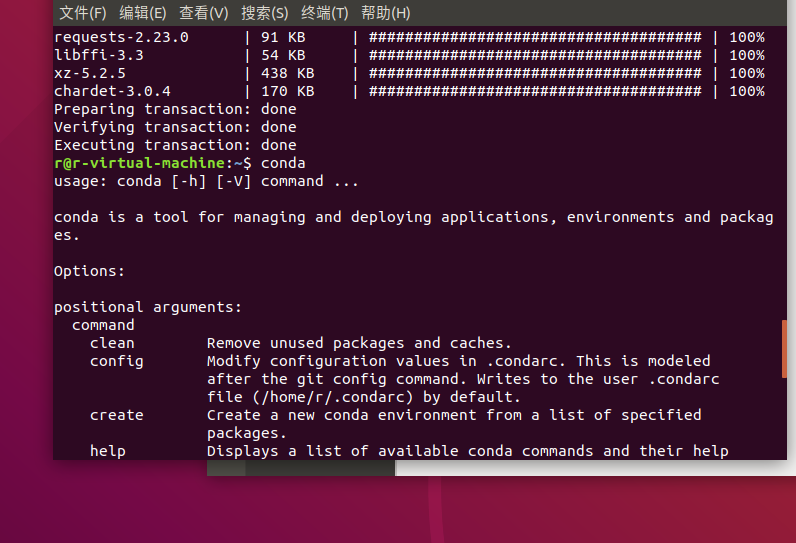
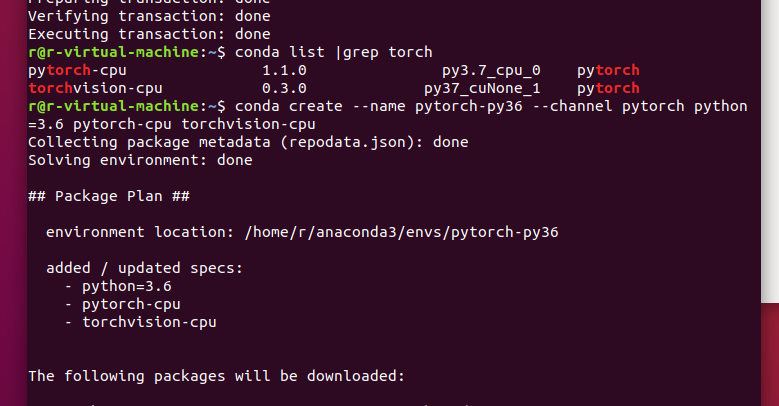
一 代码文件+配置环境

首先配置了pytroch





import torch

import torchvision

import torchvision.transforms as transforms

transform = transforms.Compose(

    [transforms.ToTensor(),

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

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)

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)

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

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

import torch.nn as nn

import torch.nn.functional as F

class Net(nn.Module):

    def \_\_init\_\_(self):

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

        self.conv1 = nn.Conv2d(3, 6, 5)

        self.pool = nn.MaxPool2d(2, 2)

        self.conv2 = nn.Conv2d(6, 16, 5)

        self.fc1 = nn.Linear(16 \* 5 \* 5, 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 = x.view(-1, 16 \* 5 \* 5)

        x = F.relu(self.fc1(x))

        x = F.relu(self.fc2(x))

        x = self.fc3(x)

        return x

net = Net()

import torch.optim as optim

criterion = nn.CrossEntropyLoss()

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

for epoch in range(2):  # loop over the dataset multiple times

    running\_loss = 0.0

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

        # get the inputs; data is a list of [inputs, labels]

        inputs, labels = data

        # zero the parameter gradients

        optimizer.zero\_grad()

        # forward + backward + optimize

        outputs = net(inputs)

        loss = criterion(outputs, labels)

        loss.backward()

        optimizer.step()

        # print statistics

        running\_loss += loss.item()

        if i % 2000 == 1999:    # print every 2000 mini-batches

            print('[%d, %5d] loss: %.3f' %

                  (epoch + 1, i + 1, running\_loss / 2000))

            running\_loss = 0.0

print('Finished Training')

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 10000 test images: %d %%' % (

    100 \* correct / total))

class\_correct = list(0. for i in range(10))

class\_total = list(0. for i in range(10))

with torch.no\_grad():

    for data in testloader:

        images, labels = data

        outputs = net(images)

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

        c = (predicted == labels).squeeze()

        for i in range(4):

            label = labels[i]

            class\_correct[label] += c[i].item()

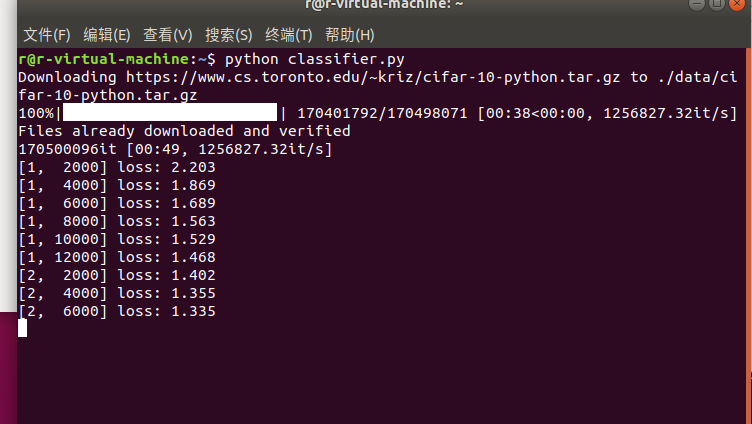
            class\_total[label] += 1

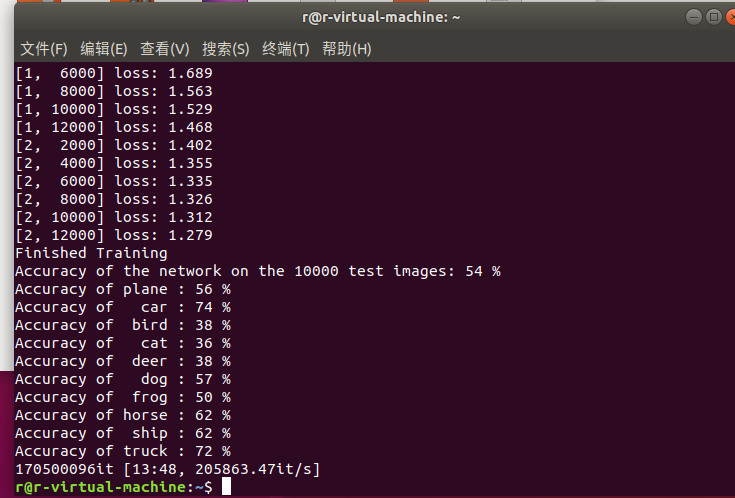
for i in range(10):

    print('Accuracy of %5s : %2d %%' % (

        classes[i], 100 \* class\_correct[i] / class\_total[i]))

进行了训练集





训练结束

r@r-virtual-machine:~$ python classifier.py

Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz

100%|███████████████████████▉| 170401792/170498071 [00:38<00:00, 1256827.32it/s]Files already downloaded and verified

170500096it [00:49, 1256827.32it/s] [1, 2000] loss: 2.203

[1, 4000] loss: 1.869

[1, 6000] loss: 1.689

[1, 8000] loss: 1.563

[1, 10000] loss: 1.529

[1, 12000] loss: 1.468

[2, 2000] loss: 1.402

[2, 4000] loss: 1.355

[2, 6000] loss: 1.335

[2, 8000] loss: 1.326

[2, 10000] loss: 1.312

[2, 12000] loss: 1.279

Finished Training

Accuracy of the network on the 10000 test images: 54 %

Accuracy of plane : 56 %

Accuracy of car : 74 %

Accuracy of bird : 38 %

Accuracy of cat : 36 %

Accuracy of deer : 38 %

Accuracy of dog : 57 %

Accuracy of frog : 50 %

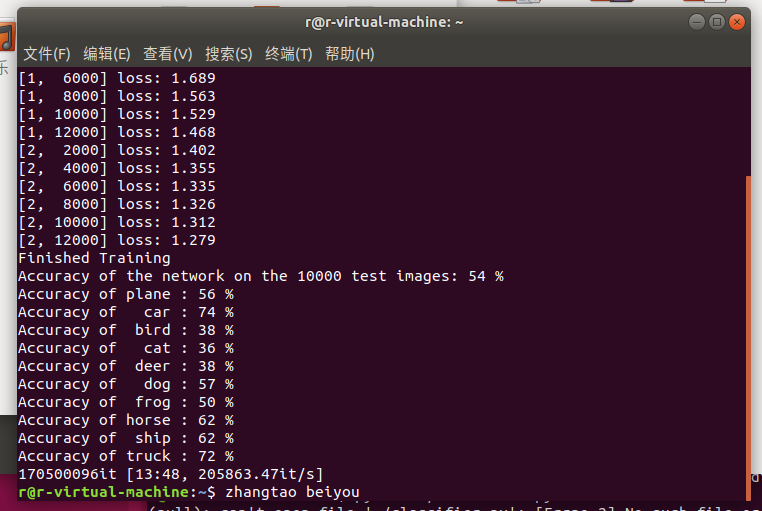
Accuracy of horse : 62 %

Accuracy of ship : 62 %

Accuracy of truck : 72 %

170500096it [13:48, 205863.47it/s]

r@r-virtual-machine:~$



最后命令行加上了自己的姓名+组名

