Deep Learning Homework IV Report

Name: Dazhi Li
RUID: 197007456

I create 3 different model and manually switch the model I want to use. Here I will only post how do I build these models in source code. Specific source code could be viewed in my attachment.

(1)LeNet5(classic)

Source code:

```
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        ############################
        #### Put your code here ####
        ############################
        self.convnet=nn.Sequential(OrderedDict([
                ('c1',nn.Conv2d(3,6,kernel_size=(5,5))),
                ('relu1',nn.ReLU()),
                ('s2',nn.MaxPool2d(kernel_size=(2,2),stride=2)),
                ('c3',nn.Conv2d(6,16,kernel_size=(5,5))),
                ('relu3',nn.ReLU()),
                ('s4',nn.MaxPool2d(kernel_size=(2,2),stride=2)),
                ('c5',nn.Conv2d(16,120,kernel_size=(5,5))),
                ('relu5',nn.ReLU())
        ]))

        self.fc=nn.Sequential(OrderedDict([
                ('f6',nn.Linear(120,84)),
                ('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.view(128,3,32,32)
        x=self.convnet(x)
        x=x.view(x.size()[0],-1)
        out=self.fc(x)
```

```
        ############################
        #### End of your codes ####
        ############################


        return out
```

Running result:

```
Train Epoch: 50 [19200/50000 (38%)]     Loss: 1.123232
Train Epoch: 50 [20480/50000 (41%)]     Loss: 1.112376
Train Epoch: 50 [21760/50000 (43%)]     Loss: 1.211237
Train Epoch: 50 [23040/50000 (46%)]     Loss: 1.231867
Train Epoch: 50 [24320/50000 (49%)]     Loss: 1.517631
Train Epoch: 50 [25600/50000 (51%)]     Loss: 1.207361
Train Epoch: 50 [26880/50000 (54%)]     Loss: 1.236603
Train Epoch: 50 [28160/50000 (56%)]     Loss: 1.289385
Train Epoch: 50 [29440/50000 (59%)]     Loss: 1.415432
Train Epoch: 50 [30720/50000 (61%)]     Loss: 1.159418
Train Epoch: 50 [32000/50000 (64%)]     Loss: 1.485812
Train Epoch: 50 [33280/50000 (66%)]     Loss: 1.329686
Train Epoch: 50 [34560/50000 (69%)]     Loss: 1.228698
Train Epoch: 50 [35840/50000 (72%)]     Loss: 1.063656
Train Epoch: 50 [37120/50000 (74%)]     Loss: 1.398037
Train Epoch: 50 [38400/50000 (77%)]     Loss: 1.134353
Train Epoch: 50 [39680/50000 (79%)]     Loss: 1.288119
Train Epoch: 50 [40960/50000 (82%)]     Loss: 1.167127
Train Epoch: 50 [42240/50000 (84%)]     Loss: 1.351845
Train Epoch: 50 [43520/50000 (87%)]     Loss: 1.266286
Train Epoch: 50 [44800/50000 (90%)]     Loss: 1.379644
Train Epoch: 50 [46080/50000 (92%)]     Loss: 1.333201
Train Epoch: 50 [47360/50000 (95%)]     Loss: 1.350057
Train Epoch: 50 [48640/50000 (97%)]     Loss: 1.350939
Train Epoch: 50 [31200/50000 (100%)]    Loss: 0.958694

Test set: Average loss: 1.1802, Accuracy: 6025/10000 (60%)

Traning and Testing total excution time is: 648.3741579055786 seconds
```

Fig 1-1

(2) LeNet5 with Dropout

Dropout rate=0.5, inserted between the first fully-connected layer and the ReLU activating function.

Source code:

```
class LeNet_dropout(nn.Module):
    def __init__(self):
        super(LeNet_dropout, self).__init__()
        self.convnet=nn.Sequential(OrderedDict([
                ('c1',nn.Conv2d(3,6,kernel_size=(5,5))),
                ('relu1',nn.ReLU()),
                ('s2',nn.MaxPool2d(kernel_size=(2,2),stride=2)),
                ('c3',nn.Conv2d(6,16,kernel_size=(5,5))),
                ('relu3',nn.ReLU()),
                ('s4',nn.MaxPool2d(kernel_size=(2,2),stride=2)),
```

```
                ('c5',nn.Conv2d(16,120,kernel_size=(5,5))),
                ('relu5',nn.ReLU())
        ]))

        self.fc=nn.Sequential(OrderedDict([
                ('f6',nn.Linear(120,84)),
                ('drop6',nn.Dropout(0.5)),
                ('relu6',nn.ReLU()),
                ('f7',nn.Linear(84,10)),
                ('sig7',nn.LogSoftmax(dim=-1))
        ]))
    def forward(self, x):
        #x.view(128,3,32,32)
        x=self.convnet(x)
        x=x.view(x.size()[0],-1)
        out=self.fc(x)
        return out
```

Running result:

```
Train Epoch: 50 [19200/50000 (38%)]     Loss: 1.355727
Train Epoch: 50 [20480/50000 (41%)]     Loss: 1.483970
Train Epoch: 50 [21760/50000 (43%)]     Loss: 1.264506
Train Epoch: 50 [23040/50000 (46%)]     Loss: 1.268804
Train Epoch: 50 [24320/50000 (49%)]     Loss: 1.351647
Train Epoch: 50 [25600/50000 (51%)]     Loss: 1.368118
Train Epoch: 50 [26880/50000 (54%)]     Loss: 1.282315
Train Epoch: 50 [28160/50000 (56%)]     Loss: 1.422349
Train Epoch: 50 [29440/50000 (59%)]     Loss: 1.519642
Train Epoch: 50 [30720/50000 (61%)]     Loss: 1.242310
Train Epoch: 50 [32000/50000 (64%)]     Loss: 1.579738
Train Epoch: 50 [33280/50000 (66%)]     Loss: 1.359537
Train Epoch: 50 [34560/50000 (69%)]     Loss: 1.297768
Train Epoch: 50 [35840/50000 (72%)]     Loss: 1.267301
Train Epoch: 50 [37120/50000 (74%)]     Loss: 1.384160
Train Epoch: 50 [38400/50000 (77%)]     Loss: 1.311643
Train Epoch: 50 [39680/50000 (79%)]     Loss: 1.504042
Train Epoch: 50 [40960/50000 (82%)]     Loss: 1.312995
Train Epoch: 50 [42240/50000 (84%)]     Loss: 1.401075
Train Epoch: 50 [43520/50000 (87%)]     Loss: 1.400843
Train Epoch: 50 [44800/50000 (90%)]     Loss: 1.344902
Train Epoch: 50 [46080/50000 (92%)]     Loss: 1.517275
Train Epoch: 50 [47360/50000 (95%)]     Loss: 1.078959
Train Epoch: 50 [48640/50000 (97%)]     Loss: 1.612176
Train Epoch: 50 [31200/50000 (100%)]    Loss: 1.301404

Test set: Average loss: 1.3272, Accuracy: 5428/10000 (54%)

Traning and Testing total excution time is: 637.2968056201935 seconds
```

Fig 1-2

(3) LeNet5 with batch normalization

Batch normalization layer is inserted between the second convolution layer and the following ReLU activating function.

Source code:

```
class LeNet_batchnormalized(nn.Module):
    def __init__(self):
        super(LeNet_batchnormalized, self).__init__()
        self.convnet=nn.Sequential(OrderedDict([
                ('c1',nn.Conv2d(3,6,kernel_size=(5,5))),
                ('relu1',nn.ReLU()),
                ('s2',nn.MaxPool2d(kernel_size=(2,2),stride=2)),
                ('c3',nn.Conv2d(6,16,kernel_size=(5,5))),
                ('bn3',nn.BatchNorm2d(16)),
                ('relu3',nn.ReLU()),
                ('s4',nn.MaxPool2d(kernel_size=(2,2),stride=2)),
                ('c5',nn.Conv2d(16,120,kernel_size=(5,5))),
                ('relu5',nn.ReLU())
        ]))

        self.fc=nn.Sequential(OrderedDict([
                ('f6',nn.Linear(120,84)),
                ('relu6',nn.ReLU()),
                ('f7',nn.Linear(84,10)),
                ('sig7',nn.LogSoftmax(dim=-1))
        ]))
    def forward(self, x):
        #x.view(128,3,32,32)
        x=self.convnet(x)
        x=x.view(x.size()[0],-1)
        out=self.fc(x)
        return out
```

Running result:

```
Train Epoch: 50 [19200/50000 (38%)]     Loss: 0.950440
Train Epoch: 50 [20480/50000 (41%)]     Loss: 0.994719
Train Epoch: 50 [21760/50000 (43%)]     Loss: 0.951769
Train Epoch: 50 [23040/50000 (46%)]     Loss: 0.899475
Train Epoch: 50 [24320/50000 (49%)]     Loss: 0.797735
Train Epoch: 50 [25600/50000 (51%)]     Loss: 0.924349
Train Epoch: 50 [26880/50000 (54%)]     Loss: 1.045191
Train Epoch: 50 [28160/50000 (56%)]     Loss: 1.103298
Train Epoch: 50 [29440/50000 (59%)]     Loss: 1.054363
Train Epoch: 50 [30720/50000 (61%)]     Loss: 0.822823
Train Epoch: 50 [32000/50000 (64%)]     Loss: 1.041641
Train Epoch: 50 [33280/50000 (66%)]     Loss: 1.116076
Train Epoch: 50 [34560/50000 (69%)]     Loss: 0.961281
Train Epoch: 50 [35840/50000 (72%)]     Loss: 0.758461
Train Epoch: 50 [37120/50000 (74%)]     Loss: 0.944091
Train Epoch: 50 [38400/50000 (77%)]     Loss: 0.879171
Train Epoch: 50 [39680/50000 (79%)]     Loss: 1.224752
Train Epoch: 50 [40960/50000 (82%)]     Loss: 1.005856
Train Epoch: 50 [42240/50000 (84%)]     Loss: 0.899253
Train Epoch: 50 [43520/50000 (87%)]     Loss: 0.852403
Train Epoch: 50 [44800/50000 (90%)]     Loss: 0.960315
Train Epoch: 50 [46080/50000 (92%)]     Loss: 1.087994
Train Epoch: 50 [47360/50000 (95%)]     Loss: 1.090542
Train Epoch: 50 [48640/50000 (97%)]     Loss: 1.068784
Train Epoch: 50 [31200/50000 (100%)]    Loss: 0.837866

Test set: Average loss: 0.8817, Accuracy: 6931/10000 (69%)

Traning and Testing total excution time is: 641.7426285743713 seconds
```

Fig 1-3