Spring 2020
Introduction to Deep Learning
Homework Assignment 4

Problem : Build LeNet for colorful image classification

Performance Requirement : The test accuracy should achieve above 50%

Sub-Parts to the Problem :

1) Network without dropout/batch normalization,

2) Network with one additional dropout layer and

3) Network with one additional batch normalization over normal network (1)

4) Network with additional batch Normalization Layer over Network with one additional dropout layer

Part 1:

LeNet without dropout/batch normalization

Below is the snippet of the Le Net created using Pytorch Libraries(Working Code has been attached)

```
class LeNet(nn.Module):

    def __init__(self):

        super(LeNet, self).__init__()
        #Sequential Model for the Convolution Neural Network
        self.convnet=nn.Sequential(nn.Conv2d(3, 6, 5),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(6, 16, 
5),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(16, 120, 5),nn.ReLU())
        #Sequential Model for the Fully Connected Layers ahead of the Convolution Part
        self.fc=nn.Sequential(nn.Linear(120,84),nn.ReLU(),nn.Linear(84,10),nn.LogSoftmax(dim=-1))


    def forward(self, x):
        out = self.convnet(x)
        out = self.fc(out.view(-1,120))
        return out
```

Observation:

We have created LeNet Network:

Layer 1: Convolution Layer

Input is : 32* 32*3 RGB Images

With Layer Constituted of 5*5 filter with 6 in numbers(dimension) with RELU activation and the Max Pool of 2* 2

Layer 2: Convolution Layer

Input is of 6 in dimension (third dimension as of the number of filters)

With Layer Constituted of 5*5 filter with 16 in numbers(dimension) with RELU activation and the Max Pool of 2* 2

Layer3: Convolution Layer

Input is of 16 in dimension (third dimension as of the number of filters)

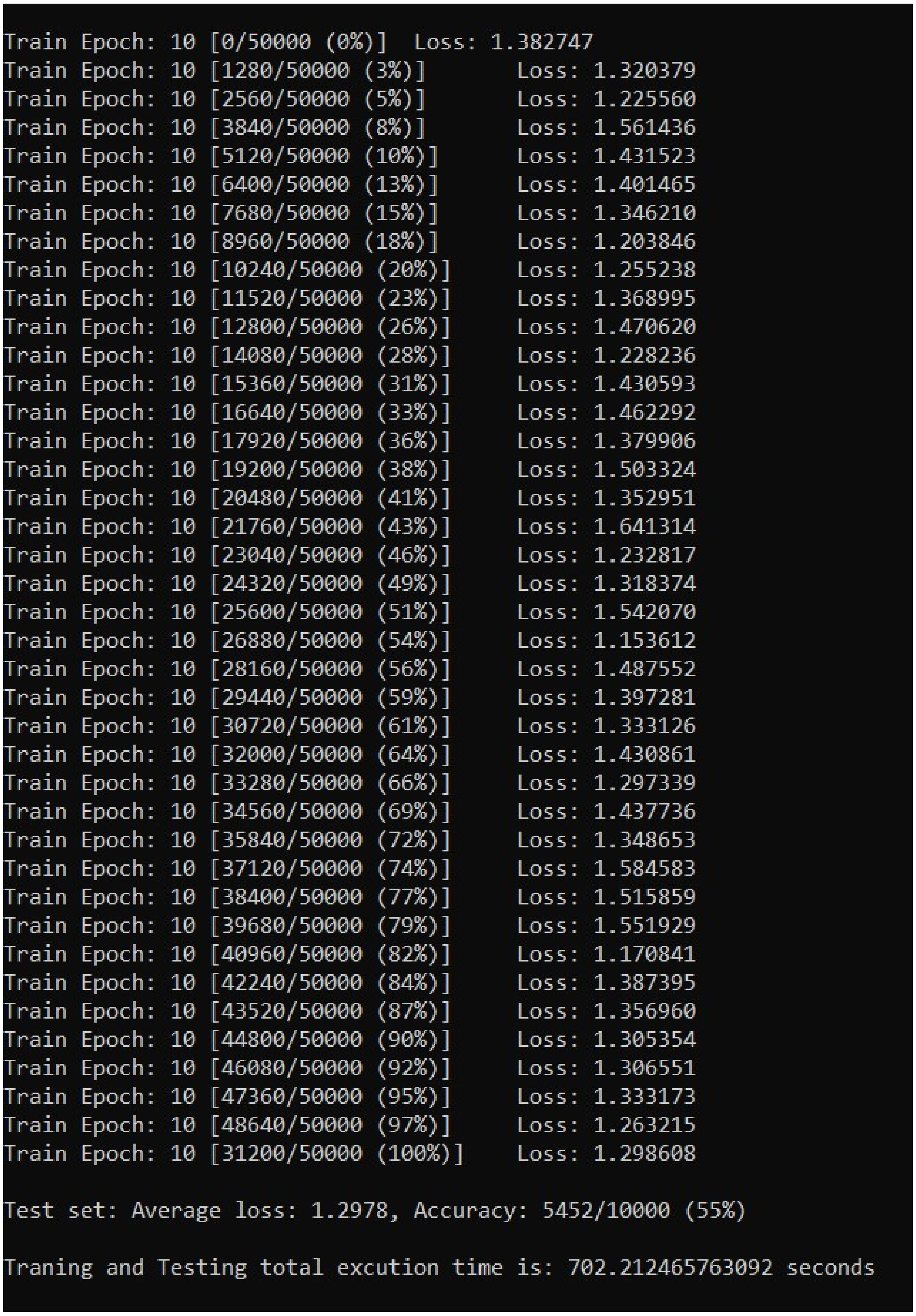
With Layer Constituted of 5*5 filter with 120 in numbers(dimension) with RELU activation

After Convolution there are : Fully Connected Layers

Layer 4: 120 inputs and 84 outputs with RELU activation

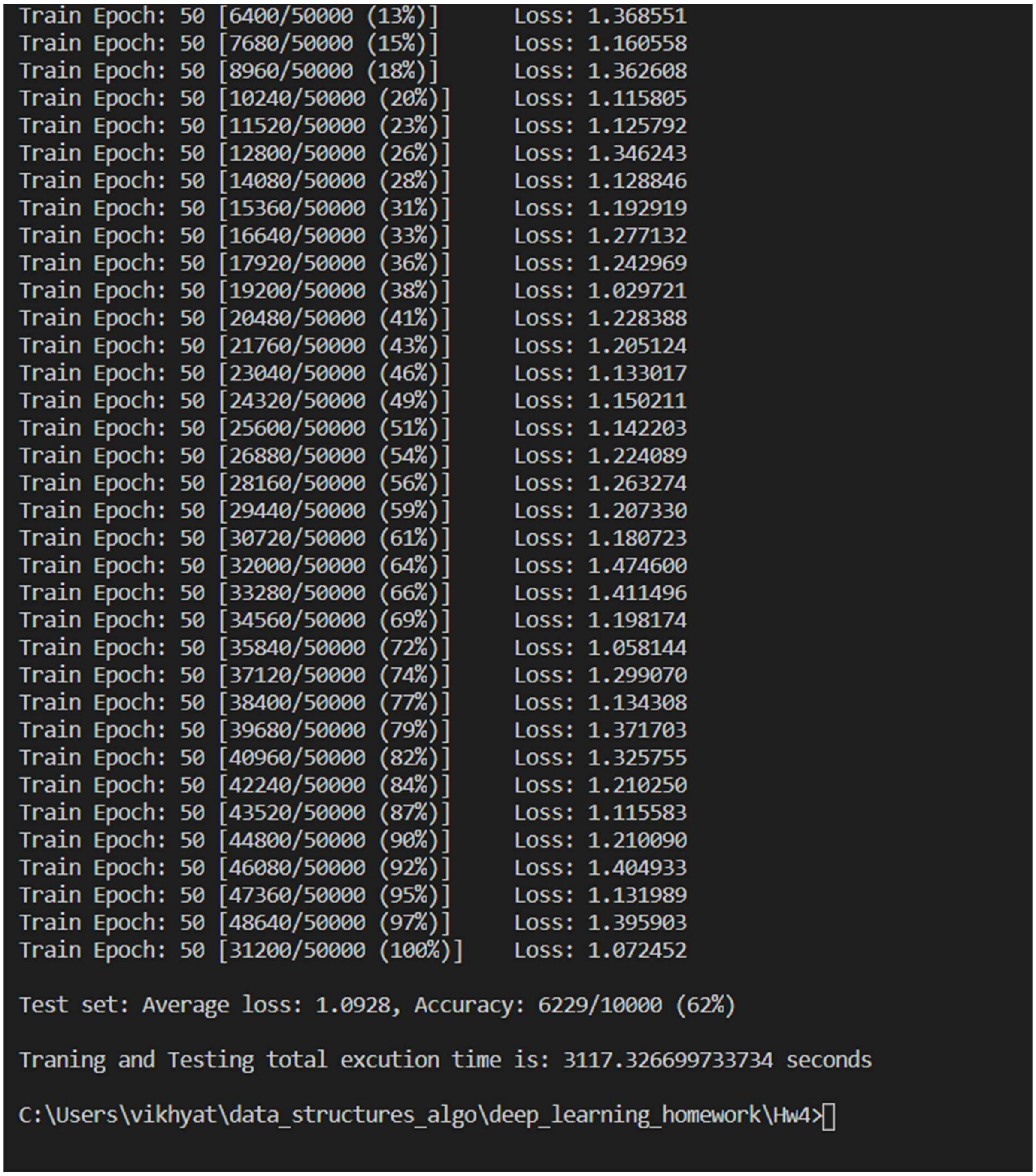
Layer 5: 84 inputs and 10 classified outputs with SoftMax Activation

Results:(Code with lenet_cifar_fill.py) In less than 10 epochs more than 55 percent accuracy



```
Train Epoch: 10 [0/50000 (0%)]  Loss: 1.382747
Train Epoch: 10 [1280/50000 (3%)]       Loss: 1.320379
Train Epoch: 10 [2560/50000 (5%)]       Loss: 1.225560
Train Epoch: 10 [3840/50000 (8%)]       Loss: 1.561436
Train Epoch: 10 [5120/50000 (10%)]      Loss: 1.431523
Train Epoch: 10 [6400/50000 (13%)]      Loss: 1.401465
Train Epoch: 10 [7680/50000 (15%)]      Loss: 1.346210
Train Epoch: 10 [8960/50000 (18%)]      Loss: 1.203846
Train Epoch: 10 [10240/50000 (20%)]     Loss: 1.255238
Train Epoch: 10 [11520/50000 (23%)]     Loss: 1.368995
Train Epoch: 10 [12800/50000 (26%)]     Loss: 1.470620
Train Epoch: 10 [14080/50000 (28%)]     Loss: 1.228236
Train Epoch: 10 [15360/50000 (31%)]     Loss: 1.430593
Train Epoch: 10 [16640/50000 (33%)]     Loss: 1.462292
Train Epoch: 10 [17920/50000 (36%)]     Loss: 1.379906
Train Epoch: 10 [19200/50000 (38%)]     Loss: 1.503324
Train Epoch: 10 [20480/50000 (41%)]     Loss: 1.352951
Train Epoch: 10 [21760/50000 (43%)]     Loss: 1.641314
Train Epoch: 10 [23040/50000 (46%)]     Loss: 1.232817
Train Epoch: 10 [24320/50000 (49%)]     Loss: 1.318374
Train Epoch: 10 [25600/50000 (51%)]     Loss: 1.542070
Train Epoch: 10 [26880/50000 (54%)]     Loss: 1.153612
Train Epoch: 10 [28160/50000 (56%)]     Loss: 1.487552
Train Epoch: 10 [29440/50000 (59%)]     Loss: 1.397281
Train Epoch: 10 [30720/50000 (61%)]     Loss: 1.333126
Train Epoch: 10 [32000/50000 (64%)]     Loss: 1.430861
Train Epoch: 10 [33280/50000 (66%)]     Loss: 1.297339
Train Epoch: 10 [34560/50000 (69%)]     Loss: 1.437736
Train Epoch: 10 [35840/50000 (72%)]     Loss: 1.348653
Train Epoch: 10 [37120/50000 (74%)]     Loss: 1.584583
Train Epoch: 10 [38400/50000 (77%)]     Loss: 1.515859
Train Epoch: 10 [39680/50000 (79%)]     Loss: 1.551929
Train Epoch: 10 [40960/50000 (82%)]     Loss: 1.170841
Train Epoch: 10 [42240/50000 (84%)]     Loss: 1.387395
Train Epoch: 10 [43520/50000 (87%)]     Loss: 1.356960
Train Epoch: 10 [44800/50000 (90%)]     Loss: 1.305354
Train Epoch: 10 [46080/50000 (92%)]     Loss: 1.306551
Train Epoch: 10 [47360/50000 (95%)]     Loss: 1.333173
Train Epoch: 10 [48640/50000 (97%)]     Loss: 1.263215
Train Epoch: 10 [31200/50000 (100%)]    Loss: 1.298608

Test set: Average loss: 1.2978, Accuracy: 5452/10000 (55%)

Traning and Testing total excution time is: 702.212465763092 seconds
```

In less than 50 epochs more than 62 percent accuracy

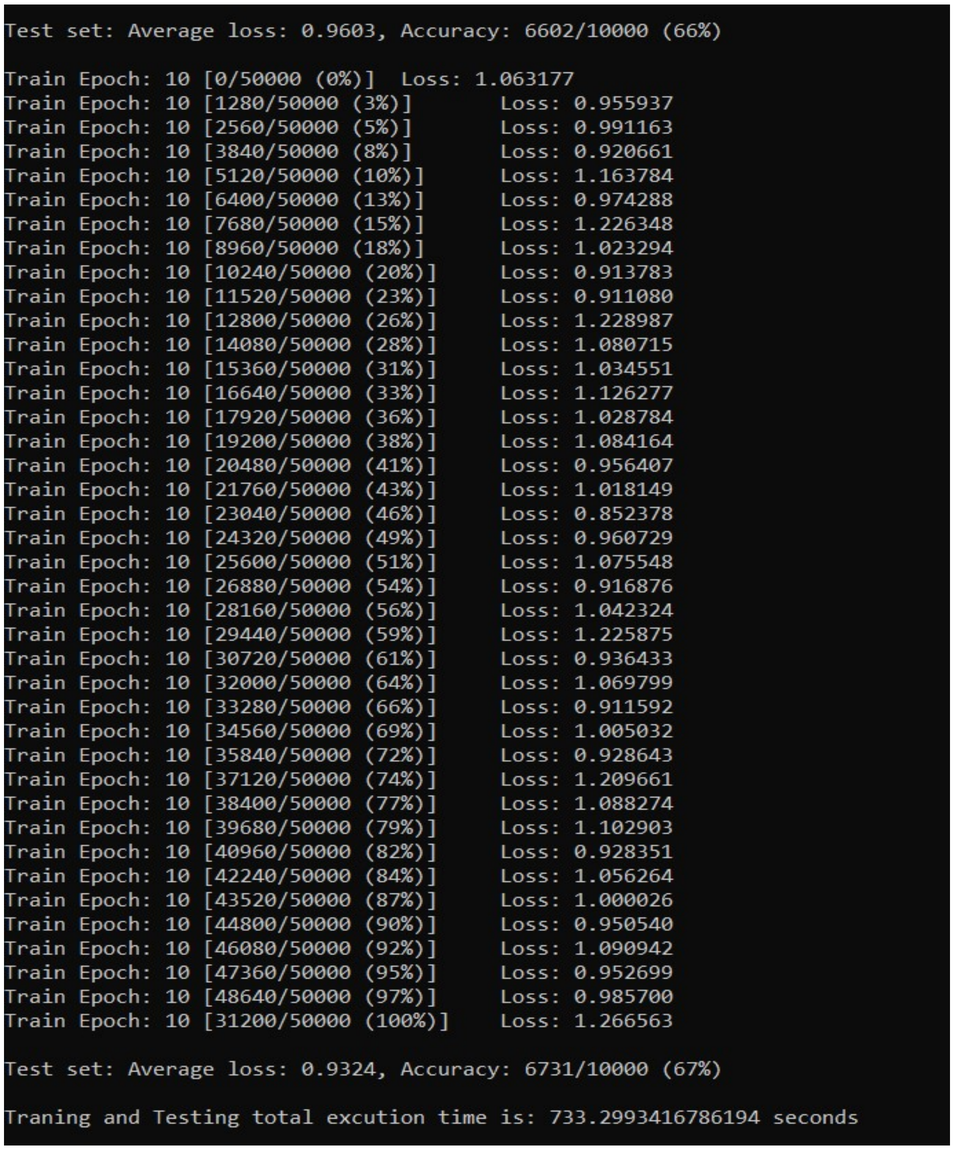


```
Train Epoch: 50 [6400/50000 (13%)]      Loss: 1.368551
Train Epoch: 50 [7680/50000 (15%)]      Loss: 1.160558
Train Epoch: 50 [8960/50000 (18%)]      Loss: 1.362608
Train Epoch: 50 [10240/50000 (20%)]     Loss: 1.115805
Train Epoch: 50 [11520/50000 (23%)]     Loss: 1.125792
Train Epoch: 50 [12800/50000 (26%)]     Loss: 1.346243
Train Epoch: 50 [14080/50000 (28%)]     Loss: 1.128846
Train Epoch: 50 [15360/50000 (31%)]     Loss: 1.192919
Train Epoch: 50 [16640/50000 (33%)]     Loss: 1.277132
Train Epoch: 50 [17920/50000 (36%)]     Loss: 1.242969
Train Epoch: 50 [19200/50000 (38%)]     Loss: 1.029721
Train Epoch: 50 [20480/50000 (41%)]     Loss: 1.228388
Train Epoch: 50 [21760/50000 (43%)]     Loss: 1.205124
Train Epoch: 50 [23040/50000 (46%)]     Loss: 1.133017
Train Epoch: 50 [24320/50000 (49%)]     Loss: 1.150211
Train Epoch: 50 [25600/50000 (51%)]     Loss: 1.142203
Train Epoch: 50 [26880/50000 (54%)]     Loss: 1.224089
Train Epoch: 50 [28160/50000 (56%)]     Loss: 1.263274
Train Epoch: 50 [29440/50000 (59%)]     Loss: 1.207330
Train Epoch: 50 [30720/50000 (61%)]     Loss: 1.180723
Train Epoch: 50 [32000/50000 (64%)]     Loss: 1.474600
Train Epoch: 50 [33280/50000 (66%)]     Loss: 1.411496
Train Epoch: 50 [34560/50000 (69%)]     Loss: 1.198174
Train Epoch: 50 [35840/50000 (72%)]     Loss: 1.058144
Train Epoch: 50 [37120/50000 (74%)]     Loss: 1.299070
Train Epoch: 50 [38400/50000 (77%)]     Loss: 1.134308
Train Epoch: 50 [39680/50000 (79%)]     Loss: 1.371703
Train Epoch: 50 [40960/50000 (82%)]     Loss: 1.325755
Train Epoch: 50 [42240/50000 (84%)]     Loss: 1.210250
Train Epoch: 50 [43520/50000 (87%)]     Loss: 1.115583
Train Epoch: 50 [44800/50000 (90%)]     Loss: 1.210090
Train Epoch: 50 [46080/50000 (92%)]     Loss: 1.404933
Train Epoch: 50 [47360/50000 (95%)]     Loss: 1.131989
Train Epoch: 50 [48640/50000 (97%)]     Loss: 1.395903
Train Epoch: 50 [31200/50000 (100%)]    Loss: 1.072452

Test set: Average loss: 1.0928, Accuracy: 6229/10000 (62%)

Traning and Testing total excution time is: 3117.326699733734 seconds

C:\Users\vikhyat\data_structures_algo\deep_learning_homework\Hw4>
```

Part 3: Network with one additional Batch Normalization Layer

```
class LeNet(nn.Module):

    def __init__(self):

        super(LeNet, self).__init__()
        #Sequential Model for the Convolution Neural Network
        self.convnet=nn.Sequential(nn.Conv2d(3, 6,
5),nn.BatchNorm2d(6),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(6, 16,
5),nn.BatchNorm2d(16),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(16, 120,
5),nn.BatchNorm2d(120),nn.ReLU())
        #Sequential Model for the Fully Connected Layers ahead of the Convolution Part

self.fc=nn.Sequential(nn.Linear(120,84),nn.BatchNorm1d(84),nn.ReLU(),nn.Linear(84,10),nn.BatchNorm
1d(10),nn.LogSoftmax(dim=-1))


    def forward(self, x):
        out = self.convnet(x)
        out = self.fc(out.view(-1,120))
        return out
```

Case 1: I have added Batch Normalization layer with each layer of CNN : **(Code lenet_cifar_fill_3.py)**

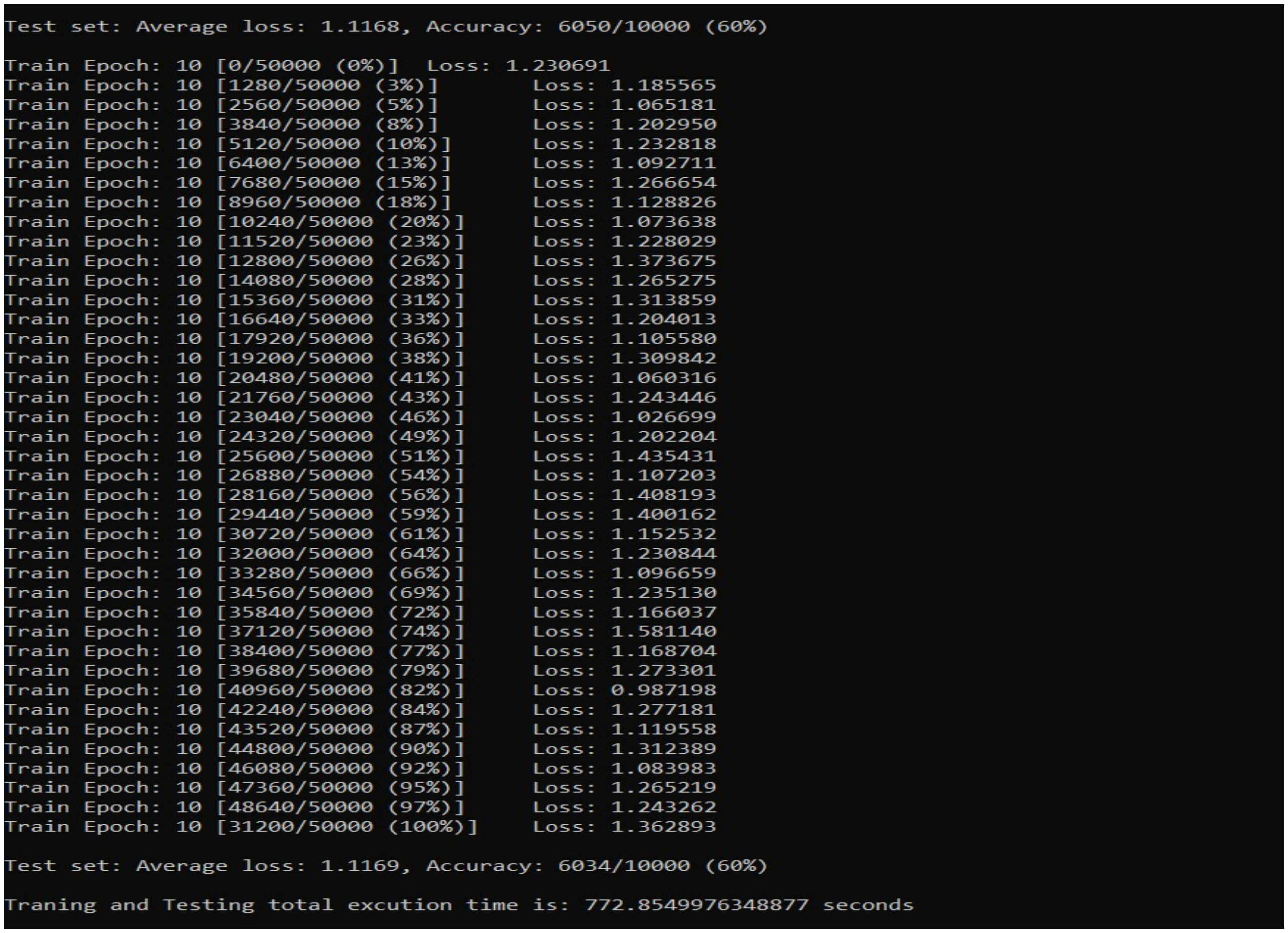


```
Test set: Average loss: 0.9603, Accuracy: 6602/10000 (66%)

Train Epoch: 10 [0/50000 (0%)]  Loss: 1.063177
Train Epoch: 10 [1280/50000 (3%)]       Loss: 0.955937
Train Epoch: 10 [2560/50000 (5%)]       Loss: 0.991163
Train Epoch: 10 [3840/50000 (8%)]       Loss: 0.920661
Train Epoch: 10 [5120/50000 (10%)]      Loss: 1.163784
Train Epoch: 10 [6400/50000 (13%)]      Loss: 0.974288
Train Epoch: 10 [7680/50000 (15%)]      Loss: 1.226348
Train Epoch: 10 [8960/50000 (18%)]      Loss: 1.023294
Train Epoch: 10 [10240/50000 (20%)]     Loss: 0.913783
Train Epoch: 10 [11520/50000 (23%)]     Loss: 0.911080
Train Epoch: 10 [12800/50000 (26%)]     Loss: 1.228987
Train Epoch: 10 [14080/50000 (28%)]     Loss: 1.080715
Train Epoch: 10 [15360/50000 (31%)]     Loss: 1.034551
Train Epoch: 10 [16640/50000 (33%)]     Loss: 1.126277
Train Epoch: 10 [17920/50000 (36%)]     Loss: 1.028784
Train Epoch: 10 [19200/50000 (38%)]     Loss: 1.084164
Train Epoch: 10 [20480/50000 (41%)]     Loss: 0.956407
Train Epoch: 10 [21760/50000 (43%)]     Loss: 1.018149
Train Epoch: 10 [23040/50000 (46%)]     Loss: 0.852378
Train Epoch: 10 [24320/50000 (49%)]     Loss: 0.960729
Train Epoch: 10 [25600/50000 (51%)]     Loss: 1.075548
Train Epoch: 10 [26880/50000 (54%)]     Loss: 0.916876
Train Epoch: 10 [28160/50000 (56%)]     Loss: 1.042324
Train Epoch: 10 [29440/50000 (59%)]     Loss: 1.225875
Train Epoch: 10 [30720/50000 (61%)]     Loss: 0.936433
Train Epoch: 10 [32000/50000 (64%)]     Loss: 1.069799
Train Epoch: 10 [33280/50000 (66%)]     Loss: 0.911592
Train Epoch: 10 [34560/50000 (69%)]     Loss: 1.005032
Train Epoch: 10 [35840/50000 (72%)]     Loss: 0.928643
Train Epoch: 10 [37120/50000 (74%)]     Loss: 1.209661
Train Epoch: 10 [38400/50000 (77%)]     Loss: 1.088274
Train Epoch: 10 [39680/50000 (79%)]     Loss: 1.102903
Train Epoch: 10 [40960/50000 (82%)]     Loss: 0.928351
Train Epoch: 10 [42240/50000 (84%)]     Loss: 1.056264
Train Epoch: 10 [43520/50000 (87%)]     Loss: 1.000026
Train Epoch: 10 [44800/50000 (90%)]     Loss: 0.950540
Train Epoch: 10 [46080/50000 (92%)]     Loss: 1.090942
Train Epoch: 10 [47360/50000 (95%)]     Loss: 0.952699
Train Epoch: 10 [48640/50000 (97%)]     Loss: 0.985700
Train Epoch: 10 [31200/50000 (100%)]    Loss: 1.266563

Test set: Average loss: 0.9324, Accuracy: 6731/10000 (67%)

Traning and Testing total excution time is: 733.2993416786194 seconds
```

Observation: With Batch Normalization , In just 10 epochs we have achieved the accuracy which we could not achieve in the 50 epochs of the normal CNN. So , Batch Normalization has brought considerable improvement in the CNN.

Case 2 : I have added Batch Normalization layer with each layer of CNN : (Code lenet_cifar_fill_3_1.py)

```
class LeNet(nn.Module):
    def __init__(self):

        super(LeNet, self).__init__()
        #Sequential Model for the Convolution Neural Network
        self.convnet=nn.Sequential(nn.Conv2d(3, 6,
5),nn.BatchNorm2d(6),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(6, 16,
5),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(16, 120, 5),nn.ReLU())
        #Sequential Model for the Fully Connected Layers ahead of the Convolution Part
        self.fc=nn.Sequential(nn.Linear(120,84),nn.ReLU(),nn.Linear(84,10),nn.LogSoftmax(dim=-1))

    def forward(self, x):
        out = self.convnet(x)
        out = self.fc(out.view(-1,120))
        return out
```



```
Test set: Average loss: 1.1168, Accuracy: 6050/10000 (60%)

Train Epoch: 10 [0/50000 (0%)]  Loss: 1.230691
Train Epoch: 10 [1280/50000 (3%)]       Loss: 1.185565
Train Epoch: 10 [2560/50000 (5%)]       Loss: 1.065181
Train Epoch: 10 [3840/50000 (8%)]       Loss: 1.202950
Train Epoch: 10 [5120/50000 (10%)]      Loss: 1.232818
Train Epoch: 10 [6400/50000 (13%)]      Loss: 1.092711
Train Epoch: 10 [7680/50000 (15%)]      Loss: 1.266654
Train Epoch: 10 [8960/50000 (18%)]      Loss: 1.128826
Train Epoch: 10 [10240/50000 (20%)]     Loss: 1.073638
Train Epoch: 10 [11520/50000 (23%)]     Loss: 1.228029
Train Epoch: 10 [12800/50000 (26%)]     Loss: 1.373675
Train Epoch: 10 [14080/50000 (28%)]     Loss: 1.265275
Train Epoch: 10 [15360/50000 (31%)]     Loss: 1.313859
Train Epoch: 10 [16640/50000 (33%)]     Loss: 1.204013
Train Epoch: 10 [17920/50000 (36%)]     Loss: 1.105580
Train Epoch: 10 [19200/50000 (38%)]     Loss: 1.309842
Train Epoch: 10 [20480/50000 (41%)]     Loss: 1.060316
Train Epoch: 10 [21760/50000 (43%)]     Loss: 1.243446
Train Epoch: 10 [23040/50000 (46%)]     Loss: 1.026699
Train Epoch: 10 [24320/50000 (49%)]     Loss: 1.202204
Train Epoch: 10 [25600/50000 (51%)]     Loss: 1.435431
Train Epoch: 10 [26880/50000 (54%)]     Loss: 1.107203
Train Epoch: 10 [28160/50000 (56%)]     Loss: 1.408193
Train Epoch: 10 [29440/50000 (59%)]     Loss: 1.400162
Train Epoch: 10 [30720/50000 (61%)]     Loss: 1.152532
Train Epoch: 10 [32000/50000 (64%)]     Loss: 1.230844
Train Epoch: 10 [33280/50000 (66%)]     Loss: 1.096659
Train Epoch: 10 [34560/50000 (69%)]     Loss: 1.235130
Train Epoch: 10 [35840/50000 (72%)]     Loss: 1.166037
Train Epoch: 10 [37120/50000 (74%)]     Loss: 1.581140
Train Epoch: 10 [38400/50000 (77%)]     Loss: 1.168704
Train Epoch: 10 [39680/50000 (79%)]     Loss: 1.273301
Train Epoch: 10 [40960/50000 (82%)]     Loss: 0.987198
Train Epoch: 10 [42240/50000 (84%)]     Loss: 1.277181
Train Epoch: 10 [43520/50000 (87%)]     Loss: 1.119558
Train Epoch: 10 [44800/50000 (90%)]     Loss: 1.312389
Train Epoch: 10 [46080/50000 (92%)]     Loss: 1.083983
Train Epoch: 10 [47360/50000 (95%)]     Loss: 1.265219
Train Epoch: 10 [48640/50000 (97%)]     Loss: 1.243262
Train Epoch: 10 [31200/50000 (100%)]    Loss: 1.362893

Test set: Average loss: 1.1169, Accuracy: 6034/10000 (60%)

Traning and Testing total excution time is: 772.8549976348877 seconds
```

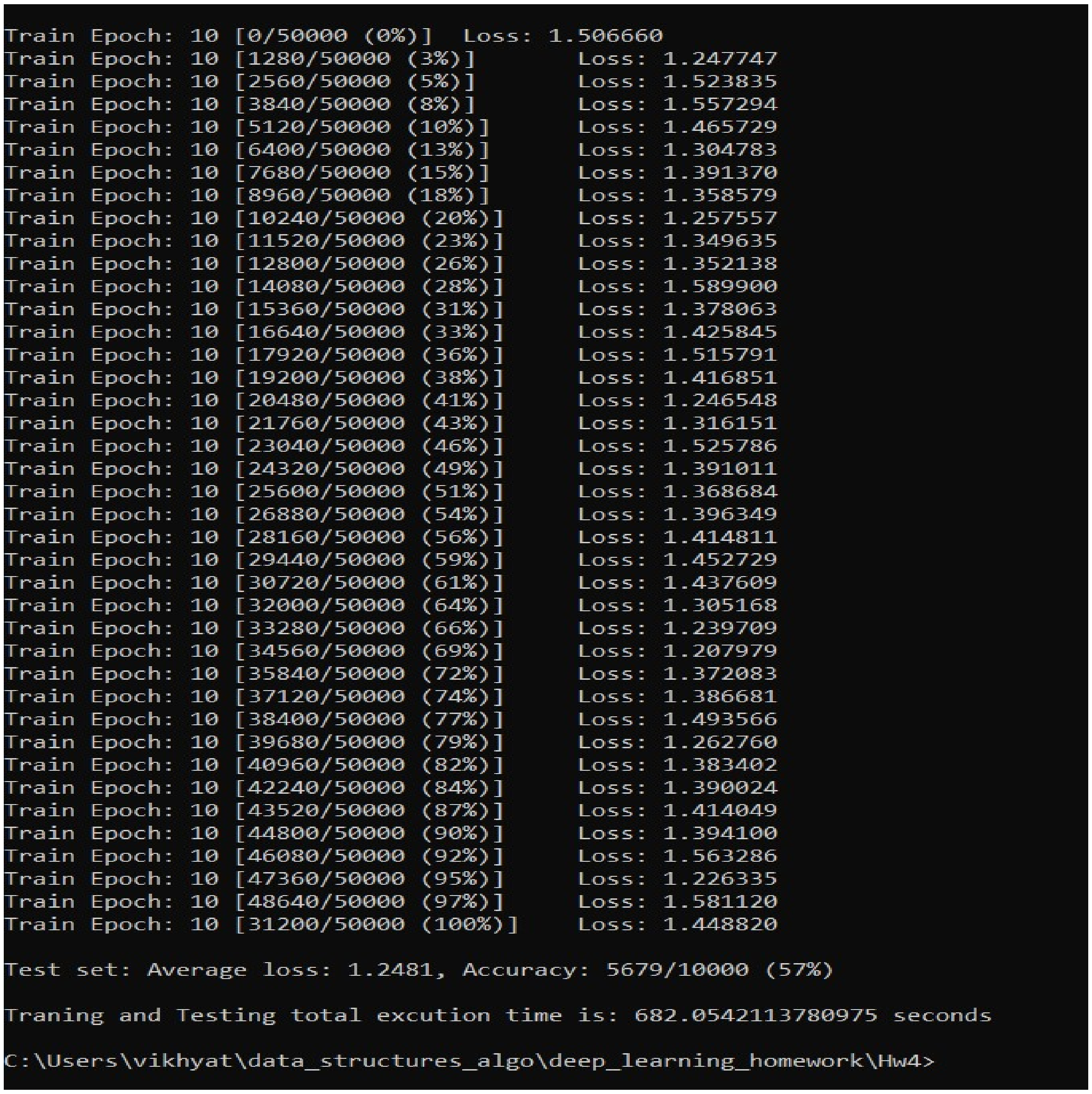
Observation:

Even with Batch Normalization Layer in the CNN as mentioned have brought more faster accuracy in testing than without BN(60 % in 10 epochs vs 62 percent in 50 epochs). So from the above two experiments it is clear that Batch Normalization is quite effective in training the Neural Network.

Part 2: The effect of Drop Out Layer on the CNN:

Case 1: I have added Drop Out layer in the FC Layer of CNN : <u>(Code lenet_cifar_fill_2_1.py)</u>

```
  class LeNet(nn.Module):

    def __init__(self):

        super(LeNet, self).__init__()
        #Sequential Model for the Convolution Neural Network
        self.convnet=nn.Sequential(nn.Conv2d(3, 6, 5),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(6, 16, 
5),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(16, 120, 5),nn.ReLU())
        #Sequential Model for the Fully Connected Layers ahead of the Convolution Part

self.fc=nn.Sequential(nn.Linear(120,84),nn.Dropout(0.3),nn.ReLU(),nn.Linear(84,10),nn.LogSoftmax(d
im=-1))


    def forward(self, x):
        out = self.convnet(x)
        out = self.fc(out.view(-1,120))
        return out
```

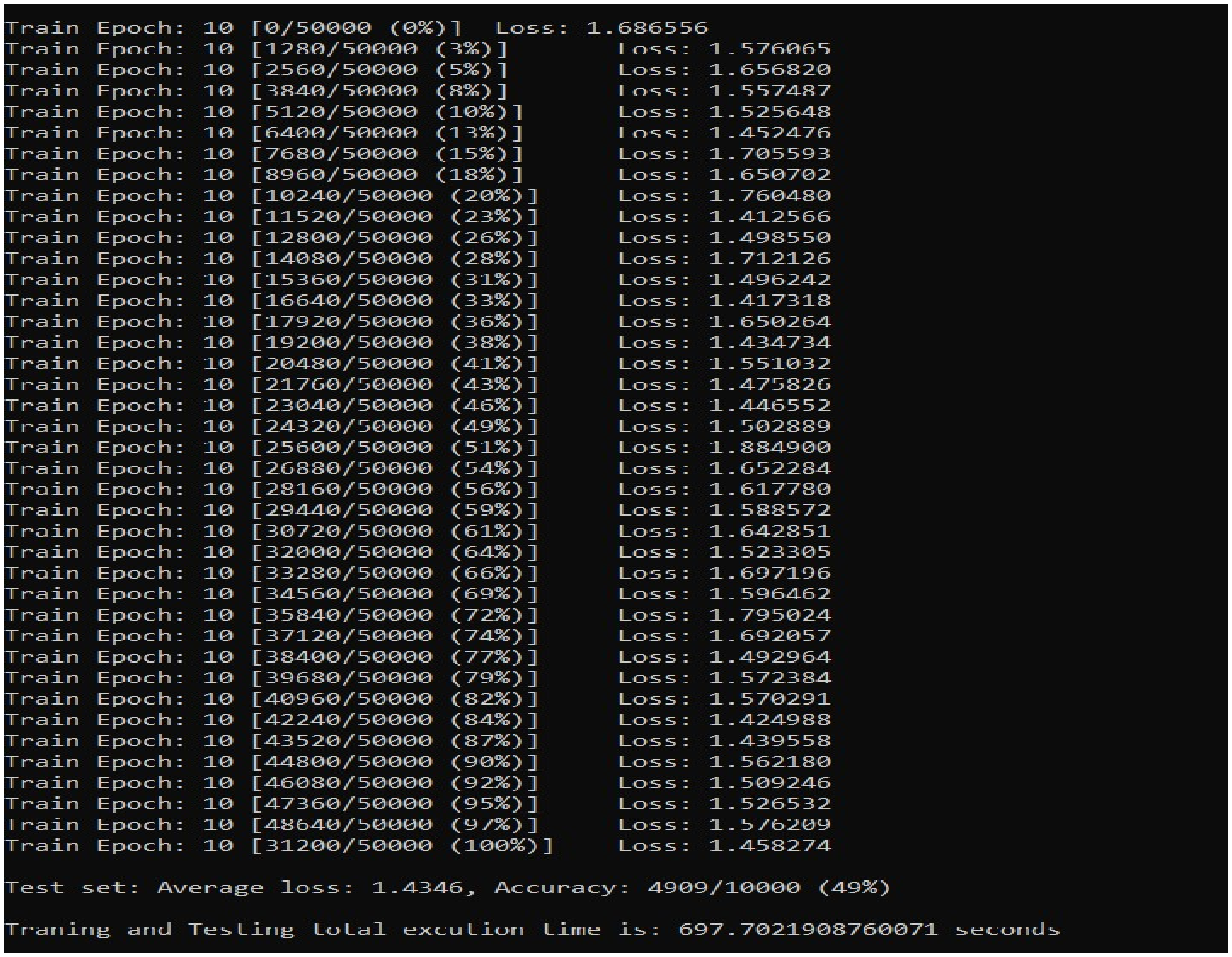


```
Train Epoch: 10 [0/50000 (0%)]  Loss: 1.506660
Train Epoch: 10 [1280/50000 (3%)]       Loss: 1.247747
Train Epoch: 10 [2560/50000 (5%)]       Loss: 1.523835
Train Epoch: 10 [3840/50000 (8%)]       Loss: 1.557294
Train Epoch: 10 [5120/50000 (10%)]      Loss: 1.465729
Train Epoch: 10 [6400/50000 (13%)]      Loss: 1.304783
Train Epoch: 10 [7680/50000 (15%)]      Loss: 1.391370
Train Epoch: 10 [8960/50000 (18%)]      Loss: 1.358579
Train Epoch: 10 [10240/50000 (20%)]     Loss: 1.257557
Train Epoch: 10 [11520/50000 (23%)]     Loss: 1.349635
Train Epoch: 10 [12800/50000 (26%)]     Loss: 1.352138
Train Epoch: 10 [14080/50000 (28%)]     Loss: 1.589900
Train Epoch: 10 [15360/50000 (31%)]     Loss: 1.378063
Train Epoch: 10 [16640/50000 (33%)]     Loss: 1.425845
Train Epoch: 10 [17920/50000 (36%)]     Loss: 1.515791
Train Epoch: 10 [19200/50000 (38%)]     Loss: 1.416851
Train Epoch: 10 [20480/50000 (41%)]     Loss: 1.246548
Train Epoch: 10 [21760/50000 (43%)]     Loss: 1.316151
Train Epoch: 10 [23040/50000 (46%)]     Loss: 1.525786
Train Epoch: 10 [24320/50000 (49%)]     Loss: 1.391011
Train Epoch: 10 [25600/50000 (51%)]     Loss: 1.368684
Train Epoch: 10 [26880/50000 (54%)]     Loss: 1.396349
Train Epoch: 10 [28160/50000 (56%)]     Loss: 1.414811
Train Epoch: 10 [29440/50000 (59%)]     Loss: 1.452729
Train Epoch: 10 [30720/50000 (61%)]     Loss: 1.437609
Train Epoch: 10 [32000/50000 (64%)]     Loss: 1.305168
Train Epoch: 10 [33280/50000 (66%)]     Loss: 1.239709
Train Epoch: 10 [34560/50000 (69%)]     Loss: 1.207979
Train Epoch: 10 [35840/50000 (72%)]     Loss: 1.372083
Train Epoch: 10 [37120/50000 (74%)]     Loss: 1.386681
Train Epoch: 10 [38400/50000 (77%)]     Loss: 1.493566
Train Epoch: 10 [39680/50000 (79%)]     Loss: 1.262760
Train Epoch: 10 [40960/50000 (82%)]     Loss: 1.383402
Train Epoch: 10 [42240/50000 (84%)]     Loss: 1.390024
Train Epoch: 10 [43520/50000 (87%)]     Loss: 1.414049
Train Epoch: 10 [44800/50000 (90%)]     Loss: 1.394100
Train Epoch: 10 [46080/50000 (92%)]     Loss: 1.563286
Train Epoch: 10 [47360/50000 (95%)]     Loss: 1.226335
Train Epoch: 10 [48640/50000 (97%)]     Loss: 1.581120
Train Epoch: 10 [31200/50000 (100%)]    Loss: 1.448820

Test set: Average loss: 1.2481, Accuracy: 5679/10000 (57%)

Traning and Testing total excution time is: 682.0542113780975 seconds

C:\Users\vikhyat\data_structures_algo\deep_learning_homework\Hw4>
```

Observations: We have achieved higher accuracy in comparison to normal CNN by including the Drop Out Layer in the FC Layer as it can be seen 57 percent accuracy in 10 epochs in 682 seconds in comparison to 55 percent in 702 seconds so in less time slightly more accuracy has been achieved. But it can be seen that Batch Normalization has offered better effect in the improvement of training of the CNN and hence accuracy.

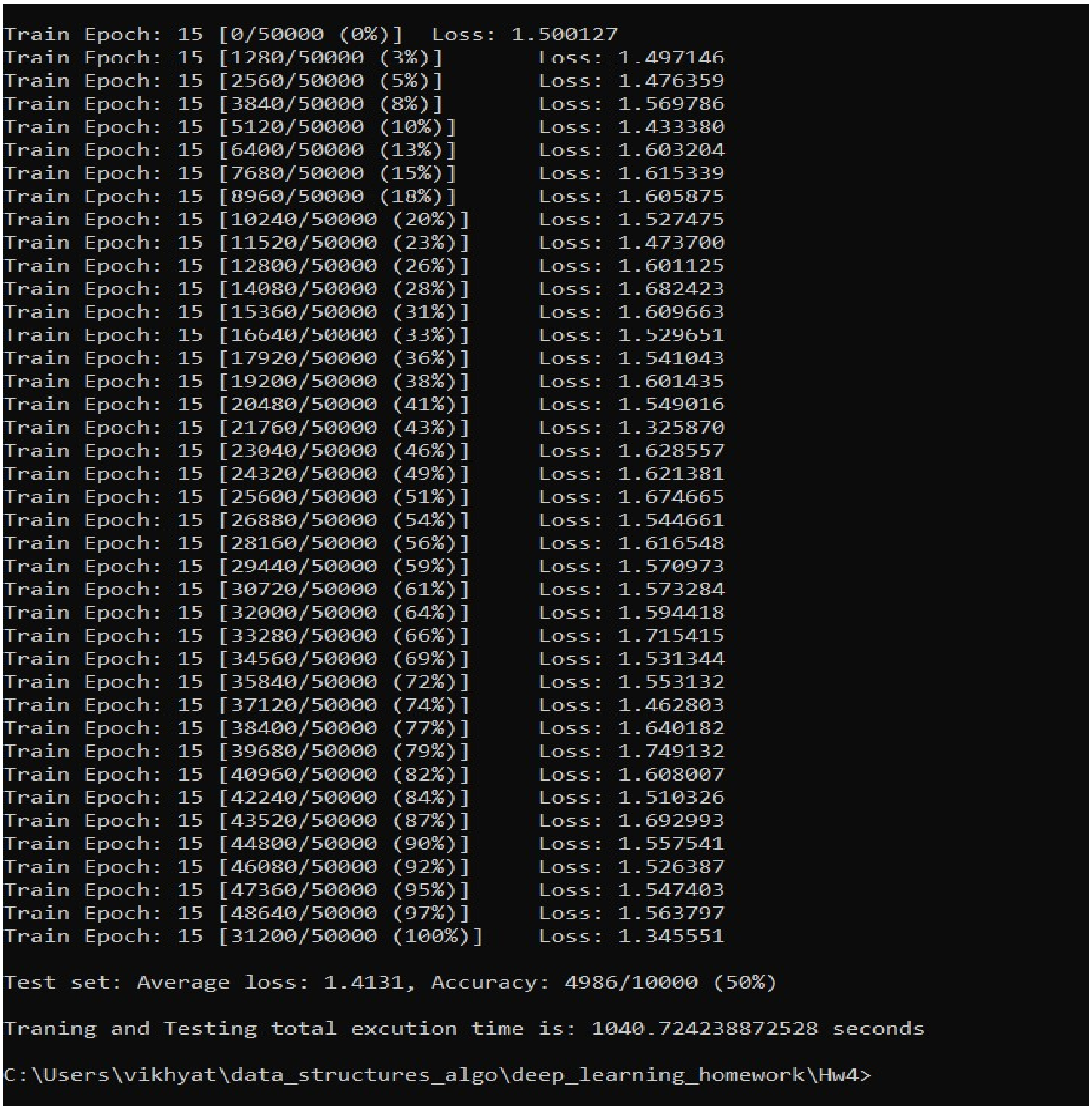
Case 2 : I have added Drop Out layer in the Convolution Layer of CNN to the above: **Code lenet_cifar_fill_2.py**

```
class LeNet(nn.Module):

    def __init__(self):

        super(LeNet, self).__init__()
        #Sequential Model for the Convolution Neural Network
        self.convnet=nn.Sequential(nn.Conv2d(3, 6,
5),nn.Dropout(0.3),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(6, 16,
5),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(16, 120, 5),nn.ReLU())
        #Sequential Model for the Fully Connected Layers ahead of the Convolution Part

self.fc=nn.Sequential(nn.Linear(120,84),nn.Dropout(0.5),nn.ReLU(),nn.Linear(84,10),nn.LogSoftmax(d
im=-1))


    def forward(self, x):
        out = self.convnet(x)
        out = self.fc(out.view(-1,120))
        return out
```



```
Train Epoch: 10 [0/50000 (0%)]  Loss: 1.686556
Train Epoch: 10 [1280/50000 (3%)]       Loss: 1.576065
Train Epoch: 10 [2560/50000 (5%)]       Loss: 1.656820
Train Epoch: 10 [3840/50000 (8%)]       Loss: 1.557487
Train Epoch: 10 [5120/50000 (10%)]      Loss: 1.525648
Train Epoch: 10 [6400/50000 (13%)]      Loss: 1.452476
Train Epoch: 10 [7680/50000 (15%)]      Loss: 1.705593
Train Epoch: 10 [8960/50000 (18%)]      Loss: 1.650702
Train Epoch: 10 [10240/50000 (20%)]     Loss: 1.760480
Train Epoch: 10 [11520/50000 (23%)]     Loss: 1.412566
Train Epoch: 10 [12800/50000 (26%)]     Loss: 1.498550
Train Epoch: 10 [14080/50000 (28%)]     Loss: 1.712126
Train Epoch: 10 [15360/50000 (31%)]     Loss: 1.496242
Train Epoch: 10 [16640/50000 (33%)]     Loss: 1.417318
Train Epoch: 10 [17920/50000 (36%)]     Loss: 1.650264
Train Epoch: 10 [19200/50000 (38%)]     Loss: 1.434734
Train Epoch: 10 [20480/50000 (41%)]     Loss: 1.551032
Train Epoch: 10 [21760/50000 (43%)]     Loss: 1.475826
Train Epoch: 10 [23040/50000 (46%)]     Loss: 1.446552
Train Epoch: 10 [24320/50000 (49%)]     Loss: 1.502889
Train Epoch: 10 [25600/50000 (51%)]     Loss: 1.884900
Train Epoch: 10 [26880/50000 (54%)]     Loss: 1.652284
Train Epoch: 10 [28160/50000 (56%)]     Loss: 1.617780
Train Epoch: 10 [29440/50000 (59%)]     Loss: 1.588572
Train Epoch: 10 [30720/50000 (61%)]     Loss: 1.642851
Train Epoch: 10 [32000/50000 (64%)]     Loss: 1.523305
Train Epoch: 10 [33280/50000 (66%)]     Loss: 1.697196
Train Epoch: 10 [34560/50000 (69%)]     Loss: 1.596462
Train Epoch: 10 [35840/50000 (72%)]     Loss: 1.795024
Train Epoch: 10 [37120/50000 (74%)]     Loss: 1.692057
Train Epoch: 10 [38400/50000 (77%)]     Loss: 1.492964
Train Epoch: 10 [39680/50000 (79%)]     Loss: 1.572384
Train Epoch: 10 [40960/50000 (82%)]     Loss: 1.570291
Train Epoch: 10 [42240/50000 (84%)]     Loss: 1.424988
Train Epoch: 10 [43520/50000 (87%)]     Loss: 1.439558
Train Epoch: 10 [44800/50000 (90%)]     Loss: 1.562180
Train Epoch: 10 [46080/50000 (92%)]     Loss: 1.509246
Train Epoch: 10 [47360/50000 (95%)]     Loss: 1.526532
Train Epoch: 10 [48640/50000 (97%)]     Loss: 1.576209
Train Epoch: 10 [31200/50000 (100%)]    Loss: 1.458274

Test set: Average loss: 1.4346, Accuracy: 4909/10000 (49%)

Traning and Testing total excution time is: 697.7021908760071 seconds
```

Observations : As it can be seen that the training accuracy has gone done even in comparison with the normal CNN when drop out has been applied on the Convolution Layer . Here I have compared in case of 10 epochs how the CNN with one drop out layer in Convolution layer has been performing.



```
Train Epoch: 15 [0/50000 (0%)]  Loss: 1.500127
Train Epoch: 15 [1280/50000 (3%)]       Loss: 1.497146
Train Epoch: 15 [2560/50000 (5%)]       Loss: 1.476359
Train Epoch: 15 [3840/50000 (8%)]       Loss: 1.569786
Train Epoch: 15 [5120/50000 (10%)]      Loss: 1.433380
Train Epoch: 15 [6400/50000 (13%)]      Loss: 1.603204
Train Epoch: 15 [7680/50000 (15%)]      Loss: 1.615339
Train Epoch: 15 [8960/50000 (18%)]      Loss: 1.605875
Train Epoch: 15 [10240/50000 (20%)]     Loss: 1.527475
Train Epoch: 15 [11520/50000 (23%)]     Loss: 1.473700
Train Epoch: 15 [12800/50000 (26%)]     Loss: 1.601125
Train Epoch: 15 [14080/50000 (28%)]     Loss: 1.682423
Train Epoch: 15 [15360/50000 (31%)]     Loss: 1.609663
Train Epoch: 15 [16640/50000 (33%)]     Loss: 1.529651
Train Epoch: 15 [17920/50000 (36%)]     Loss: 1.541043
Train Epoch: 15 [19200/50000 (38%)]     Loss: 1.601435
Train Epoch: 15 [20480/50000 (41%)]     Loss: 1.549016
Train Epoch: 15 [21760/50000 (43%)]     Loss: 1.325870
Train Epoch: 15 [23040/50000 (46%)]     Loss: 1.628557
Train Epoch: 15 [24320/50000 (49%)]     Loss: 1.621381
Train Epoch: 15 [25600/50000 (51%)]     Loss: 1.674665
Train Epoch: 15 [26880/50000 (54%)]     Loss: 1.544661
Train Epoch: 15 [28160/50000 (56%)]     Loss: 1.616548
Train Epoch: 15 [29440/50000 (59%)]     Loss: 1.570973
Train Epoch: 15 [30720/50000 (61%)]     Loss: 1.573284
Train Epoch: 15 [32000/50000 (64%)]     Loss: 1.594418
Train Epoch: 15 [33280/50000 (66%)]     Loss: 1.715415
Train Epoch: 15 [34560/50000 (69%)]     Loss: 1.531344
Train Epoch: 15 [35840/50000 (72%)]     Loss: 1.553132
Train Epoch: 15 [37120/50000 (74%)]     Loss: 1.462803
Train Epoch: 15 [38400/50000 (77%)]     Loss: 1.640182
Train Epoch: 15 [39680/50000 (79%)]     Loss: 1.749132
Train Epoch: 15 [40960/50000 (82%)]     Loss: 1.608007
Train Epoch: 15 [42240/50000 (84%)]     Loss: 1.510326
Train Epoch: 15 [43520/50000 (87%)]     Loss: 1.692993
Train Epoch: 15 [44800/50000 (90%)]     Loss: 1.557541
Train Epoch: 15 [46080/50000 (92%)]     Loss: 1.526387
Train Epoch: 15 [47360/50000 (95%)]     Loss: 1.547403
Train Epoch: 15 [48640/50000 (97%)]     Loss: 1.563797
Train Epoch: 15 [31200/50000 (100%)]    Loss: 1.345551

Test set: Average loss: 1.4131, Accuracy: 4986/10000 (50%)

Traning and Testing total excution time is: 1040.724238872528 seconds

C:\Users\vikhyat\data_structures_algo\deep_learning_homework\Hw4>
```

Observations : As it can be seen that the training accuracy has reached just 50 percent in 15 epochs case of Drop out in Convolution Layer so performance is less in comparison to normal.

Possible Reasons for above results are :-

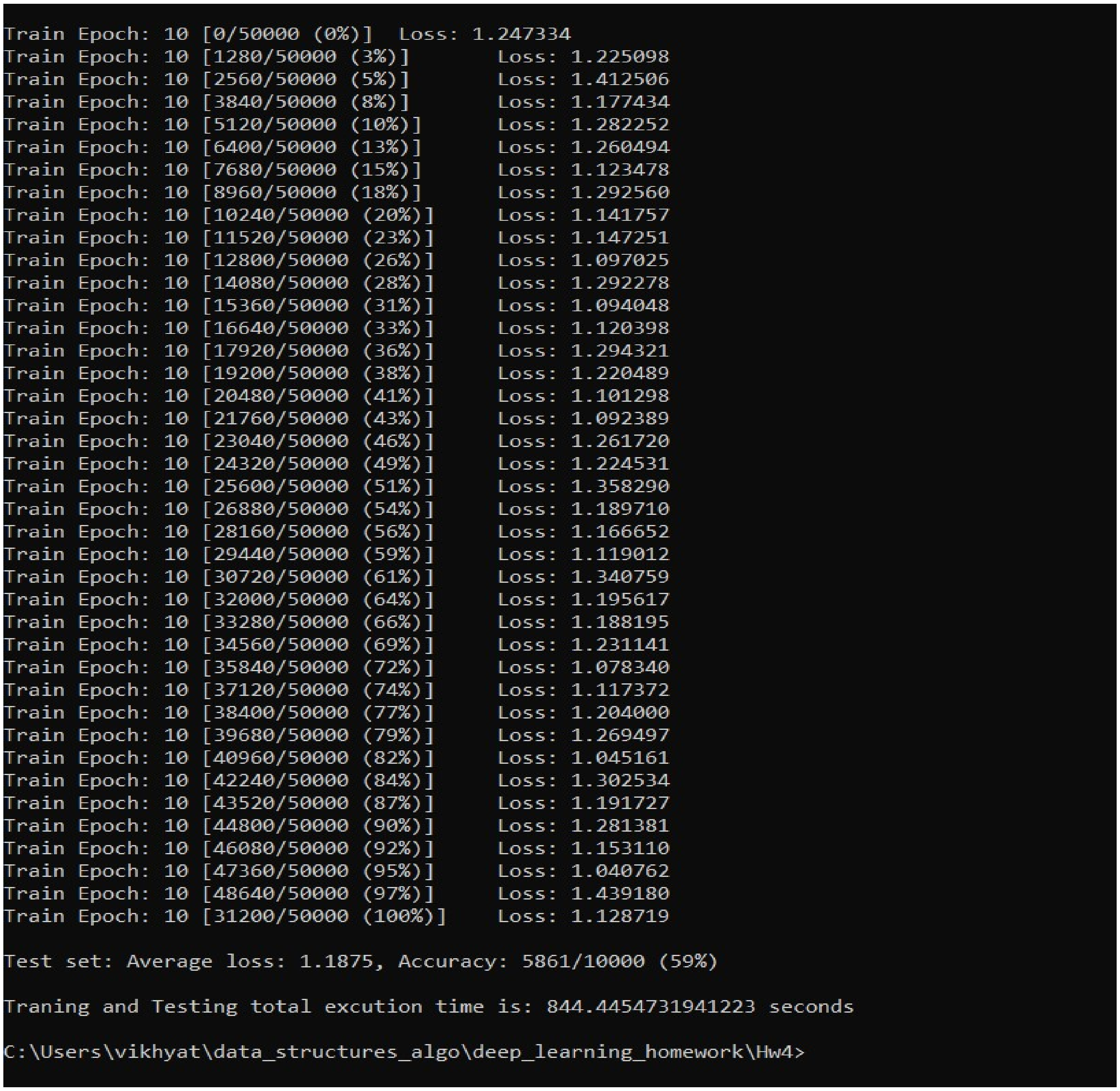
Dropout is falling out of favor in recent applications, there are two main reasons.

1. Convolutional layers have few parameters, so they need less regularization to begin with. Furthermore, because of the spatial relationships encoded in feature maps, activations can become highly correlated. This renders dropout ineffective. So mostly they are preferable in fully connected layers.
2. But Large models included fully connected layers at the end of the network. For such models, overfitting was combatted by including dropout between fully connected layers. Unfortunately, recent architectures moving away from this fully connected block.

The high performance of the batch-normalized model supports the claim that batch normalization should be used between convolutions. Furthermore , dropout may not be placed between convolutions, as models with dropout tended as shown may perform worse than the normal model.

Part 4: The effect of one Drop Out Layer and one Batch Normalization Layer on the CNN: Code lenet_cifar_fill_4.py)

```
class LeNet(nn.Module):

    def __init__(self):

        super(LeNet, self).__init__()
        #Sequential Model for the Convolution Neural Network
        self.convnet=nn.Sequential(nn.Conv2d(3, 6, 5),nn.ReLU(),nn.MaxPool2d(2,2),nn.Con
v2d(6, 16, 5),nn.ReLU(),nn.MaxPool2d(2,2),nn.Conv2d(16, 120, 5),nn.ReLU())
        #Sequential Model for the Fully Connected Layers ahead of the Convolution Part
        self.fc=nn.Sequential(nn.Linear(120,84),nn.BatchNorm1d(84),nn.Dropout(0.3),nn.Re
LU(),nn.Linear(84,10),nn.LogSoftmax(dim=-1))


    def forward(self, x):
        out = self.convnet(x)
        out = self.fc(out.view(-1,120))
        return out
```



```
Train Epoch: 10 [0/50000 (0%)]  Loss: 1.247334
Train Epoch: 10 [1280/50000 (3%)]       Loss: 1.225098
Train Epoch: 10 [2560/50000 (5%)]       Loss: 1.412506
Train Epoch: 10 [3840/50000 (8%)]       Loss: 1.177434
Train Epoch: 10 [5120/50000 (10%)]      Loss: 1.282252
Train Epoch: 10 [6400/50000 (13%)]      Loss: 1.260494
Train Epoch: 10 [7680/50000 (15%)]      Loss: 1.123478
Train Epoch: 10 [8960/50000 (18%)]      Loss: 1.292560
Train Epoch: 10 [10240/50000 (20%)]     Loss: 1.141757
Train Epoch: 10 [11520/50000 (23%)]     Loss: 1.147251
Train Epoch: 10 [12800/50000 (26%)]     Loss: 1.097025
Train Epoch: 10 [14080/50000 (28%)]     Loss: 1.292278
Train Epoch: 10 [15360/50000 (31%)]     Loss: 1.094048
Train Epoch: 10 [16640/50000 (33%)]     Loss: 1.120398
Train Epoch: 10 [17920/50000 (36%)]     Loss: 1.294321
Train Epoch: 10 [19200/50000 (38%)]     Loss: 1.220489
Train Epoch: 10 [20480/50000 (41%)]     Loss: 1.101298
Train Epoch: 10 [21760/50000 (43%)]     Loss: 1.092389
Train Epoch: 10 [23040/50000 (46%)]     Loss: 1.261720
Train Epoch: 10 [24320/50000 (49%)]     Loss: 1.224531
Train Epoch: 10 [25600/50000 (51%)]     Loss: 1.358290
Train Epoch: 10 [26880/50000 (54%)]     Loss: 1.189710
Train Epoch: 10 [28160/50000 (56%)]     Loss: 1.166652
Train Epoch: 10 [29440/50000 (59%)]     Loss: 1.119012
Train Epoch: 10 [30720/50000 (61%)]     Loss: 1.340759
Train Epoch: 10 [32000/50000 (64%)]     Loss: 1.195617
Train Epoch: 10 [33280/50000 (66%)]     Loss: 1.188195
Train Epoch: 10 [34560/50000 (69%)]     Loss: 1.231141
Train Epoch: 10 [35840/50000 (72%)]     Loss: 1.078340
Train Epoch: 10 [37120/50000 (74%)]     Loss: 1.117372
Train Epoch: 10 [38400/50000 (77%)]     Loss: 1.204000
Train Epoch: 10 [39680/50000 (79%)]     Loss: 1.269497
Train Epoch: 10 [40960/50000 (82%)]     Loss: 1.045161
Train Epoch: 10 [42240/50000 (84%)]     Loss: 1.302534
Train Epoch: 10 [43520/50000 (87%)]     Loss: 1.191727
Train Epoch: 10 [44800/50000 (90%)]     Loss: 1.281381
Train Epoch: 10 [46080/50000 (92%)]     Loss: 1.153110
Train Epoch: 10 [47360/50000 (95%)]     Loss: 1.040762
Train Epoch: 10 [48640/50000 (97%)]     Loss: 1.439180
Train Epoch: 10 [31200/50000 (100%)]    Loss: 1.128719

Test set: Average loss: 1.1875, Accuracy: 5861/10000 (59%)

Traning and Testing total excution time is: 844.4454731941223 seconds

C:\Users\vikhyat\data_structures_algo\deep_learning_homework\Hw4>
```

Observation : The time taken has increased a bit in comparison to others networks we discussed for 10 epochs but as you can see the performance is better than normal plain vanilla network and also from network with drop out layer only as here the accuracy of around 59 percent we are able to achieve in 10 epochs in slightly more time.