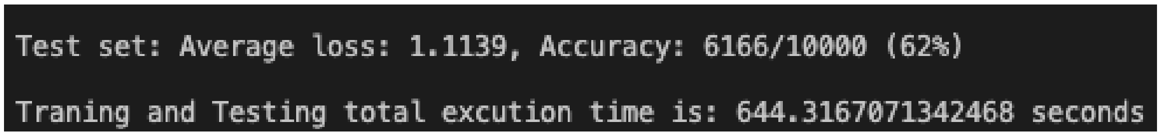
Homework Assignment 4

Chaoji Zuo chaoji.zuo@rutgers.edu
CZ296 / 190003416

Problem (Build LeNet for colorful image classification). In this problem, you are asked to train and test a neural network for entire CIFAR-10 colorful image dataset.

1. network without dropout/batch normalization:



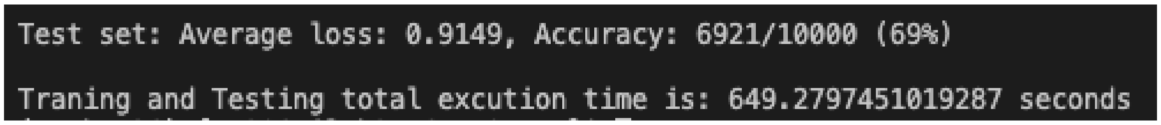
```
Test set: Average loss: 1.1139, Accuracy: 6166/10000 (62%)

Traning and Testing total excution time is: 644.3167071342468 seconds
```

Figure 1: original LeNet

2. network with one additional dropout layer:

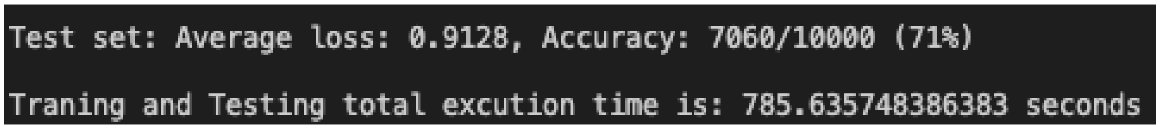
   I try to adjust the position of dropout layer in this network.



```
Test set: Average loss: 0.9149, Accuracy: 6921/10000 (69%)

Traning and Testing total excution time is: 649.2797451019287 seconds
```

Figure 2: dropout layer before FC layer



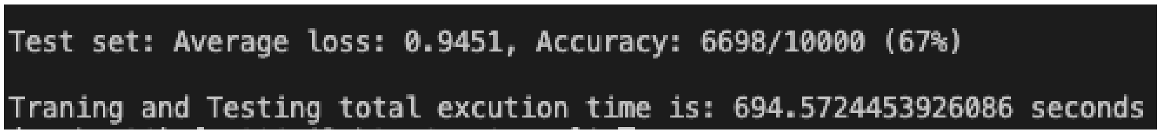
```
Test set: Average loss: 0.9128, Accuracy: 7060/10000 (71%)

Traning and Testing total excution time is: 785.635748386383 seconds
```

Figure 3: dropout layer after FC layer

3. network with one additional batch normalization:

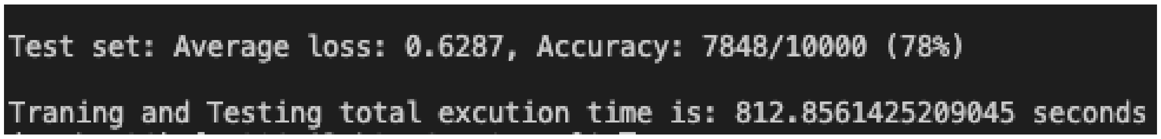
   I try to adjust the position of batch normalization layer in this network.



```
Test set: Average loss: 0.9451, Accuracy: 6698/10000 (67%)

Traning and Testing total excution time is: 694.5724453926086 seconds
```

Figure 4: batch normalization layer before FC layer



```
Test set: Average loss: 0.6287, Accuracy: 7848/10000 (78%)

Traning and Testing total excution time is: 812.8561425209045 seconds
```

Figure 5: dropout layer before ReLU layer

sample code here: I attach the whole code files in attachment.

```

class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        ##############################
        #### Put your code here ####
        ##############################
        self.convnet = nn.Sequential(OrderedDict([
            ('c1',nn.Conv2d(3,18,kernel_size=(5,5))),
            ('relu1',nn.ReLU()),
            ('s2',nn.MaxPool2d(kernel_size=(2,2),stride=2)),
            ('c3',nn.Conv2d(18,48,kernel_size=(5,5))),
            ('relu3',nn.ReLU()),
            ('s4',nn.MaxPool2d(kernel_size=(2,2),stride=2)),
            ('c5',nn.Conv2d(48,360,kernel_size=(5,5))),
            ('batch_norm',nn.BatchNorm2d(360)),
            ('relu5',nn.ReLU())
            ]))
        self.fc = nn.Sequential(OrderedDict([
            ('f6',nn.Linear(360,84)),
            ('relu6',nn.ReLU()),
            ('f7',nn.Linear(84,10)),
            ('sig7',nn.LogSoftmax(dim=-1))
        ]))
        ############################
        #### End of your codes ####
        ############################
    def forward(self, x):
        ##############################
        #### Put your code here ####
        ##############################
        x_convnet = self.convnet(x)
        #print(x_convnet.size())
        x_convnet = x_convnet.view((-1,x_convnet.size()[1]))
        #print(x_convnet.size())
        x_fc = self.fc(x_convnet)
        out = x_fc
        ############################
        #### End of your codes ####
        ############################
        return out
def train(model, device, train_loader, optimizer, epoch):
    model.train()
    count = 0
    criterion = nn.NLLLoss()
    for batch_idx, (data, target) in enumerate(train_loader):
        #print(data.shape)
        data, target = data.to(device), target.to(device)
        #############################
        #### Put your code here ####
        #############################
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output,target)
        loss.backward()
        optimizer.step()
        loss_list.append(loss.item())
        ############################
        #### End of your codes ####
        ############################
        if batch_idx % 10 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
```