Xi'an Jiaotong-Liverpool University

西交利物浦大學

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE408(18/19) Deep learning in computer vision

Lab 1

Lab Report

Student Name	:	Enge Xu
Student ID	:	1821635
Date	:	2019/4/22
Professor	:	Jimin Xiao

ABSTRACT

This assessment aims at evaluating ability to exploit the deep learning knowledge, which is accumulated during lectures, and after-class study, to analyze, design, implement, develop, test and document image classification methods using deep learning. The assessment will be based on the Pytorch software。

BACKGROUND

1. Deep learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

2. CNN

In machine learning, a convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field.

3. Pytorch

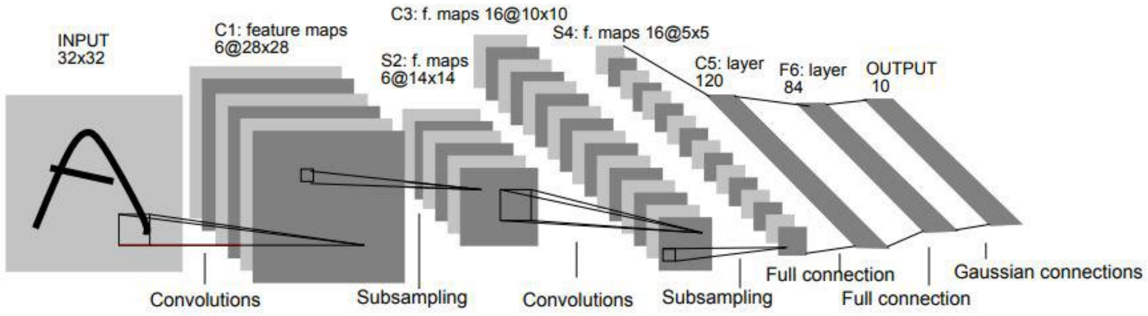
PyTorch is an open-source machine learning library for Python, based on Torch, used for applications such as natural language processing. It is primarily developed by Facebook's artificial-intelligence research group, and Uber's "Pyro" Probabilistic programming language software is built on it.

4. CIFAR-10

The CIFAR-10 dataset (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research.

5. LeNet-5

Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner proposed a neural network architecture for handwritten and machine-printed character recognition in 1990's which they called LeNet-5. The architecture is straightforward and simple to understand that's why it is mostly used as a first step for teaching Convolutional Neural Network.



The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier.

TASK INTRODUCTION

1. Please use the " trainval_net_cifar10.py " to train the image classification convolutional neural network (CNN). The network structure is specified in the file. Please draw an illustrative CNN network configuration figure for it, and explain the feature size in each network layer. Please also write down the number of parameters and number of neurons in each layer. (20%)

2. Plot a figure to show how training loss and testing loss change with training iteration number. Compare the training loss with the testing loss, and explain why this happens. (20%)

3. CNN network structure will affect the image classification performance. Please change the network structure by modifying the corresponding code in " trainval_net_cifar10.py " and provide the test accuracy with each new network structure. The modified code for all the network models should be included in the report. (20%)

(The modifications should include 4 items: (a) half the channel number in each layer; (b) double the channel number in each layer; (c) change to CNN layer number from 3 to 0 and 1;　(d) change the CNN layer number from 3 to 4 and 5.)

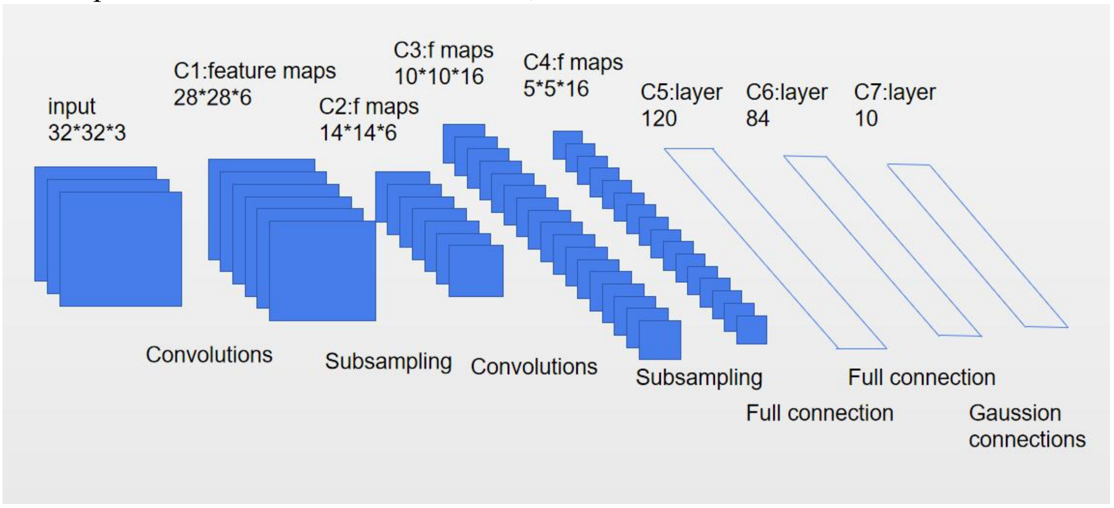
4. Data augment technique is commonly used for the deep learning training process. Please propose a data augment method, and train the CNN network specified in “trainval_net_cifar10.py”　with the augmented dataset.　Please write down the adopted data augment method and record the final test accuracy.　(20%)

6. Please propose a method to boost the image classification performance, and detail your method and report the final image classification performance that you achieved. (20%)

METHODOLOGY

TASK 1

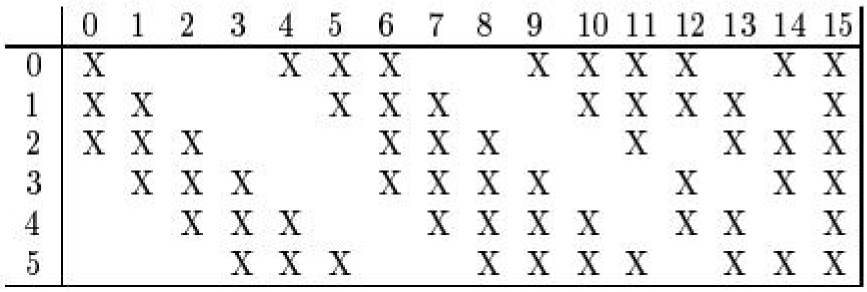
The required structure is shown as below;



First Layer:
The input is a 32×32×3 grayscale image which passes through the first convolutional layer with 6 feature maps or filters having size 5×5 and a stride of one. The image dimensions changes from 32x32x3 to 28x28x6.
Parameter:　（（5x5x3）+1）x6 = 456;
Neuron: 28x28x6 = 4704.

Second Layer:
Then it applies average pooling layer or sub-sampling layer with a filter size 2×2 and a stride of two. The resulting image dimensions will be reduced to 14x14x6.
Parameter: 6x(1+1) = 12;
Neuron: 14x14x6 = 1176.

Third Layer:
Next, there is a second convolutional layer with 16 feature maps having size 5×5 and a stride of 1. In this layer, only 10 out of 16 feature maps are connected to 6 feature maps of the previous layer as shown below.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

The main reason is to break the symmetry in the network and keeps the number of connections within reasonable bounds. That' s why the number of training parameters in this layers are 1516 instead of 2400 and similarly, the number of connections are 151600 instead of 240000.
Parameter: (5x5x3+1)x6+(5x5x4+1)x9+(5x5x6+1)=1516 ;
Neuron: 10x10x16 = 1600.

Fourth Layer:
The fourth layer (S4) is again an average pooling layer with filter size 2×2 and a stride of 2. This layer is the same as the second layer (S2) except it has 16 feature maps so the output will be reduced to 5x5x16.
Parameter: 16x2 = 32 ;
Neuron: 5x5x16 = 400.

Fifth Layer:
The fifth layer (C5) is a fully connected convolutional layer with 120 feature maps each of size 1×1. Each of the 120 units in C5 is connected to all the 400 nodes (5x5x16) in the fourth layer S4.
Parameter: 120x(5x5x16+1) = 48120;
Neuron: 1x1x120 = 120.

Sixth Layer:
The sixth layer is a fully connected layer (F6) with 84 units.
Parameter: (120+1)x84 = 10164;
Neuron: 1x1x84 = 84.

Output Layer:
Finally, there is a fully connected softmax output layer ŷ with 10 possible values

corresponding to the digits from 0 to 9.

TASK 2

Train loss keeps decreasing, test accuracy keeps rising, indicating that the network is still learning.

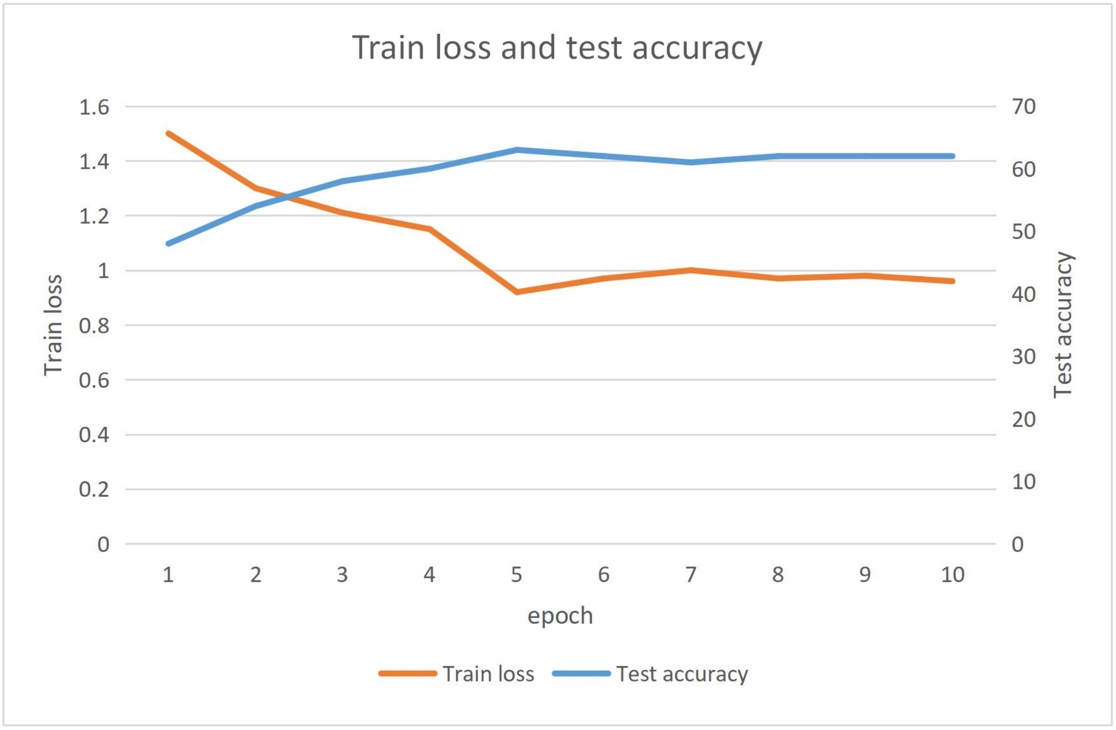
Train loss decreases continuously while test accuracy tends to remain unchanged, indicating that the network is overfitting.

Train loss tends to remain unchanged, while test accuracy keeps rising, indicating that the data set is 100% defective.

Train loss tends to remain unchanged, while test accuracy tends to remain unchanged, indicating that learning rate or batch number should be reduced when learning meets a bottleneck.

Train loss keeps rising, and test accuracy keeps decreasing, which indicates problems such as improper network structure design, improper training parameter setting and data set cleaning.

The figure of train loss and test accuracy is shown as below;

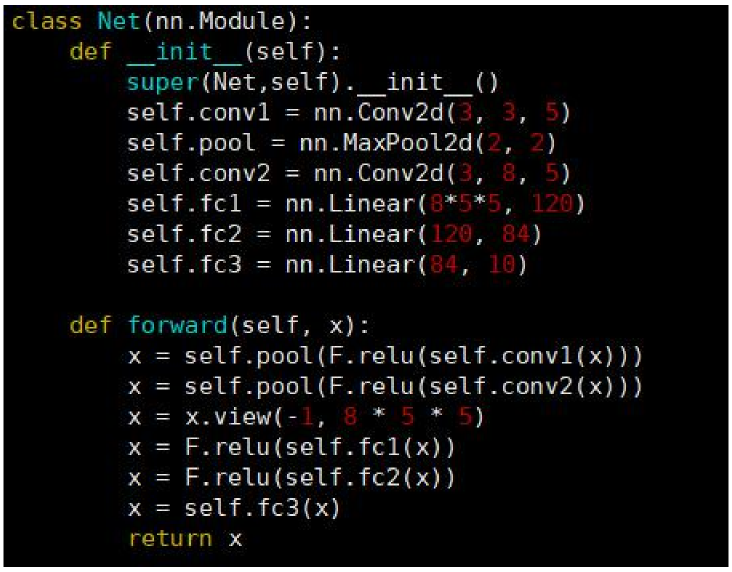


As in the figure, in the first 5 epochs, train loss keeps declaring and test accuracy keeps rising. It means the network is still learning. In the second 5 epochs, train loss and test accuracy tend to remain unchanged. It means learning rate or batch number should be reduced when learning meets a bottleneck. Among the whole process, train loss is inverse ratio with test accuracy.

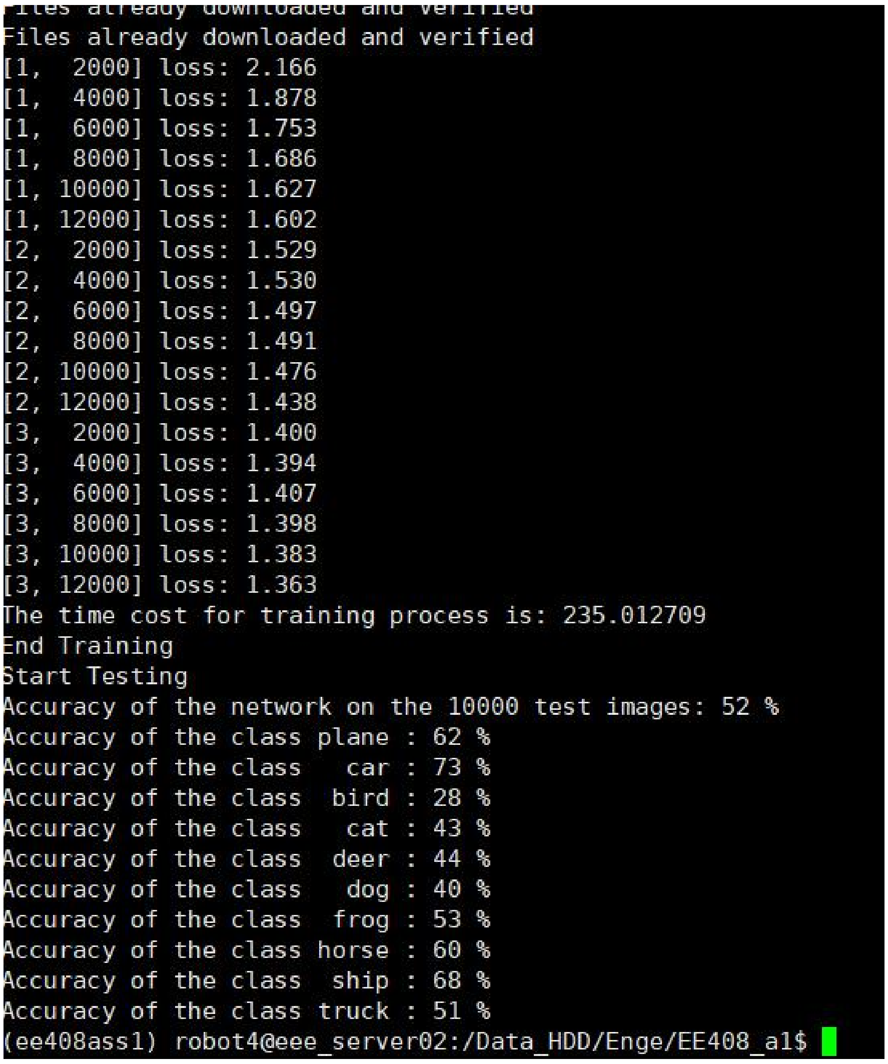
TASK 3

Without considering the performance of the network, the core code of different layers is shown as below, adding with train loss and test accuracy;

(a) Half the channel number in each layer;

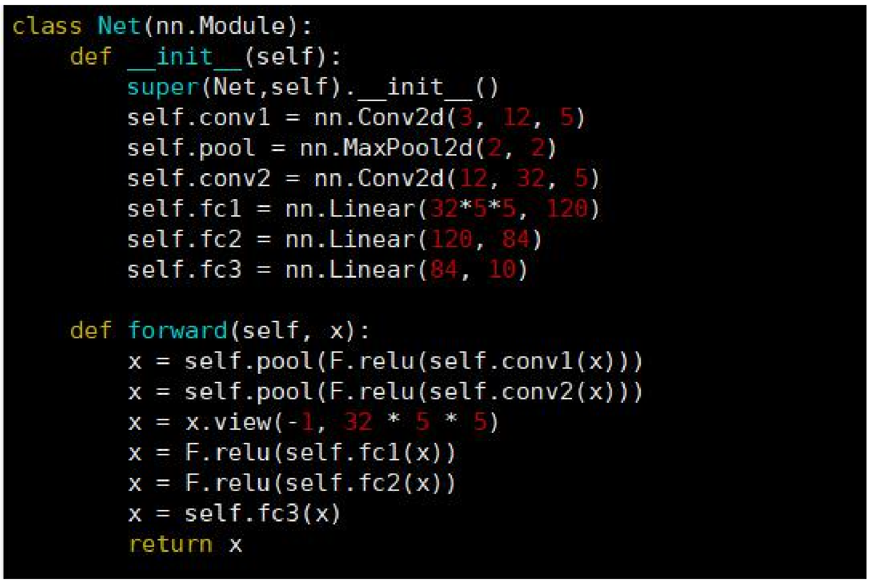


```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(3, 3, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(3, 8, 5)
        self.fc1 = nn.Linear(8*5*5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = 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, 8 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

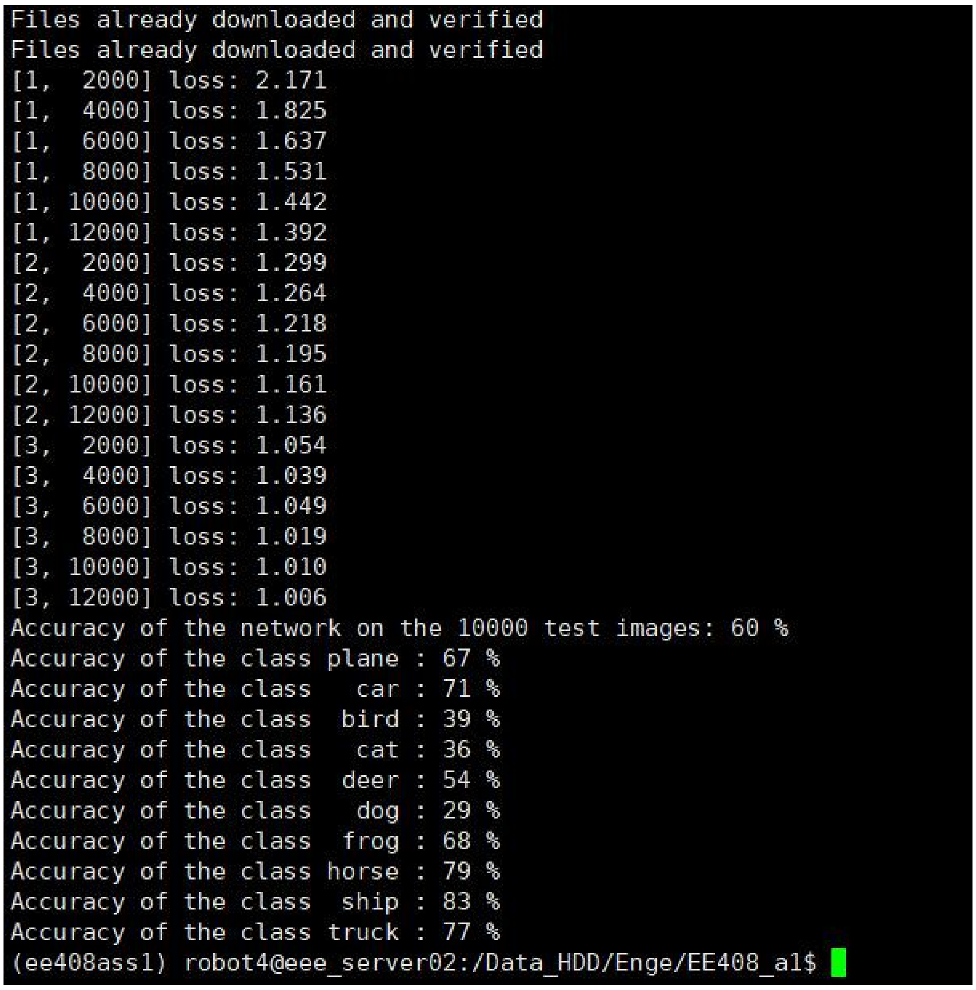


```
Files already downloaded and verified
[1,  2000] loss: 2.166
[1,  4000] loss: 1.878
[1,  6000] loss: 1.753
[1,  8000] loss: 1.686
[1, 10000] loss: 1.627
[1, 12000] loss: 1.602
[2,  2000] loss: 1.529
[2,  4000] loss: 1.530
[2,  6000] loss: 1.497
[2,  8000] loss: 1.491
[2, 10000] loss: 1.476
[2, 12000] loss: 1.438
[3,  2000] loss: 1.400
[3,  4000] loss: 1.394
[3,  6000] loss: 1.407
[3,  8000] loss: 1.398
[3, 10000] loss: 1.383
[3, 12000] loss: 1.363
The time cost for training process is: 235.012709
End Training
Start Testing
Accuracy of the network on the 10000 test images: 52 %
Accuracy of the class plane : 62 %
Accuracy of the class   car : 73 %
Accuracy of the class  bird : 28 %
Accuracy of the class   cat : 43 %
Accuracy of the class  deer : 44 %
Accuracy of the class   dog : 40 %
Accuracy of the class  frog : 53 %
Accuracy of the class horse : 60 %
Accuracy of the class  ship : 68 %
Accuracy of the class truck : 51 %
(ee408ass1) robot4@eee_server02:/Data_HDD/Enge/EE408_a1$
```

(b) Double the channel number in each layer;



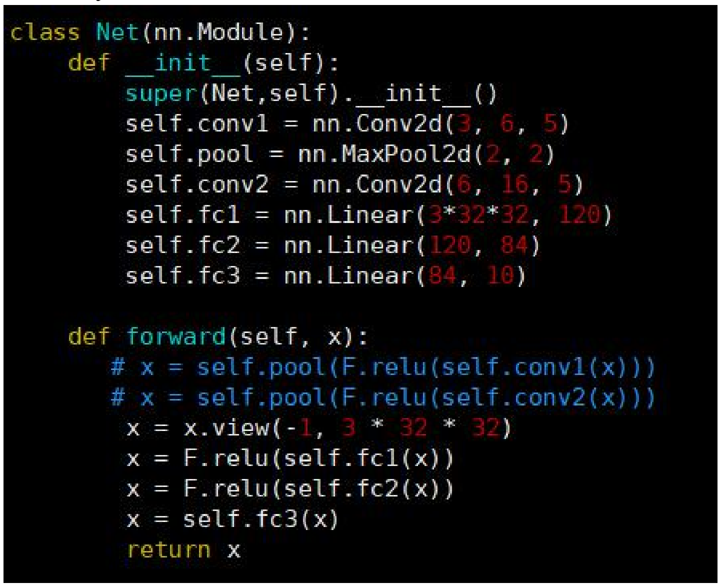
```python
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(3, 12, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(12, 32, 5)
        self.fc1 = nn.Linear(32*5*5, 120)
        self.fc2 = nn.Linear(120, 84)
        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, 32 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```



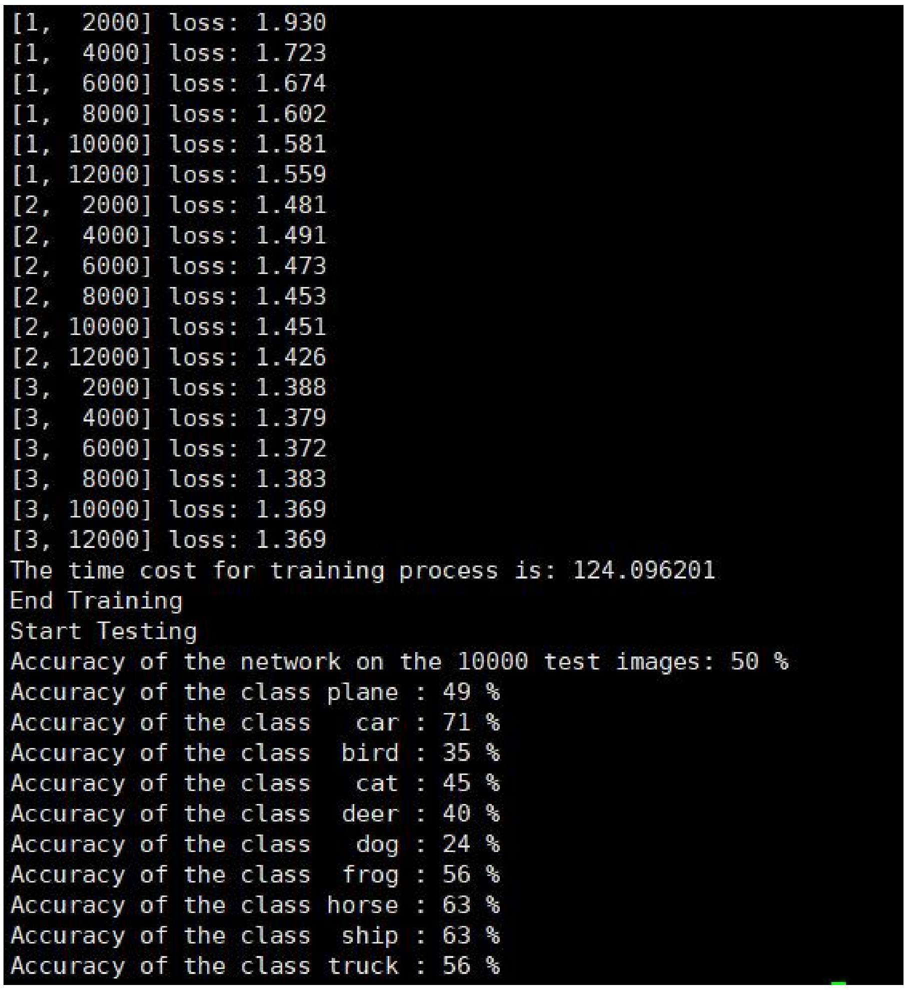
```
Files already downloaded and verified
Files already downloaded and verified
[1,  2000] loss: 2.171
[1,  4000] loss: 1.825
[1,  6000] loss: 1.637
[1,  8000] loss: 1.531
[1, 10000] loss: 1.442
[1, 12000] loss: 1.392
[2,  2000] loss: 1.299
[2,  4000] loss: 1.264
[2,  6000] loss: 1.218
[2,  8000] loss: 1.195
[2, 10000] loss: 1.161
[2, 12000] loss: 1.136
[3,  2000] loss: 1.054
[3,  4000] loss: 1.039
[3,  6000] loss: 1.049
[3,  8000] loss: 1.019
[3, 10000] loss: 1.010
[3, 12000] loss: 1.006
Accuracy of the network on the 10000 test images: 60 %
Accuracy of the class plane : 67 %
Accuracy of the class   car : 71 %
Accuracy of the class  bird : 39 %
Accuracy of the class   cat : 36 %
Accuracy of the class  deer : 54 %
Accuracy of the class   dog : 29 %
Accuracy of the class  frog : 68 %
Accuracy of the class horse : 79 %
Accuracy of the class  ship : 83 %
Accuracy of the class truck : 77 %
(ee408ass1) robot4@eee_server02:/Data_HDD/Enge/EE408_a1$ 
```

(c) Change to CNN layer number from 3 to 0 and 1;

Zero layer;

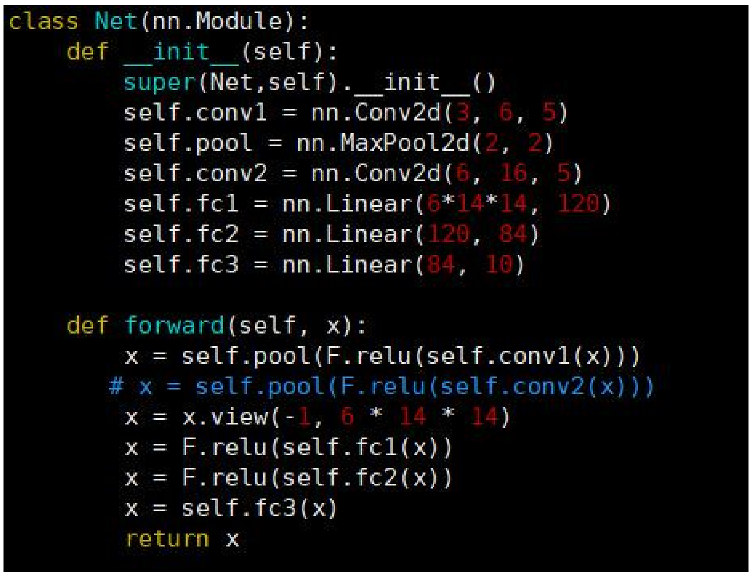


```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(3*32*32, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = 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, 3 * 32 * 32)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

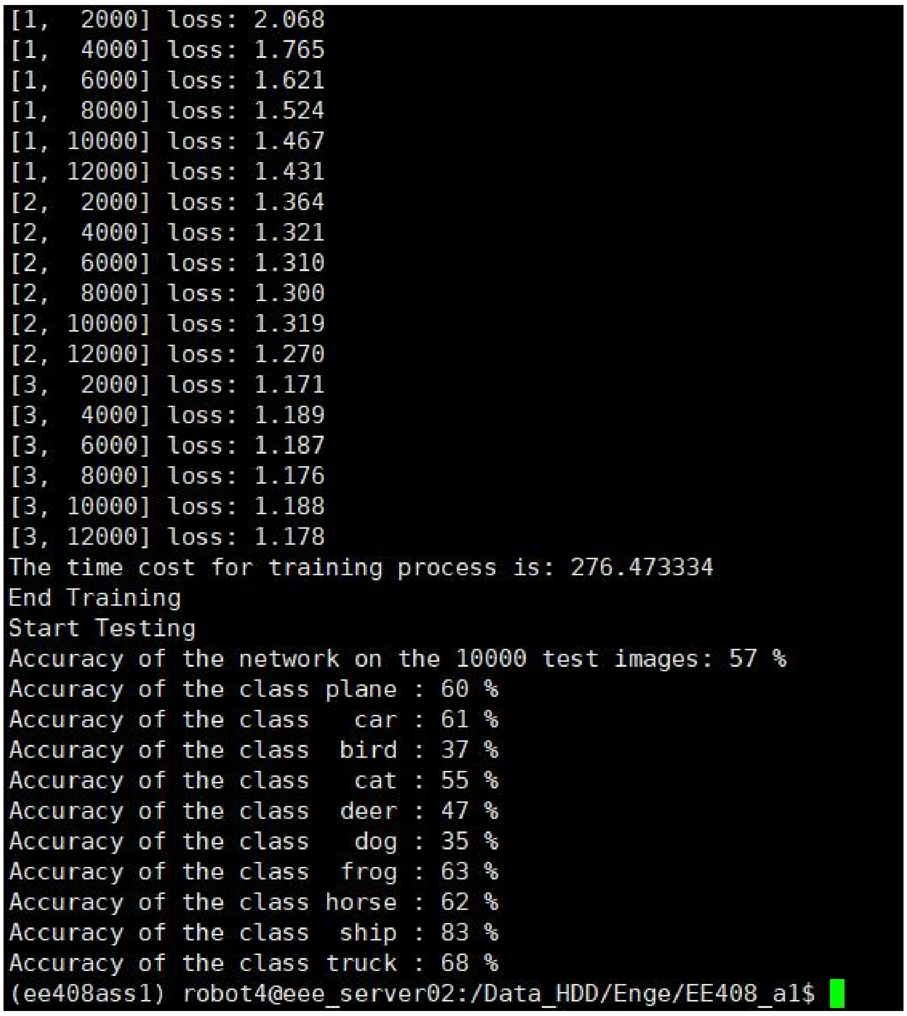


```
[1,  2000] loss: 1.930
[1,  4000] loss: 1.723
[1,  6000] loss: 1.674
[1,  8000] loss: 1.602
[1, 10000] loss: 1.581
[1, 12000] loss: 1.559
[2,  2000] loss: 1.481
[2,  4000] loss: 1.491
[2,  6000] loss: 1.473
[2,  8000] loss: 1.453
[2, 10000] loss: 1.451
[2, 12000] loss: 1.426
[3,  2000] loss: 1.388
[3,  4000] loss: 1.379
[3,  6000] loss: 1.372
[3,  8000] loss: 1.383
[3, 10000] loss: 1.369
[3, 12000] loss: 1.369
The time cost for training process is: 124.096201
End Training
Start Testing
Accuracy of the network on the 10000 test images: 50 %
Accuracy of the class plane : 49 %
Accuracy of the class   car : 71 %
Accuracy of the class  bird : 35 %
Accuracy of the class   cat : 45 %
Accuracy of the class  deer : 40 %
Accuracy of the class   dog : 24 %
Accuracy of the class  frog : 56 %
Accuracy of the class horse : 63 %
Accuracy of the class  ship : 63 %
Accuracy of the class truck : 56 %
```

One layer;



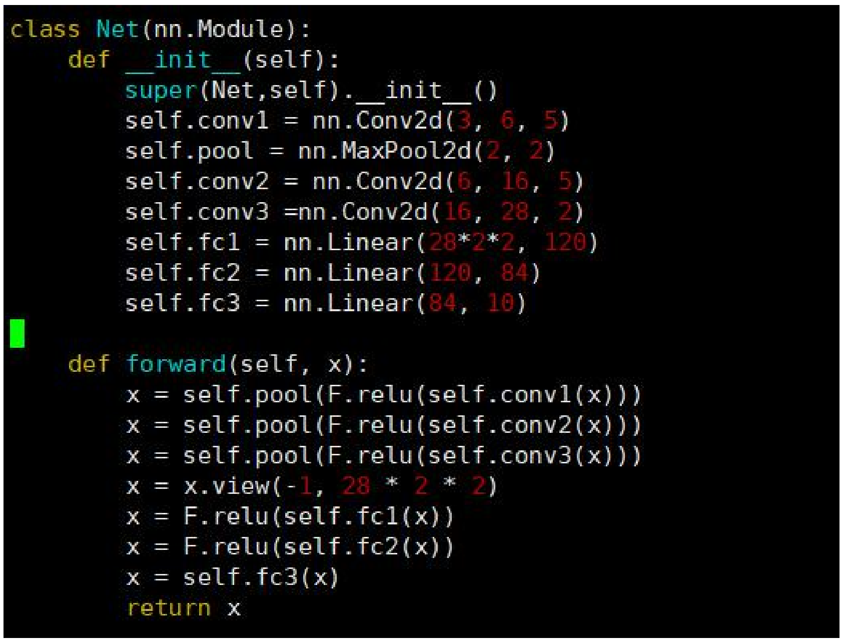
```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(6*14*14, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = 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, 6 * 14 * 14)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```



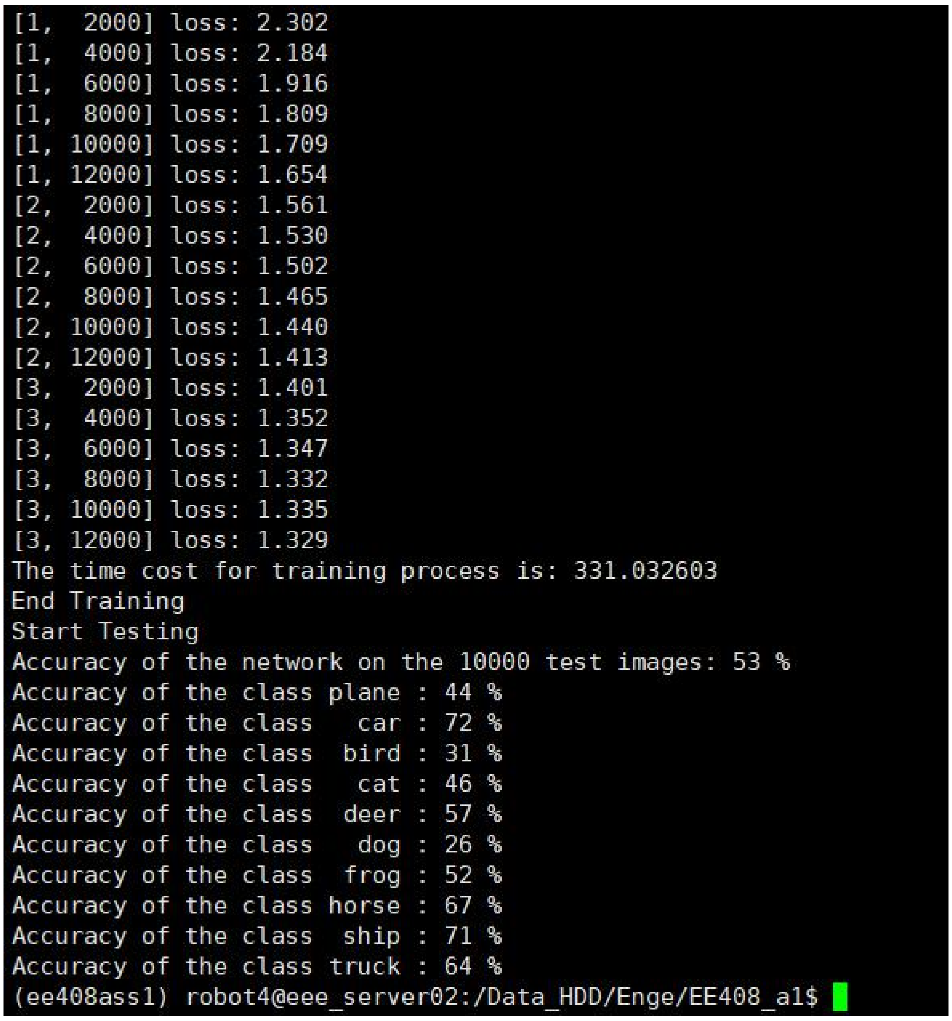
```
[1,  2000] loss: 2.068
[1,  4000] loss: 1.765
[1,  6000] loss: 1.621
[1,  8000] loss: 1.524
[1, 10000] loss: 1.467
[1, 12000] loss: 1.431
[2,  2000] loss: 1.364
[2,  4000] loss: 1.321
[2,  6000] loss: 1.310
[2,  8000] loss: 1.300
[2, 10000] loss: 1.319
[2, 12000] loss: 1.270
[3,  2000] loss: 1.171
[3,  4000] loss: 1.189
[3,  6000] loss: 1.187
[3,  8000] loss: 1.176
[3, 10000] loss: 1.188
[3, 12000] loss: 1.178
The time cost for training process is: 276.473334
End Training
Start Testing
Accuracy of the network on the 10000 test images: 57 %
Accuracy of the class plane : 60 %
Accuracy of the class   car : 61 %
Accuracy of the class  bird : 37 %
Accuracy of the class   cat : 55 %
Accuracy of the class  deer : 47 %
Accuracy of the class   dog : 35 %
Accuracy of the class  frog : 63 %
Accuracy of the class horse : 62 %
Accuracy of the class  ship : 83 %
Accuracy of the class truck : 68 %
(ee408ass1) robot4@eee_server02:/Data_HDD/Enge/EE408_a1$
```

(d) Change the CNN layer number from 3 to 4 and 5;

Three layer;

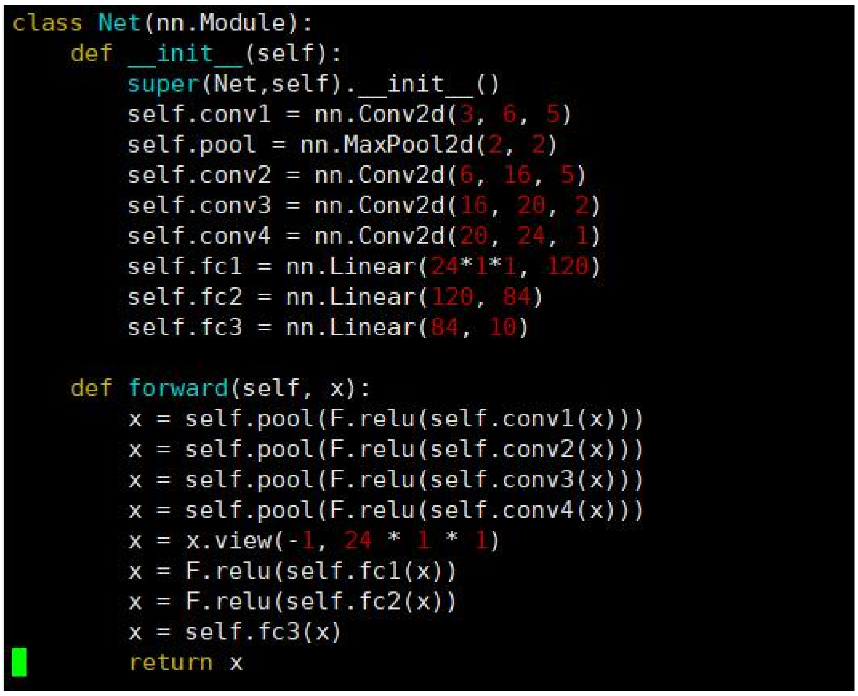


```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.conv3 =nn.Conv2d(16, 28, 2)
        self.fc1 = nn.Linear(28*2*2, 120)
        self.fc2 = 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 = self.pool(F.relu(self.conv3(x)))
        x = x.view(-1, 28 * 2 * 2)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

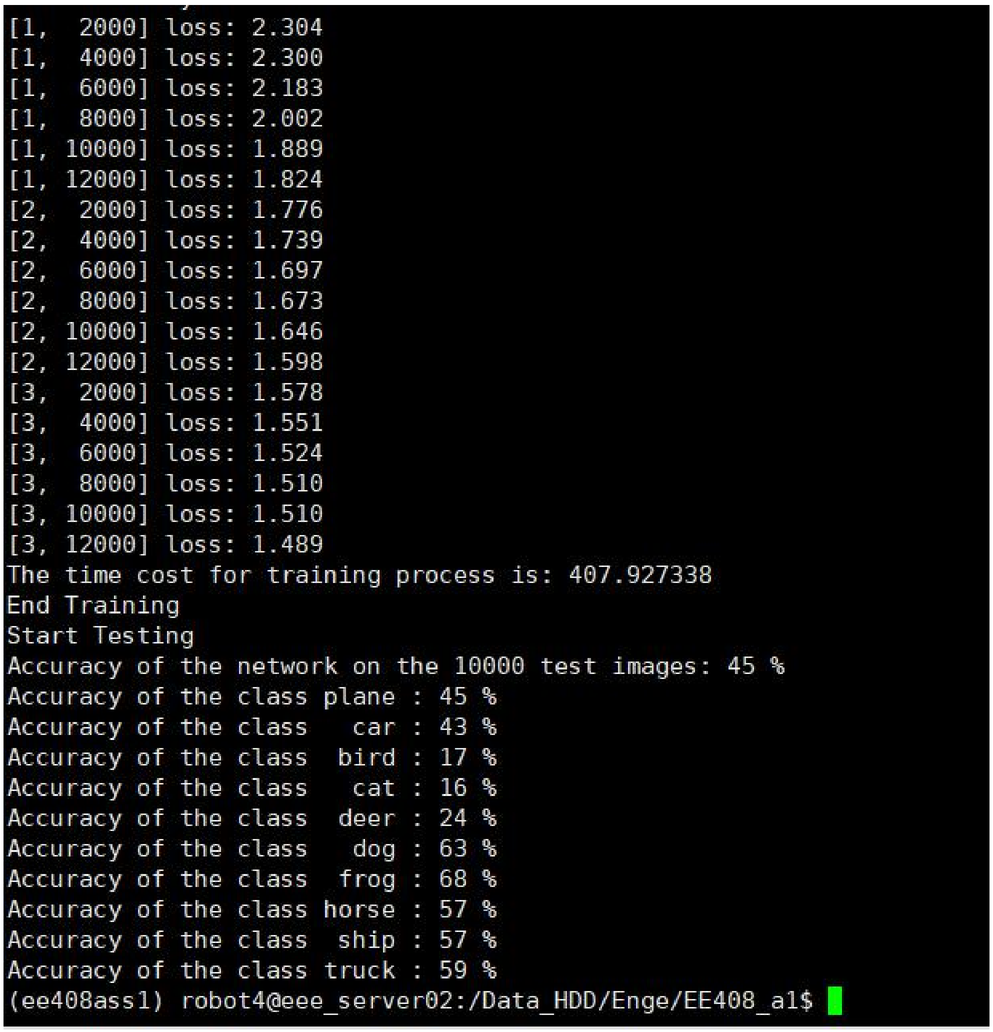


```
[1,  2000] loss: 2.302
[1,  4000] loss: 2.184
[1,  6000] loss: 1.916
[1,  8000] loss: 1.809
[1, 10000] loss: 1.709
[1, 12000] loss: 1.654
[2,  2000] loss: 1.561
[2,  4000] loss: 1.530
[2,  6000] loss: 1.502
[2,  8000] loss: 1.465
[2, 10000] loss: 1.440
[2, 12000] loss: 1.413
[3,  2000] loss: 1.401
[3,  4000] loss: 1.352
[3,  6000] loss: 1.347
[3,  8000] loss: 1.332
[3, 10000] loss: 1.335
[3, 12000] loss: 1.329
The time cost for training process is: 331.032603
End Training
Start Testing
Accuracy of the network on the 10000 test images: 53 %
Accuracy of the class plane : 44 %
Accuracy of the class   car : 72 %
Accuracy of the class  bird : 31 %
Accuracy of the class   cat : 46 %
Accuracy of the class  deer : 57 %
Accuracy of the class   dog : 26 %
Accuracy of the class  frog : 52 %
Accuracy of the class horse : 67 %
Accuracy of the class  ship : 71 %
Accuracy of the class truck : 64 %
(ee408ass1) robot4@eee_server02:/Data_HDD/Enge/EE408_a1$
```

Four layer;



```
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.conv3 = nn.Conv2d(16, 20, 2)
        self.conv4 = nn.Conv2d(20, 24, 1)
        self.fc1 = nn.Linear(24*1*1, 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 = self.pool(F.relu(self.conv3(x)))
        x = self.pool(F.relu(self.conv4(x)))
        x = x.view(-1, 24 * 1 * 1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```



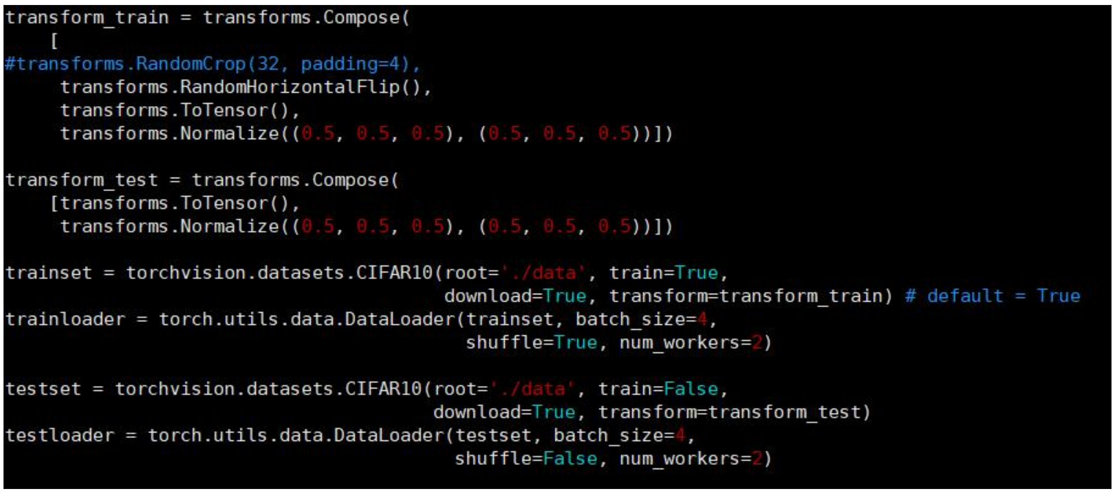
```
[1,  2000] loss: 2.304
[1,  4000] loss: 2.300
[1,  6000] loss: 2.183
[1,  8000] loss: 2.002
[1, 10000] loss: 1.889
[1, 12000] loss: 1.824
[2,  2000] loss: 1.776
[2,  4000] loss: 1.739
[2,  6000] loss: 1.697
[2,  8000] loss: 1.673
[2, 10000] loss: 1.646
[2, 12000] loss: 1.598
[3,  2000] loss: 1.578
[3,  4000] loss: 1.551
[3,  6000] loss: 1.524
[3,  8000] loss: 1.510
[3, 10000] loss: 1.510
[3, 12000] loss: 1.489
The time cost for training process is: 407.927338
End Training
Start Testing
Accuracy of the network on the 10000 test images: 45 %
Accuracy of the class plane : 45 %
Accuracy of the class   car : 43 %
Accuracy of the class  bird : 17 %
Accuracy of the class   cat : 16 %
Accuracy of the class  deer : 24 %
Accuracy of the class   dog : 63 %
Accuracy of the class  frog : 68 %
Accuracy of the class horse : 57 %
Accuracy of the class  ship : 57 %
Accuracy of the class truck : 59 %
(ee408ass1) robot4@eee_server02:/Data_HDD/Enge/EE408_a1$
```

TASK 4

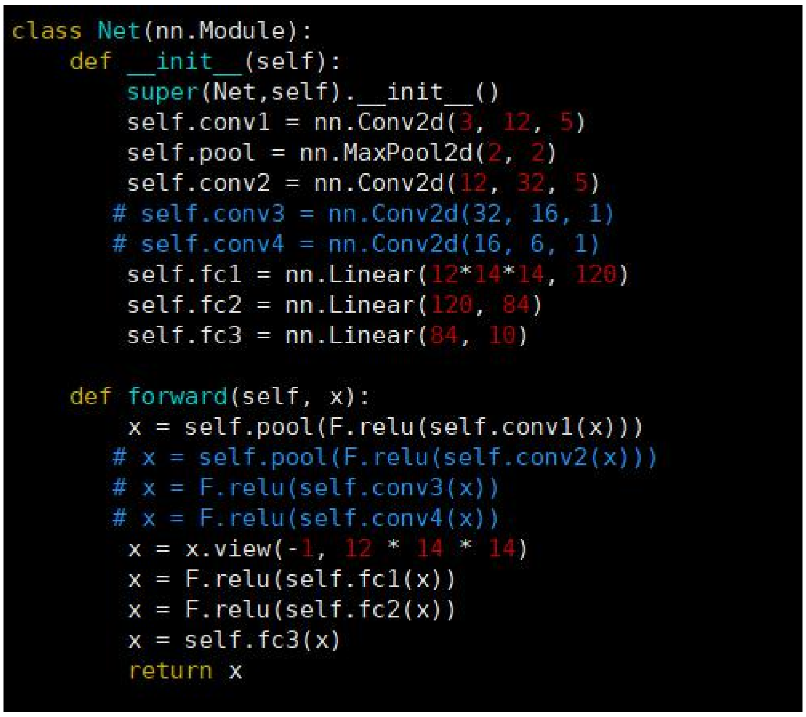
In image classification task, the image data to enhance general is one of the most people will use a method, this is due to the deep learning have a certain request for the size of the data set, if the original data set is small, can't well satisfy the training of the network model, which affect the performance of the model, and the image enhancement is to a certain processing to expand the original image data sets, can to a certain extent, improve the performance of the model.

In this task, use the randomhorizontalflip function to flip the picture horizontally with probability 0.5. It is only used in train set. By this way, the training data can be changed before every epoch starts. So every epoch dealt with images differently and data enhancement was achieved.

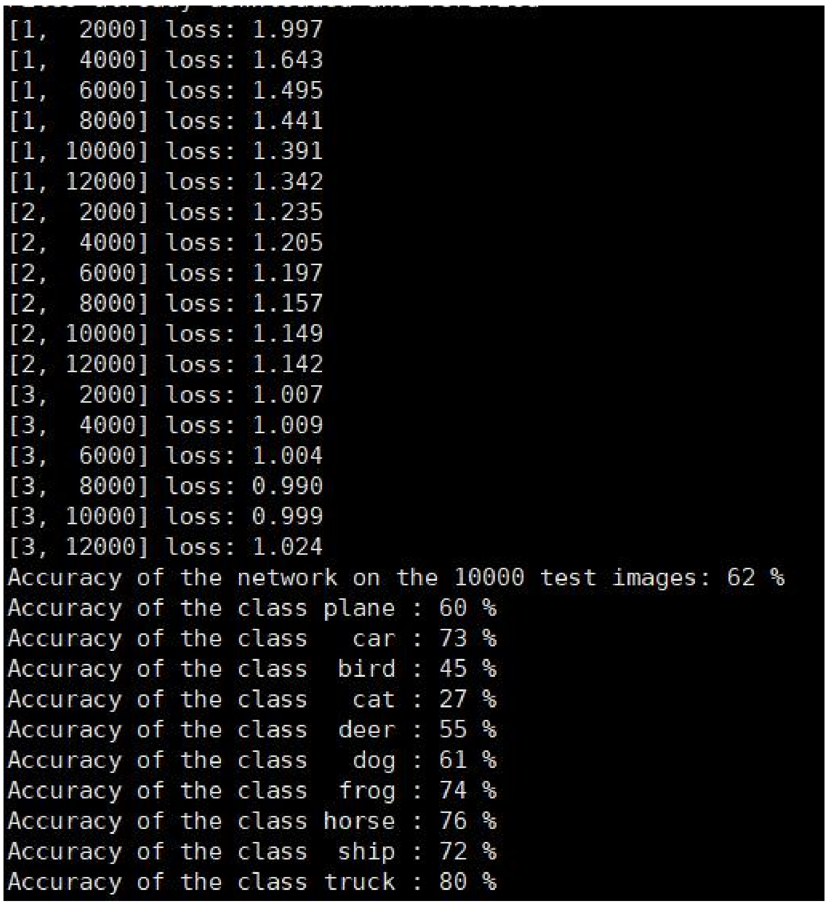
Due to the training epoch is not enough(time limited), the layer should be reduced and the number of filter should be improved to deal with the relatively large and random train database.



```
transform_train = transforms.Compose(
    [
#transforms.RandomCrop(32, padding=4),
     transforms.RandomHorizontalFlip(),
     transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

transform_test = 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_train) # default = True
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_test)
testloader = torch.utils.data.DataLoader(testset, batch_size=4,
                                         shuffle=False, num_workers=2)
```



```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(3, 12, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(12, 32, 5)
       # self.conv3 = nn.Conv2d(32, 16, 1)
       # self.conv4 = nn.Conv2d(16, 6, 1)
        self.fc1 = nn.Linear(12*14*14, 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 = F.relu(self.conv3(x))
       # x = F.relu(self.conv4(x))
        x = x.view(-1, 12 * 14 * 14)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```



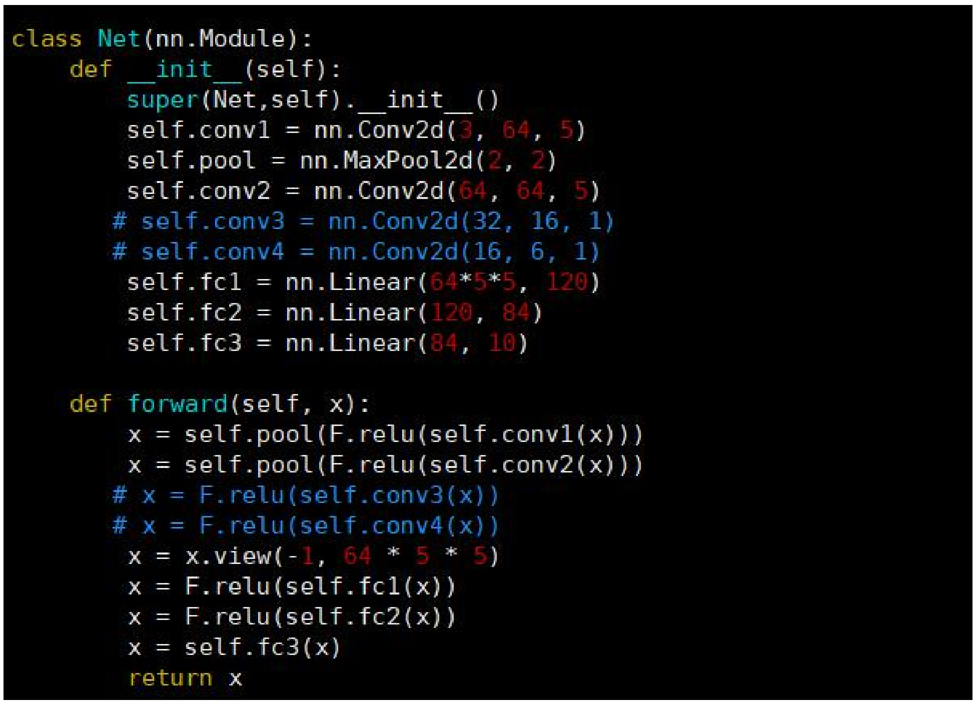
```
[1,  2000] loss: 1.997
[1,  4000] loss: 1.643
[1,  6000] loss: 1.495
[1,  8000] loss: 1.441
[1, 10000] loss: 1.391
[1, 12000] loss: 1.342
[2,  2000] loss: 1.235
[2,  4000] loss: 1.205
[2,  6000] loss: 1.197
[2,  8000] loss: 1.157
[2, 10000] loss: 1.149
[2, 12000] loss: 1.142
[3,  2000] loss: 1.007
[3,  4000] loss: 1.009
[3,  6000] loss: 1.004
[3,  8000] loss: 0.990
[3, 10000] loss: 0.999
[3, 12000] loss: 1.024
Accuracy of the network on the 10000 test images: 62 %
Accuracy of the class plane : 60 %
Accuracy of the class   car : 73 %
Accuracy of the class  bird : 45 %
Accuracy of the class   cat : 27 %
Accuracy of the class  deer : 55 %
Accuracy of the class   dog : 61 %
Accuracy of the class  frog : 74 %
Accuracy of the class horse : 76 %
Accuracy of the class  ship : 72 %
Accuracy of the class truck : 80 %
```

TASK 5

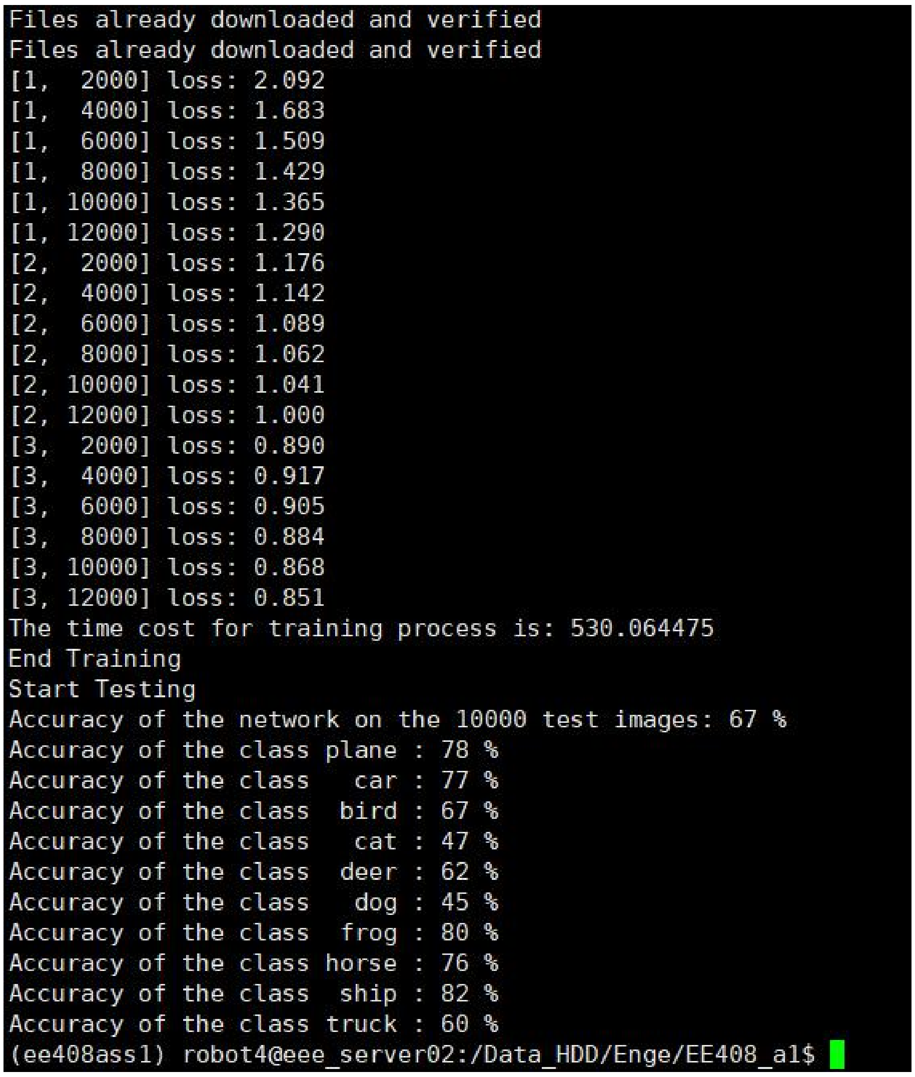
As an open task, several different networks are tested, including GoogLeNet, VGG, ResNet-18. After comparing the differences among different talent solutions, as a qualified neural network, lightweight with similar accuracy is the current trend. The same weakness among these networks is the low accuracy and time consuming among the first several epochs(Maybe the different accuracy account methods between average accuracy and top 5 error accuracy).

Based on the original code, the modified code is to change the convolution layer number. The basic idea is that the original code is the transform of LeNet network which only contain 32x32x1 image input. The RGB image input increase the information so the network should be modified to satisfy this change. One way to solve the problem is to enlarge the depth of the original network number or layer.

After testing, change the first layer into 64 filters and modify the second convolution layer can achieve good performance. It can increase the accuracy of test from 58% to 67% in three epochs and increase the accuracy of test from 62% to 72% in ten epochs. It achieve the highest improvement in the one step among data argument, deeper layer , learning rate and so on.



```python
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(3, 64, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(64, 64, 5)
       # self.conv3 = nn.Conv2d(32, 16, 1)
       # self.conv4 = nn.Conv2d(16, 6, 1)
        self.fc1 = nn.Linear(64*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 = F.relu(self.conv3(x))
       # x = F.relu(self.conv4(x))
        x = x.view(-1, 64 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```



```
Files already downloaded and verified
Files already downloaded and verified
[1,  2000] loss: 2.092
[1,  4000] loss: 1.683
[1,  6000] loss: 1.509
[1,  8000] loss: 1.429
[1, 10000] loss: 1.365
[1, 12000] loss: 1.290
[2,  2000] loss: 1.176
[2,  4000] loss: 1.142
[2,  6000] loss: 1.089
[2,  8000] loss: 1.062
[2, 10000] loss: 1.041
[2, 12000] loss: 1.000
[3,  2000] loss: 0.890
[3,  4000] loss: 0.917
[3,  6000] loss: 0.905
[3,  8000] loss: 0.884
[3, 10000] loss: 0.868
[3, 12000] loss: 0.851
The time cost for training process is: 530.064475
End Training
Start Testing
Accuracy of the network on the 10000 test images: 67 %
Accuracy of the class plane : 78 %
Accuracy of the class   car : 77 %
Accuracy of the class  bird : 67 %
Accuracy of the class   cat : 47 %
Accuracy of the class  deer : 62 %
Accuracy of the class   dog : 45 %
Accuracy of the class  frog : 80 %
Accuracy of the class horse : 76 %
Accuracy of the class  ship : 82 %
Accuracy of the class truck : 60 %
(ee408ass1) robot4@eee_server02:/Data_HDD/Enge/EE408_a1$
```

CONCLUSION

Through this experiment, I have mastered the basic method for simple pythorch programming, and for related functions also have a deep understanding, and can skilled application. The most important, is to master the internal principle of convolution layer and pooling layer. Fundamentally find the changes between different code and function, instead of limit to the surface of function application.

At the same time, in the experimental process, it also appeared many problems, such as the influence of parameters on the results, similar functions, differences, etc.. But through reading related books, search information, constantly test, I solved problems one by one. It is a good exercise of finding the problem, the ability to solve the problem.

Finally, thanks to the teacher giving us such an opportunity to learn, exercise, guidance. In the learning process in the future, I will make persistent efforts, fight for better complete of the task.