**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:

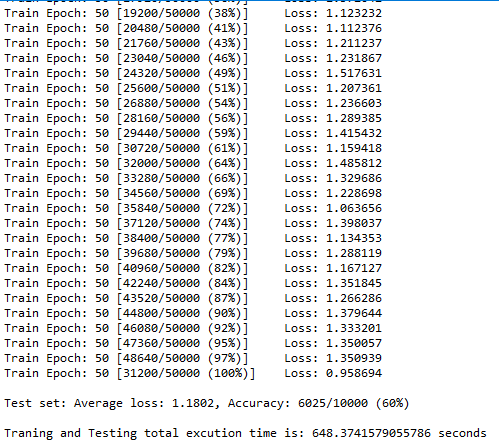


Fig 1-1

(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:

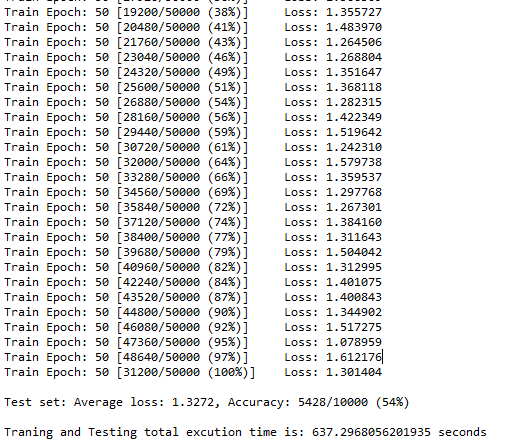


Fig 1-2

(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:

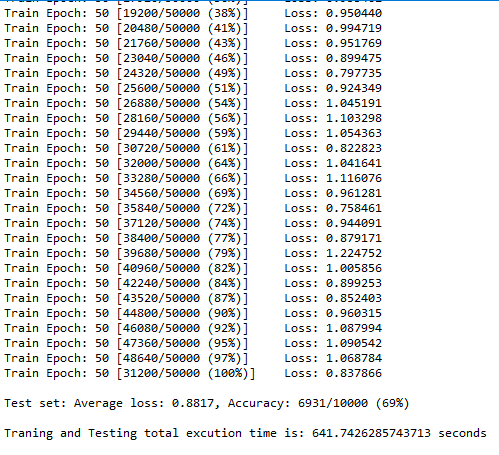


Fig 1-3