playground4-basic-4layer-hyperparamsprntfrmt_AlLyrs_ExprmntCmnts2

October 21, 2022

1 Blog - CNN - Caltech101 | Airplanes, Motorbikes & Schooners

[32]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import torch
from torchvision import datasets,transforms,io
from torch import utils
from collections import Counter
```

[49]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

Loading Images:

First step of any Neural Network/Machine Learning problem is the loading the input data. Here in this cell we will load the given input images. The problem set gives us different types of images of Bikes, Airplanes, Schooners. And we have to create a neural network model that will classify a given image in to these three categories. Here given images are in different shapes so while loading these images we will transform them into 512x512 pixels size and then convert them into Tensors. Tensor is a numpy array like data structure which is developed for handling arrays of large size for example image arrays.

Next, we will split our input dataset into three sub datasets i.e. training dataset, validation dataset and testing dataset. Here we have kept around 20% of total data aside for testing and remaining 80% data we will use for traning and validation.

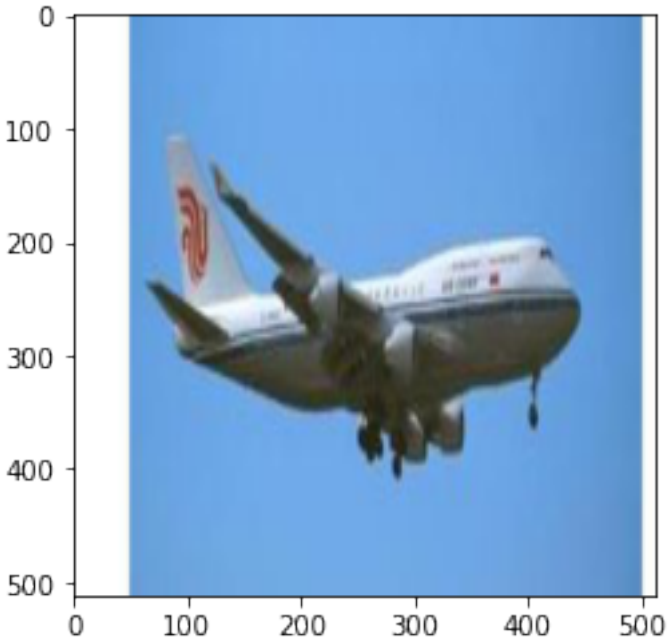
We will convert these three datasets into respective dataloader object. Dataloader object will be used for iterating over the these data and divide them in batches.

[50]:
```
# for i,data in enumerate(trainDataLoader):
#     print(len(data[0]))
```

[33]:
```
transform = transforms.Compose([transforms.Resize((512,512)),transforms.
 ↪ToTensor()])
dataset = datasets.ImageFolder('../input/
 ↪caltech101-airplanes-motorbikes-schooners/caltech101_classification/
 ↪',transform=transform)
```

```
trainData,validationData,testData = utils.data.
 ↪random_split(dataset,[930,400,331],generator=torch.Generator().
 ↪manual_seed(42))
plt.imshow(trainData[230][0].permute(1,2,0))
trainDataLoader = torch.utils.data.DataLoader(trainData, batch_size=32,␣
 ↪shuffle=True)
validationDataLoder = torch.utils.data.DataLoader(validationData,␣
 ↪batch_size=32, shuffle=True)
testDataLoader = torch.utils.data.DataLoader(testData, batch_size=32,␣
 ↪shuffle=True)
len(validationDataLoder.dataset)
```

[33]: 400



[52]:
```
# validationDataLoder
```

Creating CNN Architectures:

In this code block we have defined the architecture of our Convolutional Networks. We have created 3 convolutional layers and 2 fully connected layers. Each convolutional layer will have a pooling layer attached after it. This block of code is the essence of our CNN. Brief information about these layers is as mentioned below. **Conv Layer:** In this layer number of kernels of size specified by user will be used to convolve on the given input image matrix. Kernel is basically a small matrix which will be used to learn some specific feature in the image. For example, one kernel can be used to determine the vertical line in the image, another kernal can used to determine a curve in the image.So during training phase of the model, these kernal are learned and used later in testing

phase to determine that feature in the test image. Here the first layer, we have used a 16 kernels of size 3, stride=1 and padding=1. which will have the original image as an input and will output activation maps of dimensions 512x512x16. **ReLU:** ReLU is the activation function. It will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([512x512x16]).[1] **Pooling Layer:** As the size of the channels increases the weights associated with them also increases which leads to performance degredation. So, to solve this problem pooling layer is used. Pooling layer is used to reduce the size of input image. It will use a pooling technique to reduce the size image. Max pooling is the type of pooling which will convolve an nxn kernel on the image and will select the maximum element from the window. It will output an image of dimensions W2=(W1−F)/S+1 H2=(H1−F)/S+1 D2=D1 where W2, H2 are the dimensions of the output image and W1,H1 are the dimensions of input image and F is size of the pooling kernel and S is the stride. In CNN multiple conv layer and pool layers will be chained together, so output of the previous layer will be the input of next layer. ***Fully connected Layer:*** The fully-connected layer is called the "output layer" and in classification settings it represents the class scores.[1] The last fully connected layer will have output channesl numers equal to the number of classes in problem. For this classification problem we have three classes (Motorbike, Ariplane, Schooner) so our fully connected layer will have 3 as output channel and will output the volumn of [1x1x3].

Creating Different CNN Models Here we will create 3 different CNN model architecures and move these models to GPU if GPU is available. We will test the peroformance of these CNN models and compare them.

[34]:
```
class CNN2(torch.nn.Module):
#     __name__="Model2"
    def __init__(self):
        super(CNN2, self).__init__()
        ################################
        # Original Input image: (224,224,3)
        # Conv : (224,224,16)
        # Pool: (112,112,16)
        self.layer1 = torch.nn.Sequential(
            torch.nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ########################################
        # Input Image: (112,112,16)
        # Conv: (112,112,64)
        # Pool: (56,56,64)
        self.layer2 = torch.nn.Sequential(
            torch.nn.Conv2d(16, 64, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ########################################
        # FC 28*28*128 -> 625
```

```
        self.fc1 = torch.nn.Linear(128*128 * 64, 3, bias=True) # size of image␣
↪input to this layer * 128
        torch.nn.init.xavier_uniform_(self.fc1.weight)
        self.layer4 = torch.nn.Sequential(
            self.fc1,
            torch.nn.ReLU()
        )

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(out.size(0), -1)   # Flatten them for FC
        out = self.fc1(out)
        return out


#instantiate CNN model
model2 = CNN2()
model2
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model2.to(device)
```

[34]:
```
CNN2(
  (layer1): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(16, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (fc1): Linear(in_features=1048576, out_features=3, bias=True)
  (layer4): Sequential(
    (0): Linear(in_features=1048576, out_features=3, bias=True)
    (1): ReLU()
  )
)
```

[35]:
```
class CNN3(torch.nn.Module):
    __name__="Model3"
    def __init__(self):
        super(CNN3, self).__init__()
        ################################
```

```
        # Original Input image: (512,512,3)
        # Conv : (512,512,16)
        # Pool: (256,256,16)
        self.layer1 = torch.nn.Sequential(
            torch.nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ########################################
        # Input Image: (256,256,16)
        # Conv: (256,256,64)
        # Pool: (128,128,64)
        self.layer2 = torch.nn.Sequential(
            torch.nn.Conv2d(16, 64, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ################################
        # Input image: (128,128,64)
        # Conv : (128,128,128)
        # Pool: (64,64,128)
        self.layer3 = torch.nn.Sequential(
            torch.nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ############################################
        # FC 28*28*128 -> 625
        self.fc1 = torch.nn.Linear(64*64 * 128, 256, bias=True) # size of image␣
↪input to this layer * 128
        torch.nn.init.xavier_uniform_(self.fc1.weight)
        self.layer4 = torch.nn.Sequential(
            self.fc1,
            torch.nn.ReLU()
        )
        ############################################
        # FC 256 -> 3 Classes
        self.fc2 = torch.nn.Linear(256, 3, bias=True) # size of image input to␣
↪this layer * 128
        torch.nn.init.xavier_uniform_(self.fc2.weight)
        self.layer5 = torch.nn.Sequential(
            self.fc2,
            torch.nn.ReLU()
        )

    def forward(self, x):
        out = self.layer1(x)
```

```
        out = self.layer2(out)
        out = self.layer3(out)
        out = out.view(out.size(0), -1)   # Flatten them for FC
        out = self.fc1(out)
        out = self.fc2(out)
        return out


#instantiate CNN model
model3 = CNN3()
model3
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model3.to(device)
```

[35]:
```
CNN3(
  (layer1): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(16, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (layer3): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (fc1): Linear(in_features=524288, out_features=256, bias=True)
  (layer4): Sequential(
    (0): Linear(in_features=524288, out_features=256, bias=True)
    (1): ReLU()
  )
  (fc2): Linear(in_features=256, out_features=3, bias=True)
  (layer5): Sequential(
    (0): Linear(in_features=256, out_features=3, bias=True)
    (1): ReLU()
  )
)
```

[36]:
```
class CNN4(torch.nn.Module):
    __name__="Model4"
```

```
    def __init__(self):
        super(CNN4, self).__init__()
        ################################
        # Original Input image: (512,512,3)
        # Conv : (512,512,16)
        # Pool: (256,256,16)
        self.layer1 = torch.nn.Sequential(
            torch.nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ########################################
        # Input Image: (256,256,16)
        # Conv: (256,256,64)
        # Pool: (128,128,64)
        self.layer2 = torch.nn.Sequential(
            torch.nn.Conv2d(16, 64, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ################################
        # Input image: (128,128,64)
        # Conv : (128,128,128)
        # Pool: (64,64,128)
        self.layer3 = torch.nn.Sequential(
            torch.nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ################################
        # Input image: (64,64,128)
        # Conv : (64,64,512)
        # Pool: (32,32,512)
        self.layer4 = torch.nn.Sequential(
            torch.nn.Conv2d(128, 512, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ############################################
        # FC 28*28*128 -> 625
        self.fc1 = torch.nn.Linear(32*32 * 512, 256, bias=True) # size of image␣
↪input to this layer * 128
        torch.nn.init.xavier_uniform_(self.fc1.weight)
        self.layer5 = torch.nn.Sequential(
            self.fc1,
            torch.nn.ReLU()
        )
```

```
        ##############################################
        # FC 28*28*128 -> 625
        self.fc2 = torch.nn.Linear(256,512, bias=True) # size of image input to␣
↪this layer * 128
        torch.nn.init.xavier_uniform_(self.fc2.weight)
        self.layer6 = torch.nn.Sequential(
            self.fc2,
            torch.nn.ReLU()
        )
        ##############################################
        # FC 256 -> 3 Classes
        self.fc3 = torch.nn.Linear(512, 3, bias=True) # size of image input to␣
↪this layer * 128
        torch.nn.init.xavier_uniform_(self.fc3.weight)
        self.layer7 = torch.nn.Sequential(
            self.fc3,
            torch.nn.ReLU()
        )

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = out.view(out.size(0), -1)   # Flatten them for FC
        out = self.fc1(out)
        out = self.fc2(out)
        out = self.fc3(out)
        return out


#instantiate CNN model
model4 = CNN4()
model4
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model4.to(device)
```

[36]:
```
CNN4(
  (layer1): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(16, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
```

```
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (layer3): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (layer4): Sequential(
    (0): Conv2d(128, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (fc1): Linear(in_features=524288, out_features=256, bias=True)
  (layer5): Sequential(
    (0): Linear(in_features=524288, out_features=256, bias=True)
    (1): ReLU()
  )
  (fc2): Linear(in_features=256, out_features=512, bias=True)
  (layer6): Sequential(
    (0): Linear(in_features=256, out_features=512, bias=True)
    (1): ReLU()
  )
  (fc3): Linear(in_features=512, out_features=3, bias=True)
  (layer7): Sequential(
    (0): Linear(in_features=512, out_features=3, bias=True)
    (1): ReLU()
  )
)
```

Training CNN model: This block will be used to train the CNN model with the training dataset. "train_cnn_function()" function will take number of epochs, learning rate and models to use as input and use them to train the CNN model.

[37]:
```
def train_cnn_function(no_epochos,lr,model):
    import torch.optim as optim

    criterion = torch.nn.CrossEntropyLoss()
    optimizer = torch.optim.SGD(model.parameters(), lr=lr, momentum=0.9)

    no_of_epochos = no_epochos
    for epoch in range(no_of_epochos):
        running_loss = 0.0
        for i,data in enumerate(trainDataLoader):
#           if(i%10==0):
```

```
#             print("i=",i)
          inputData , lable = data[0].to(device), data[1].to(device)
          optimizer.zero_grad()
          output = model(inputData)
          loss = criterion(output,lable)
          loss.backward()
          optimizer.step()
          running_loss = running_loss+loss.item()
          if i % 5 == 4:    # print every 100 mini-batches
            print('Epoch={} Batch={} Loss= {}'.format(epoch + 1, i + 1,␣
 ↪running_loss / 5))
            running_loss = 0.0
    print("####Finished Training######")
```

Testing model: In this block we will test our model on validation and test dataset. "test_validation_function()" and "test_test_function()" will take model as input. First we will move our input data to GPU and then use our CNN model to predict the class of the images. For each image we will calculate loss and correct predictions. These same steps will be used for test dataset as well.

[38]:
```
def test_validation_function(model):
    valcorrect = 0
    valtotal = 0
    with torch.no_grad():
        for data in validationDataLoder:
            images, labels = data[0].to(device), data[1].to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            valtotal = valtotal + labels.size(0)
            valcorrect = valcorrect + (predicted == labels).sum().item()

    print('Accuracy of the network on validation images: %d %%' % (
        100 * valcorrect / valtotal))
    return(valcorrect / valtotal)
def test_test_function(model):
    testcorrect = 0
    testtotal = 0
    with torch.no_grad():
        for data in testDataLoader:
            images, labels = data[0].to(device), data[1].to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            testtotal = testtotal + labels.size(0)
            testcorrect = testcorrect + (predicted == labels).sum().item()

    print('Accuracy of the network on test images: %d %%' % (
        100 * testcorrect / testtotal))
```

```
    return(testcorrect / testtotal)
```

Experimentation/Contributions: In this block of code we will manually grid search different parameters combinations. Here we will test three different models with 2 layers, 3 layers and 4 layers. For each model we have trained our cnn model with different combinations of number of epoches and learning rate. For theese different combinations, we will save their prediction accuracy on validation set and their running time. We will choose the best parameters based on validation dataset accuracy and running time for training and use those parameters for test dataset.

[39]:
```
import itertools
from datetime import datetime
# test_accuracy=[]
no_epo_list =[1,3,5,7]
lr_list=[0.001,0.01]
overallData=[]
testAccuracy={}
combined = list(itertools.product(no_epo_list,lr_list))
models={'2':model2,'3':model3,'4':model4}
for model in models:
    validation_accuracy=[]
    runningTime=[]
    print("$$$$$$$$$$$$$$$$$$$$$$$$ For model with {} layers␣
↪$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$\n".
↪format(model))
    for i in range(len(combined)):
        print("Started training for parametrs: {}".format(combined[i]))
        start = datetime.now()
        train_cnn_function(combined[i][0],combined[i][1],models[model])
        end = datetime.now()
        print(end-start)
        totalTime=round((end - start).seconds+((end-start).microseconds /
↪1000000),4)
        runningTime.append(totalTime)
        result = test_validation_function(models[model])
        validation_accuracy.append(result)
    #     test_accuracy.append(result[1])
        print("Params: {},validation_accu:{}, time={} secs".
↪format(combined[i],result,totalTime))
        print("_____________________________________________________")
    # print(combined)
    # print(validation_accuracy)
    # print(test_accuracy)
              ####Table print#####
    from tabulate import tabulate
    data=[]
    for i in range(len(combined)):
```

```
        data.append([str(model)+"␣
↪layers",combined[i],validation_accuracy[i],runningTime[i]])
    print("\n")
    overallData.extend(data)
    print(tabulate(data,headers=['Model␣
↪Layers','Parameters(Epochs,LR)','Validation Accu','Running Time(seconds)']))
    bestIndex=-1
    minTime=float("inf")
    maxAccu=max(validation_accuracy)
    for i,v in enumerate(validation_accuracy):
        if(v==maxAccu):
            if(runningTime[i]<minTime):
                minTime=runningTime[i]
                bestIndex=i
    print("\nBest Parameters: {}, Validation Acuuracy: {}, Running Time: {}".
↪format(combined[bestIndex],validation_accuracy[bestIndex],runningTime[bestIndex]))
    print("\nBased on these parameters, the test accuracy is:\n")
   ␣
↪train_cnn_function(combined[bestIndex][0],combined[bestIndex][1],models[model])
    testResult=test_test_function(models[model])
    print(testResult)
    testAccuracy[model]=testResult
    for i in range(len(overallData)):
        if(int(overallData[i][0][0])==int(model)):
            if(overallData[i][1]==combined[bestIndex]):
                overallData[i].insert(1,round(testAccuracy[model]*100,3))
            else:
                overallData[i].insert(1," ")
   ␣
↪print("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

print("Overall Performance\n")
print(tabulate(overallData,headers=['Model Layers','Test Accu for Best␣
↪Parameters','Parameters(Epochs,LR)','Validation Accu','Running␣
↪Time(seconds)']))
```

```
$$$$$$$$$$$$$$$$$$$$$$$$ For model with 2 layers
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Started training for parametrs: (1, 0.001)
Epoch=1 Batch=5 Loss= 3.3674951553344727
Epoch=1 Batch=10 Loss= 1.5113786697387694
Epoch=1 Batch=15 Loss= 0.7572620272636413
Epoch=1 Batch=20 Loss= 0.5173098564147949
Epoch=1 Batch=25 Loss= 0.6130441188812256
Epoch=1 Batch=30 Loss= 0.3770324409008026
####Finished Training######
```

```
0:00:14.769318
Accuracy of the network on validation images: 89 %
Params: (1, 0.001),validation_accu:0.8975, time=14.7693 secs
----------------------------------------------------
Started training for parametrs: (1, 0.01)
Epoch=1 Batch=5 Loss= 0.5197314441204071
Epoch=1 Batch=10 Loss= 0.49592100381851195
Epoch=1 Batch=15 Loss= 0.9829528063535691
Epoch=1 Batch=20 Loss= 15.984781575202941
Epoch=1 Batch=25 Loss= 0.8886783480644226
Epoch=1 Batch=30 Loss= 0.9020085096359253
####Finished Training######
0:00:14.855420
Accuracy of the network on validation images: 53 %
Params: (1, 0.01),validation_accu:0.5325, time=14.8554 secs
----------------------------------------------------
Started training for parametrs: (3, 0.001)
Epoch=1 Batch=5 Loss= 0.8310508966445923
Epoch=1 Batch=10 Loss= 0.7120774865150452
Epoch=1 Batch=15 Loss= 0.594299191236496
Epoch=1 Batch=20 Loss= 0.4447524070739746
Epoch=1 Batch=25 Loss= 0.4119450807571411
Epoch=1 Batch=30 Loss= 0.40930811539292333
Epoch=2 Batch=5 Loss= 0.4726304769515991
Epoch=2 Batch=10 Loss= 0.4020621746778488
Epoch=2 Batch=15 Loss= 0.4776314914226532
Epoch=2 Batch=20 Loss= 0.41322217881679535
Epoch=2 Batch=25 Loss= 0.25175673365592954
Epoch=2 Batch=30 Loss= 0.36125091202557086
Epoch=3 Batch=5 Loss= 0.2513641893863678
Epoch=3 Batch=10 Loss= 0.26502406895160674
Epoch=3 Batch=15 Loss= 0.2343291833996773
Epoch=3 Batch=20 Loss= 0.23981145322322844
Epoch=3 Batch=25 Loss= 0.20822810828685762
Epoch=3 Batch=30 Loss= 0.32895279824733736
####Finished Training######
0:00:44.469294
Accuracy of the network on validation images: 86 %
Params: (3, 0.001),validation_accu:0.865, time=44.4693 secs
----------------------------------------------------
Started training for parametrs: (3, 0.01)
Epoch=1 Batch=5 Loss= 0.15097521468997002
Epoch=1 Batch=10 Loss= 0.9339207172393799
Epoch=1 Batch=15 Loss= 1.0838355779647828
Epoch=1 Batch=20 Loss= 1.054560661315918
Epoch=1 Batch=25 Loss= 1.020944094657898
Epoch=1 Batch=30 Loss= 0.9789568662643433
Epoch=2 Batch=5 Loss= 0.8791822552680969
```

```
Epoch=2 Batch=10 Loss= 0.9707504987716675
Epoch=2 Batch=15 Loss= 0.9548141956329346
Epoch=2 Batch=20 Loss= 0.9083970665931702
Epoch=2 Batch=25 Loss= 0.9061710000038147
Epoch=2 Batch=30 Loss= 0.8706798076629638
Epoch=3 Batch=5 Loss= 0.8405835151672363
Epoch=3 Batch=10 Loss= 0.957617461681366
Epoch=3 Batch=15 Loss= 0.9057376861572266
Epoch=3 Batch=20 Loss= 0.8582675099372864
Epoch=3 Batch=25 Loss= 0.7866731762886048
Epoch=3 Batch=30 Loss= 0.8380166053771972
####Finished Training######
0:00:43.948116
Accuracy of the network on validation images: 48 %
Params: (3, 0.01),validation_accu:0.4825, time=43.9481 secs
-----------------------------------------------------
Started training for parametrs: (5, 0.001)
Epoch=1 Batch=5 Loss= 0.885728633403778
Epoch=1 Batch=10 Loss= 0.8284660100936889
Epoch=1 Batch=15 Loss= 0.8236006140708924
Epoch=1 Batch=20 Loss= 0.840395724773407
Epoch=1 Batch=25 Loss= 0.7430536031723023
Epoch=1 Batch=30 Loss= 0.7793845653533935
Epoch=2 Batch=5 Loss= 0.7084688782691956
Epoch=2 Batch=10 Loss= 0.8759822726249695
Epoch=2 Batch=15 Loss= 0.741322910785675
Epoch=2 Batch=20 Loss= 0.6843673706054687
Epoch=2 Batch=25 Loss= 0.5630631864070892
Epoch=2 Batch=30 Loss= 0.6300214409828186
Epoch=3 Batch=5 Loss= 0.5122363448143006
Epoch=3 Batch=10 Loss= 0.5283590793609619
Epoch=3 Batch=15 Loss= 0.5172950863838196
Epoch=3 Batch=20 Loss= 0.43097885847091677
Epoch=3 Batch=25 Loss= 0.37543157637119295
Epoch=3 Batch=30 Loss= 0.435703307390213
Epoch=4 Batch=5 Loss= 0.42007399201393125
Epoch=4 Batch=10 Loss= 0.5194069266319274
Epoch=4 Batch=15 Loss= 0.390775328874588
Epoch=4 Batch=20 Loss= 0.28961786031723025
Epoch=4 Batch=25 Loss= 0.3125646531581879
Epoch=4 Batch=30 Loss= 0.41305142641067505
Epoch=5 Batch=5 Loss= 0.38128172159194945
Epoch=5 Batch=10 Loss= 0.3056245446205139
Epoch=5 Batch=15 Loss= 0.3821032166481018
Epoch=5 Batch=20 Loss= 0.28681778609752656
Epoch=5 Batch=25 Loss= 0.32804860472679137
Epoch=5 Batch=30 Loss= 0.25510757714509963
####Finished Training######
```

```
0:01:13.681759
Accuracy of the network on validation images: 92 %
Params: (5, 0.001),validation_accu:0.9275, time=73.6818 secs
----------------------------------------------------
Started training for parametrs: (5, 0.01)
Epoch=1 Batch=5 Loss= 0.23846447169780732
Epoch=1 Batch=10 Loss= 0.6445171535015106
Epoch=1 Batch=15 Loss= 0.8292264580726624
Epoch=1 Batch=20 Loss= 0.7757205367088318
Epoch=1 Batch=25 Loss= 0.7726304769515991
Epoch=1 Batch=30 Loss= 0.8770725846290588
Epoch=2 Batch=5 Loss= 0.5578522205352783
Epoch=2 Batch=10 Loss= 0.3808296024799347
Epoch=2 Batch=15 Loss= 0.3168769270181656
Epoch=2 Batch=20 Loss= 0.3190039932727814
Epoch=2 Batch=25 Loss= 0.3499178320169449
Epoch=2 Batch=30 Loss= 0.2873468786478043
Epoch=3 Batch=5 Loss= 0.16836130172014235
Epoch=3 Batch=10 Loss= 0.1741483137011528
Epoch=3 Batch=15 Loss= 0.12019851282238961
Epoch=3 Batch=20 Loss= 0.35662978440523146
Epoch=3 Batch=25 Loss= 0.23283300697803497
Epoch=3 Batch=30 Loss= 0.3243037313222885
Epoch=4 Batch=5 Loss= 0.20273930728435516
Epoch=4 Batch=10 Loss= 0.3258573591709137
Epoch=4 Batch=15 Loss= 0.20075261108577253
Epoch=4 Batch=20 Loss= 0.25462195575237273
Epoch=4 Batch=25 Loss= 0.1572013184428215
Epoch=4 Batch=30 Loss= 0.09575636367153492
Epoch=5 Batch=5 Loss= 0.0860686524771154
Epoch=5 Batch=10 Loss= 0.07829080000519753
Epoch=5 Batch=15 Loss= 0.1659482513088733
Epoch=5 Batch=20 Loss= 0.03607106674462557
Epoch=5 Batch=25 Loss= 0.07783496584743262
Epoch=5 Batch=30 Loss= 0.07668553851544857
####Finished Training######
0:01:13.414097
Accuracy of the network on validation images: 95 %
Params: (5, 0.01),validation_accu:0.955, time=73.4141 secs
----------------------------------------------------
Started training for parametrs: (7, 0.001)
Epoch=1 Batch=5 Loss= 0.04635193087160587
Epoch=1 Batch=10 Loss= 0.05677814967930317
Epoch=1 Batch=15 Loss= 0.033725216053426266
Epoch=1 Batch=20 Loss= 0.023239102214574814
Epoch=1 Batch=25 Loss= 0.042495880648493765
Epoch=1 Batch=30 Loss= 0.051482923608273265
Epoch=2 Batch=5 Loss= 0.011804598104208707
```

```
Epoch=2 Batch=10 Loss= 0.031835551373660564
Epoch=2 Batch=15 Loss= 0.04354059621691704
Epoch=2 Batch=20 Loss= 0.019770900020375848
Epoch=2 Batch=25 Loss= 0.01702315448783338
Epoch=2 Batch=30 Loss= 0.021795920841395856
Epoch=3 Batch=5 Loss= 0.023893525172024966
Epoch=3 Batch=10 Loss= 0.006953948969021439
Epoch=3 Batch=15 Loss= 0.03916551675647497
Epoch=3 Batch=20 Loss= 0.015838374383747578
Epoch=3 Batch=25 Loss= 0.012863151775673033
Epoch=3 Batch=30 Loss= 0.0272889587087775
Epoch=4 Batch=5 Loss= 0.01021727742627263
Epoch=4 Batch=10 Loss= 0.008645813446491957
Epoch=4 Batch=15 Loss= 0.030629338350263425
Epoch=4 Batch=20 Loss= 0.02713524820283055
Epoch=4 Batch=25 Loss= 0.019291380420327187
Epoch=4 Batch=30 Loss= 0.01696334984153509
Epoch=5 Batch=5 Loss= 0.008135749632492662
Epoch=5 Batch=10 Loss= 0.01433247965760529
Epoch=5 Batch=15 Loss= 0.01675231121480465
Epoch=5 Batch=20 Loss= 0.024215751118026672
Epoch=5 Batch=25 Loss= 0.014611131162382662
Epoch=5 Batch=30 Loss= 0.01982901983865304
Epoch=6 Batch=5 Loss= 0.005396313080564141
Epoch=6 Batch=10 Loss= 0.027814406901597977
Epoch=6 Batch=15 Loss= 0.006943251844495535
Epoch=6 Batch=20 Loss= 0.008341550547629594
Epoch=6 Batch=25 Loss= 0.006576969567686319
Epoch=6 Batch=30 Loss= 0.03290957435965467
Epoch=7 Batch=5 Loss= 0.01417563707800582
Epoch=7 Batch=10 Loss= 0.01963765420950949
Epoch=7 Batch=15 Loss= 0.008306387951597572
Epoch=7 Batch=20 Loss= 0.006541313463822007
Epoch=7 Batch=25 Loss= 0.009444220596924425
Epoch=7 Batch=30 Loss= 0.02201605698792264
####Finished Training######
0:01:42.658212
Accuracy of the network on validation images: 95 %
Params: (7, 0.001),validation_accu:0.955, time=102.6582 secs

----------------------------------------------------------
Started training for parametrs: (7, 0.01)
Epoch=1 Batch=5 Loss= 0.021686525270342826
Epoch=1 Batch=10 Loss= 0.0078075221506878735
Epoch=1 Batch=15 Loss= 0.02369622573023662
Epoch=1 Batch=20 Loss= 0.017003442160785198
Epoch=1 Batch=25 Loss= 0.011102679109899326
Epoch=1 Batch=30 Loss= 0.005080884759081528
Epoch=2 Batch=5 Loss= 0.005991184304002672
```

```
Epoch=2 Batch=10 Loss= 0.003443839168176055
Epoch=2 Batch=15 Loss= 0.05437174511607736
Epoch=2 Batch=20 Loss= 0.08906762647384311
Epoch=2 Batch=25 Loss= 0.13937767148017882
Epoch=2 Batch=30 Loss= 0.06910348907113076
Epoch=3 Batch=5 Loss= 0.15074920654296875
Epoch=3 Batch=10 Loss= 0.05817391499876976
Epoch=3 Batch=15 Loss= 0.10717485400382429
Epoch=3 Batch=20 Loss= 0.051007986441254614
Epoch=3 Batch=25 Loss= 0.10926215127110481
Epoch=3 Batch=30 Loss= 0.08911044596889042
Epoch=4 Batch=5 Loss= 0.030083353444933892
Epoch=4 Batch=10 Loss= 0.02598490414675325
Epoch=4 Batch=15 Loss= 0.01146834883838892
Epoch=4 Batch=20 Loss= 0.03157823220826685
Epoch=4 Batch=25 Loss= 0.046509708336088806
Epoch=4 Batch=30 Loss= 0.003176493558567017
Epoch=5 Batch=5 Loss= 0.011543311132118105
Epoch=5 Batch=10 Loss= 0.0303592402357026
Epoch=5 Batch=15 Loss= 0.01911489963531494
Epoch=5 Batch=20 Loss= 0.02117414935491979
Epoch=5 Batch=25 Loss= 0.027022673143073918
Epoch=5 Batch=30 Loss= 0.01334672379307449
Epoch=6 Batch=5 Loss= 0.0032485148145497077
Epoch=6 Batch=10 Loss= 0.0036268804222345354
Epoch=6 Batch=15 Loss= 0.001303813025879208
Epoch=6 Batch=20 Loss= 0.0016003187804017216
Epoch=6 Batch=25 Loss= 0.0038055282318964602
Epoch=6 Batch=30 Loss= 0.0004345653978816699
Epoch=7 Batch=5 Loss= 0.0021431798764751874
Epoch=7 Batch=10 Loss= 0.0014000397073687053
Epoch=7 Batch=15 Loss= 0.0018174937315052376
Epoch=7 Batch=20 Loss= 0.00037651881139026954
Epoch=7 Batch=25 Loss= 0.0004199268914817367
Epoch=7 Batch=30 Loss= 0.0004356287699920358
####Finished Training######
0:01:42.584910
Accuracy of the network on validation images: 96 %
Params: (7, 0.01),validation_accu:0.96, time=102.5849 secs
-----------------------------------------------------


Model Layers    Parameters(Epochs,LR)      Validation Accu    Running
Time(seconds)
--------------  ----------------------  -----------------
----------------------
2 layers        (1, 0.001)                          0.8975
14.7693
```

```
2 layers        (1, 0.01)                       0.5325
14.8554
2 layers        (3, 0.001)                      0.865
44.4693
2 layers        (3, 0.01)                       0.4825
43.9481
2 layers        (5, 0.001)                      0.9275
73.6818
2 layers        (5, 0.01)                       0.955
73.4141
2 layers        (7, 0.001)                      0.955
102.658
2 layers        (7, 0.01)                       0.96
102.585

Best Parameters: (7, 0.01), Validation Acuuracy: 0.96, Running Time: 102.5849

Based on these parameters, the test accuracy is:

Epoch=1 Batch=5 Loss= 0.00039959500136319546
Epoch=1 Batch=10 Loss= 0.000745483921491541
Epoch=1 Batch=15 Loss= 0.0004286846975446679
Epoch=1 Batch=20 Loss= 0.0001432554781786166
Epoch=1 Batch=25 Loss= 0.0016583651071414352
Epoch=1 Batch=30 Loss= 0.0005579321179538966
Epoch=2 Batch=5 Loss= 0.0003546169813489541
Epoch=2 Batch=10 Loss= 0.00027446415260783396
Epoch=2 Batch=15 Loss= 0.00026139957262785176
Epoch=2 Batch=20 Loss= 0.0004154009628109634
Epoch=2 Batch=25 Loss= 0.0008542089664842934
Epoch=2 Batch=30 Loss= 0.001230314717395231
Epoch=3 Batch=5 Loss= 0.0008699831385456491
Epoch=3 Batch=10 Loss= 0.0002937448502052575
Epoch=3 Batch=15 Loss= 0.00015733481395727723
Epoch=3 Batch=20 Loss= 0.0004691954381996766
Epoch=3 Batch=25 Loss= 0.00044017560285283255
Epoch=3 Batch=30 Loss= 0.00032921008169068956
Epoch=4 Batch=5 Loss= 0.00035558018753363286
Epoch=4 Batch=10 Loss= 0.00055144579091575
Epoch=4 Batch=15 Loss= 0.00027175450977665603
Epoch=4 Batch=20 Loss= 0.00025496841735730416
Epoch=4 Batch=25 Loss= 0.0005103260875330307
Epoch=4 Batch=30 Loss= 0.000158756766040824
Epoch=5 Batch=5 Loss= 0.00035820427983708215
Epoch=5 Batch=10 Loss= 0.00015946460471241152
Epoch=5 Batch=15 Loss= 0.0003261756122810766
Epoch=5 Batch=20 Loss= 0.0005722049580072052
Epoch=5 Batch=25 Loss= 0.0001534644892672077
```

```
Epoch=5 Batch=30 Loss= 0.00018969211889654502
Epoch=6 Batch=5 Loss= 0.0002028526519325169
Epoch=6 Batch=10 Loss= 0.00044589296230697074
Epoch=6 Batch=15 Loss= 0.0003749160346615099
Epoch=6 Batch=20 Loss= 0.0001785919077519793
Epoch=6 Batch=25 Loss= 0.00018896974215749652
Epoch=6 Batch=30 Loss= 0.0001752760697854683
Epoch=7 Batch=5 Loss= 0.0001769560234606153
Epoch=7 Batch=10 Loss= 0.00022282380668912083
Epoch=7 Batch=15 Loss= 0.00014548364451911767
Epoch=7 Batch=20 Loss= 0.0003648273486760445
Epoch=7 Batch=25 Loss= 0.00040097072942444354
Epoch=7 Batch=30 Loss= 9.633824628849653e-05
####Finished Training######
Accuracy of the network on test images: 95 %
0.9516616314199395
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$ For model with 3 layers
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Started training for parametrs: (1, 0.001)
Epoch=1 Batch=5 Loss= 0.8766646385192871
Epoch=1 Batch=10 Loss= 0.8271377801895141
Epoch=1 Batch=15 Loss= 0.6857269525527954
Epoch=1 Batch=20 Loss= 0.5484352171421051
Epoch=1 Batch=25 Loss= 0.44345263838768006
Epoch=1 Batch=30 Loss= 0.3300754949450493
####Finished Training######
0:00:17.748765
Accuracy of the network on validation images: 88 %
Params: (1, 0.001),validation_accu:0.8825, time=17.7488 secs
------------------------------------------------------
Started training for parametrs: (1, 0.01)
Epoch=1 Batch=5 Loss= 1.0585567116737367
Epoch=1 Batch=10 Loss= 0.5641977429389954
Epoch=1 Batch=15 Loss= 0.72071453332901
Epoch=1 Batch=20 Loss= 0.47145574986934663
Epoch=1 Batch=25 Loss= 0.47625739574432374
Epoch=1 Batch=30 Loss= 0.20007186536677182
####Finished Training######
0:00:18.221084
Accuracy of the network on validation images: 92 %
Params: (1, 0.01),validation_accu:0.9225, time=18.2211 secs
------------------------------------------------------
Started training for parametrs: (3, 0.001)
Epoch=1 Batch=5 Loss= 0.27821415662765503
```

```
Epoch=1 Batch=10 Loss= 0.1419877290725708
Epoch=1 Batch=15 Loss= 0.13897354751825333
Epoch=1 Batch=20 Loss= 0.12084504514932633
Epoch=1 Batch=25 Loss= 0.1334829479455948
Epoch=1 Batch=30 Loss= 0.14955673404037953
Epoch=2 Batch=5 Loss= 0.11100610122084617
Epoch=2 Batch=10 Loss= 0.18746655732393264
Epoch=2 Batch=15 Loss= 0.06036928743124008
Epoch=2 Batch=20 Loss= 0.11986766159534454
Epoch=2 Batch=25 Loss= 0.10817293804138899
Epoch=2 Batch=30 Loss= 0.06522189415991306
Epoch=3 Batch=5 Loss= 0.06317828446626664
Epoch=3 Batch=10 Loss= 0.05697928834706545
Epoch=3 Batch=15 Loss= 0.14153298400342465
Epoch=3 Batch=20 Loss= 0.06729271933436394
Epoch=3 Batch=25 Loss= 0.07450501546263695
Epoch=3 Batch=30 Loss= 0.14083009734749793
####Finished Training######
0:00:53.348365
Accuracy of the network on validation images: 94 %
Params: (3, 0.001),validation_accu:0.9475, time=53.3484 secs
-----------------------------------------------------
Started training for parametrs: (3, 0.01)
Epoch=1 Batch=5 Loss= 0.06898455917835236
Epoch=1 Batch=10 Loss= 0.08501963838934898
Epoch=1 Batch=15 Loss= 0.16382933966815472
Epoch=1 Batch=20 Loss= 0.32661850079894067
Epoch=1 Batch=25 Loss= 0.18113182112574577
Epoch=1 Batch=30 Loss= 0.09565377477556467
Epoch=2 Batch=5 Loss= 0.03529845904558897
Epoch=2 Batch=10 Loss= 0.05591105967760086
Epoch=2 Batch=15 Loss= 0.09120783358812332
Epoch=2 Batch=20 Loss= 0.05078956310171634
Epoch=2 Batch=25 Loss= 0.11918325405567884
Epoch=2 Batch=30 Loss= 0.20812938436865808
Epoch=3 Batch=5 Loss= 1.1458428144454955
Epoch=3 Batch=10 Loss= 0.7314771294593811
Epoch=3 Batch=15 Loss= 0.5758923172950745
Epoch=3 Batch=20 Loss= 0.40205429792404174
Epoch=3 Batch=25 Loss= 0.2905722141265869
Epoch=3 Batch=30 Loss= 0.11099781207740307
####Finished Training######
0:00:53.525526
Accuracy of the network on validation images: 77 %
Params: (3, 0.01),validation_accu:0.77, time=53.5255 secs
-----------------------------------------------------
Started training for parametrs: (5, 0.001)
Epoch=1 Batch=5 Loss= 0.2703444242477417
```

```
Epoch=1 Batch=10 Loss= 0.2796662002801895
Epoch=1 Batch=15 Loss= 0.14748874604701995
Epoch=1 Batch=20 Loss= 0.08565299231559038
Epoch=1 Batch=25 Loss= 0.09811017885804177
Epoch=1 Batch=30 Loss= 0.07661209776997566
Epoch=2 Batch=5 Loss= 0.10562055334448814
Epoch=2 Batch=10 Loss= 0.06765211746096611
Epoch=2 Batch=15 Loss= 0.06363923028111458
Epoch=2 Batch=20 Loss= 0.08284222111105918
Epoch=2 Batch=25 Loss= 0.03407407253980636
Epoch=2 Batch=30 Loss= 0.1244045615196228
Epoch=3 Batch=5 Loss= 0.0451083704829216
Epoch=3 Batch=10 Loss= 0.08379135988652706
Epoch=3 Batch=15 Loss= 0.09467009603977203
Epoch=3 Batch=20 Loss= 0.06741597726941109
Epoch=3 Batch=25 Loss= 0.023550234362483026
Epoch=3 Batch=30 Loss= 0.02771017923951149
Epoch=4 Batch=5 Loss= 0.05609755367040634
Epoch=4 Batch=10 Loss= 0.04066205359995365
Epoch=4 Batch=15 Loss= 0.011487547587603331
Epoch=4 Batch=20 Loss= 0.03825008757412433
Epoch=4 Batch=25 Loss= 0.0774447776377201
Epoch=4 Batch=30 Loss= 0.02090605272242101
Epoch=5 Batch=5 Loss= 0.05037818383425474
Epoch=5 Batch=10 Loss= 0.023846224322915076
Epoch=5 Batch=15 Loss= 0.035103052575141194
Epoch=5 Batch=20 Loss= 0.042869992554187775
Epoch=5 Batch=25 Loss= 0.03154545556753874
Epoch=5 Batch=30 Loss= 0.015879731229506432
####Finished Training######
0:01:28.595642
Accuracy of the network on validation images: 96 %
Params: (5, 0.001),validation_accu:0.9625, time=88.5956 secs
-----------------------------------------------------
Started training for parametrs: (5, 0.01)
Epoch=1 Batch=5 Loss= 0.05934976935386658
Epoch=1 Batch=10 Loss= 0.035385850071907046
Epoch=1 Batch=15 Loss= 0.07730707190930844
Epoch=1 Batch=20 Loss= 0.02665743175894022
Epoch=1 Batch=25 Loss= 0.14391184151172637
Epoch=1 Batch=30 Loss= 0.11264435946941376
Epoch=2 Batch=5 Loss= 0.4594600759446621
Epoch=2 Batch=10 Loss= 0.33608160018920896
Epoch=2 Batch=15 Loss= 0.2591403812170029
Epoch=2 Batch=20 Loss= 0.17175153195858
Epoch=2 Batch=25 Loss= 0.10378115996718407
Epoch=2 Batch=30 Loss= 0.04483740031719208
Epoch=3 Batch=5 Loss= 0.10741383945569397
```

```
Epoch=3 Batch=10 Loss= 0.03696914687752724
Epoch=3 Batch=15 Loss= 0.10873744953423739
Epoch=3 Batch=20 Loss= 0.053825640305876735
Epoch=3 Batch=25 Loss= 0.09891546554863453
Epoch=3 Batch=30 Loss= 0.03690473005408421
Epoch=4 Batch=5 Loss= 0.032249337807297704
Epoch=4 Batch=10 Loss= 0.012294356897473335
Epoch=4 Batch=15 Loss= 0.038096453389152886
Epoch=4 Batch=20 Loss= 0.07276296522468328
Epoch=4 Batch=25 Loss= 0.04975491035729647
Epoch=4 Batch=30 Loss= 0.011365118914727645
Epoch=5 Batch=5 Loss= 0.1601163052022457
Epoch=5 Batch=10 Loss= 0.020489256811561062
Epoch=5 Batch=15 Loss= 0.05355374335777015
Epoch=5 Batch=20 Loss= 0.07388761509209871
Epoch=5 Batch=25 Loss= 0.18446475444361568
Epoch=5 Batch=30 Loss= 0.18829859439283608
####Finished Training######
0:01:28.679302
Accuracy of the network on validation images: 92 %
Params: (5, 0.01),validation_accu:0.9275, time=88.6793 secs
-----------------------------------------------------
Started training for parametrs: (7, 0.001)
Epoch=1 Batch=5 Loss= 0.11403643805533648
Epoch=1 Batch=10 Loss= 0.11792769320309163
Epoch=1 Batch=15 Loss= 0.05160700306296349
Epoch=1 Batch=20 Loss= 0.038523805886507036
Epoch=1 Batch=25 Loss= 0.04845060035586357
Epoch=1 Batch=30 Loss= 0.06084290146827698
Epoch=2 Batch=5 Loss= 0.023762460984289645
Epoch=2 Batch=10 Loss= 0.02585272490978241
Epoch=2 Batch=15 Loss= 0.022653747349977493
Epoch=2 Batch=20 Loss= 0.01639862284064293
Epoch=2 Batch=25 Loss= 0.04392981715500355
Epoch=2 Batch=30 Loss= 0.021093747392296792
Epoch=3 Batch=5 Loss= 0.014043291099369526
Epoch=3 Batch=10 Loss= 0.02181673739105463
Epoch=3 Batch=15 Loss= 0.020594072341918946
Epoch=3 Batch=20 Loss= 0.018116593919694425
Epoch=3 Batch=25 Loss= 0.009216264076530933
Epoch=3 Batch=30 Loss= 0.013830076566591742
Epoch=4 Batch=5 Loss= 0.005663035530596971
Epoch=4 Batch=10 Loss= 0.011090228427201509
Epoch=4 Batch=15 Loss= 0.016807919554412364
Epoch=4 Batch=20 Loss= 0.011119432235136627
Epoch=4 Batch=25 Loss= 0.01569447824731469
Epoch=4 Batch=30 Loss= 0.016547043464379387
Epoch=5 Batch=5 Loss= 0.015376895759254694
```

```
Epoch=5 Batch=10 Loss= 0.01317431409843266
Epoch=5 Batch=15 Loss= 0.0049008212401531635
Epoch=5 Batch=20 Loss= 0.012883127573877573
Epoch=5 Batch=25 Loss= 0.011783220968209208
Epoch=5 Batch=30 Loss= 0.004097406561049866
Epoch=6 Batch=5 Loss= 0.01062302601058036
Epoch=6 Batch=10 Loss= 0.0026357237074989825
Epoch=6 Batch=15 Loss= 0.007231891504488885
Epoch=6 Batch=20 Loss= 0.006284841103479266
Epoch=6 Batch=25 Loss= 0.012503913440741598
Epoch=6 Batch=30 Loss= 0.012306972546502948
Epoch=7 Batch=5 Loss= 0.004699447331950069
Epoch=7 Batch=10 Loss= 0.013414434297010303
Epoch=7 Batch=15 Loss= 0.004117652674904093
Epoch=7 Batch=20 Loss= 0.0081374891102314
Epoch=7 Batch=25 Loss= 0.009372510050889104
Epoch=7 Batch=30 Loss= 0.005111343623121911
####Finished Training######
0:02:03.850845
Accuracy of the network on validation images: 97 %
Params: (7, 0.001),validation_accu:0.9725, time=123.8508 secs
-----------------------------------------------------
Started training for parametrs: (7, 0.01)
Epoch=1 Batch=5 Loss= 0.013457197067327797
Epoch=1 Batch=10 Loss= 0.005406077997758984
Epoch=1 Batch=15 Loss= 0.004873409774154425
Epoch=1 Batch=20 Loss= 0.0033858421724289657
Epoch=1 Batch=25 Loss= 0.01088785738684237
Epoch=1 Batch=30 Loss= 0.00516244291793484
Epoch=2 Batch=5 Loss= 0.009016601357143373
Epoch=2 Batch=10 Loss= 0.04973661285766866
Epoch=2 Batch=15 Loss= 0.009168713935650885
Epoch=2 Batch=20 Loss= 0.03431244310922921
Epoch=2 Batch=25 Loss= 0.018498904653824866
Epoch=2 Batch=30 Loss= 0.055136917158961296
Epoch=3 Batch=5 Loss= 0.0198928845115006
Epoch=3 Batch=10 Loss= 0.019026364129967988
Epoch=3 Batch=15 Loss= 0.003141512209549546
Epoch=3 Batch=20 Loss= 0.023764359322376548
Epoch=3 Batch=25 Loss= 0.020441414834931493
Epoch=3 Batch=30 Loss= 0.0031804868718438685
Epoch=4 Batch=5 Loss= 0.007429623976349831
Epoch=4 Batch=10 Loss= 0.007635387300979346
Epoch=4 Batch=15 Loss= 0.017557490605395288
Epoch=4 Batch=20 Loss= 0.0010315875260857865
Epoch=4 Batch=25 Loss= 0.0009525387642497663
Epoch=4 Batch=30 Loss= 0.004078011312230956
Epoch=5 Batch=5 Loss= 0.001545791156240739
```

```
Epoch=5 Batch=10 Loss= 0.00485322208260186
Epoch=5 Batch=15 Loss= 0.0020548475265968593
Epoch=5 Batch=20 Loss= 0.00041891363289323633
Epoch=5 Batch=25 Loss= 0.003369456564541906
Epoch=5 Batch=30 Loss= 0.0013100802723784
Epoch=6 Batch=5 Loss= 0.0004732110348413698
Epoch=6 Batch=10 Loss= 0.000643428535113344
Epoch=6 Batch=15 Loss= 0.0018963057227665558
Epoch=6 Batch=20 Loss= 0.0006846531148767098
Epoch=6 Batch=25 Loss= 0.0009596200514351949
Epoch=6 Batch=30 Loss= 0.00017773416675481712
Epoch=7 Batch=5 Loss= 0.0007002568003372289
Epoch=7 Batch=10 Loss= 0.0003625431368163845
Epoch=7 Batch=15 Loss= 0.0006468913234130014
Epoch=7 Batch=20 Loss= 0.0004265482209007132
Epoch=7 Batch=25 Loss= 0.00031734867734485307
Epoch=7 Batch=30 Loss= 0.00022568401982425712
####Finished Training######
0:02:03.874252
Accuracy of the network on validation images: 97 %
Params: (7, 0.01),validation_accu:0.975, time=123.8743 secs
-----------------------------------------------------


Model Layers    Parameters(Epochs,LR)      Validation Accu    Running
Time(seconds)
--------------  ----------------------  -----------------
----------------------
3 layers        (1, 0.001)                          0.8825
17.7488
3 layers        (1, 0.01)                           0.9225
18.2211
3 layers        (3, 0.001)                          0.9475
53.3484
3 layers        (3, 0.01)                           0.77
53.5255
3 layers        (5, 0.001)                          0.9625
88.5956
3 layers        (5, 0.01)                           0.9275
88.6793
3 layers        (7, 0.001)                          0.9725
123.851
3 layers        (7, 0.01)                           0.975
123.874

Best Parameters: (7, 0.01), Validation Acuuracy: 0.975, Running Time: 123.8743

Based on these parameters, the test accuracy is:
```

```
Epoch=1 Batch=5 Loss= 6.749948661308736e-05
Epoch=1 Batch=10 Loss= 0.0002409146269201301
Epoch=1 Batch=15 Loss= 0.0006260608668526401
Epoch=1 Batch=20 Loss= 0.00010026563759311103
Epoch=1 Batch=25 Loss= 0.00019130704931740182
Epoch=1 Batch=30 Loss= 0.0003423644811846316
Epoch=2 Batch=5 Loss= 0.0003534934157869429 3
Epoch=2 Batch=10 Loss= 0.00026382402757008094
Epoch=2 Batch=15 Loss= 0.00012504218502726872
Epoch=2 Batch=20 Loss= 0.00019648144298116675
Epoch=2 Batch=25 Loss= 0.0002076168972052983
Epoch=2 Batch=30 Loss= 0.0001423557256202912
Epoch=3 Batch=5 Loss= 6.497027352452278e-05
Epoch=3 Batch=10 Loss= 0.00042062879238073946
Epoch=3 Batch=15 Loss= 0.00026078345199493925
Epoch=3 Batch=20 Loss= 6.062423053663224e-05
Epoch=3 Batch=25 Loss= 6.790430106775602e-05
Epoch=3 Batch=30 Loss= 0.0002846060793672223
Epoch=4 Batch=5 Loss= 0.0001311449086642824
Epoch=4 Batch=10 Loss= 0.0002591866519651376
Epoch=4 Batch=15 Loss= 8.750645574764348e-05
Epoch=4 Batch=20 Loss= 0.00018813153728842736
Epoch=4 Batch=25 Loss= 0.00021959271034575068
Epoch=4 Batch=30 Loss= 4.239231557221501e-05
Epoch=5 Batch=5 Loss= 0.0001885068613773910 3
Epoch=5 Batch=10 Loss= 6.18727569872135e-05
Epoch=5 Batch=15 Loss= 5.149109383637551e-05
Epoch=5 Batch=20 Loss= 0.00020354290636532825
Epoch=5 Batch=25 Loss= 0.0002620048078824766
Epoch=5 Batch=30 Loss= 3.49264832038898e-05
Epoch=6 Batch=5 Loss= 8.080115712800761e-05
Epoch=6 Batch=10 Loss= 0.0001666151478275424
Epoch=6 Batch=15 Loss= 9.426341439393582e-05
Epoch=6 Batch=20 Loss= 0.00014931991699995706
Epoch=6 Batch=25 Loss= 0.00016418874183727895
Epoch=6 Batch=30 Loss= 5.7394036139157835e-05
Epoch=7 Batch=5 Loss= 5.593996420429903e-05
Epoch=7 Batch=10 Loss= 0.0001419128766428912
Epoch=7 Batch=15 Loss= 0.00010017684817285045
Epoch=7 Batch=20 Loss= 0.00015064695526234573
Epoch=7 Batch=25 Loss= 9.924304904416204e-05
Epoch=7 Batch=30 Loss= 9.271444905607495e-05
####Finished Training######
Accuracy of the network on test images: 96 %
0.9667673716012085
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
$$$$$$$$$$$$$$$$$$$$$$ For model with 4 layers
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Started training for parametrs: (1, 0.001)
Epoch=1 Batch=5 Loss= 0.9650701880455017
Epoch=1 Batch=10 Loss= 0.777970039844513
Epoch=1 Batch=15 Loss= 0.7386423468589782
Epoch=1 Batch=20 Loss= 0.7168580055236816
Epoch=1 Batch=25 Loss= 0.6893199861049653
Epoch=1 Batch=30 Loss= 0.5855120778083801
####Finished Training######
0:00:20.615611
Accuracy of the network on validation images: 80 %
Params: (1, 0.001),validation_accu:0.8025, time=20.6156 secs
-----------------------------------------------------
Started training for parametrs: (1, 0.01)
Epoch=1 Batch=5 Loss= 0.9720582962036133
Epoch=1 Batch=10 Loss= 0.8787791848182678
Epoch=1 Batch=15 Loss= 0.8434042930603027
Epoch=1 Batch=20 Loss= 0.8643285393714905
Epoch=1 Batch=25 Loss= 0.8353563785552979
Epoch=1 Batch=30 Loss= 0.854533576965332
####Finished Training######
0:00:20.406972
Accuracy of the network on validation images: 48 %
Params: (1, 0.01),validation_accu:0.4825, time=20.407 secs
-----------------------------------------------------
Started training for parametrs: (3, 0.001)
Epoch=1 Batch=5 Loss= 0.7741445899009705
Epoch=1 Batch=10 Loss= 0.9131012678146362
Epoch=1 Batch=15 Loss= 0.7699972987174988
Epoch=1 Batch=20 Loss= 0.7797850370407104
Epoch=1 Batch=25 Loss= 0.762433671951294
Epoch=1 Batch=30 Loss= 0.7741211295127869
Epoch=2 Batch=5 Loss= 0.8072435975074768
Epoch=2 Batch=10 Loss= 0.7116515755653381
Epoch=2 Batch=15 Loss= 0.7974410891532898
Epoch=2 Batch=20 Loss= 0.7629640340805054
Epoch=2 Batch=25 Loss= 0.7681439399719239
Epoch=2 Batch=30 Loss= 0.7079346776008606
Epoch=3 Batch=5 Loss= 0.6593295931816101
Epoch=3 Batch=10 Loss= 0.6203575611114502
Epoch=3 Batch=15 Loss= 0.6582143902778625
Epoch=3 Batch=20 Loss= 0.7684571385383606
Epoch=3 Batch=25 Loss= 0.645482850074768
Epoch=3 Batch=30 Loss= 0.5490439534187317
####Finished Training######
```

```
0:01:00.817043
Accuracy of the network on validation images: 84 %
Params: (3, 0.001),validation_accu:0.8425, time=60.817 secs
----------------------------------------------------
Started training for parametrs: (3, 0.01)
Epoch=1 Batch=5 Loss= 0.6067475318908692
Epoch=1 Batch=10 Loss= 0.4555278480052948
Epoch=1 Batch=15 Loss= 0.32030077278614044
Epoch=1 Batch=20 Loss= 0.18761567771434784
Epoch=1 Batch=25 Loss= 0.5031734824180603
Epoch=1 Batch=30 Loss= 0.625692355632782
Epoch=2 Batch=5 Loss= 0.28731268346309663
Epoch=2 Batch=10 Loss= 0.44902490377426146
Epoch=2 Batch=15 Loss= 0.23308750689029695
Epoch=2 Batch=20 Loss= 0.15232986509799956
Epoch=2 Batch=25 Loss= 0.16752803176641465
Epoch=2 Batch=30 Loss= 0.23905380368232726
Epoch=3 Batch=5 Loss= 0.13281368762254714
Epoch=3 Batch=10 Loss= 0.15247527584433557
Epoch=3 Batch=15 Loss= 0.09091696068644524
Epoch=3 Batch=20 Loss= 0.03179905377328396
Epoch=3 Batch=25 Loss= 0.20459638088941573
Epoch=3 Batch=30 Loss= 0.10239893849939108
####Finished Training######
0:01:01.146761
Accuracy of the network on validation images: 97 %
Params: (3, 0.01),validation_accu:0.975, time=61.1468 secs
----------------------------------------------------
Started training for parametrs: (5, 0.001)
Epoch=1 Batch=5 Loss= 0.1116538867354393
Epoch=1 Batch=10 Loss= 0.07970353364944457
Epoch=1 Batch=15 Loss= 0.05122361481189728
Epoch=1 Batch=20 Loss= 0.04616916440427303
Epoch=1 Batch=25 Loss= 0.05305340215563774
Epoch=1 Batch=30 Loss= 0.05312325321137905
Epoch=2 Batch=5 Loss= 0.03155597373843193
Epoch=2 Batch=10 Loss= 0.058934138342738154
Epoch=2 Batch=15 Loss= 0.0630121361464262
Epoch=2 Batch=20 Loss= 0.027366083674132825
Epoch=2 Batch=25 Loss= 0.04682749398052692
Epoch=2 Batch=30 Loss= 0.011169675935525447
Epoch=3 Batch=5 Loss= 0.04003205895423889
Epoch=3 Batch=10 Loss= 0.03552557288203388
Epoch=3 Batch=15 Loss= 0.028679649671539666
Epoch=3 Batch=20 Loss= 0.017984538339078426
Epoch=3 Batch=25 Loss= 0.02967072632163763
Epoch=3 Batch=30 Loss= 0.014369786018502851
Epoch=4 Batch=5 Loss= 0.02524701189249754
```

```
Epoch=4 Batch=10 Loss= 0.03203974328935146
Epoch=4 Batch=15 Loss= 0.017372413352131844
Epoch=4 Batch=20 Loss= 0.03893873179331422
Epoch=4 Batch=25 Loss= 0.04791176542639732
Epoch=4 Batch=30 Loss= 0.013324271139117628
Epoch=5 Batch=5 Loss= 0.05364520438015461
Epoch=5 Batch=10 Loss= 0.01916833482682705
Epoch=5 Batch=15 Loss= 0.03556446693837643
Epoch=5 Batch=20 Loss= 0.02799619026482105
Epoch=5 Batch=25 Loss= 0.022527416795492174
Epoch=5 Batch=30 Loss= 0.02563473195405095
####Finished Training######
0:01:42.018606
Accuracy of the network on validation images: 97 %
Params: (5, 0.001),validation_accu:0.9775, time=102.0186 secs
-------------------------------------------------------
Started training for parametrs: (5, 0.01)
Epoch=1 Batch=5 Loss= 0.01674861488863826
Epoch=1 Batch=10 Loss= 0.05809730626642704
Epoch=1 Batch=15 Loss= 0.04148835511878133
Epoch=1 Batch=20 Loss= 0.03597035072743893
Epoch=1 Batch=25 Loss= 0.06468449532985687
Epoch=1 Batch=30 Loss= 0.05034752170322463
Epoch=2 Batch=5 Loss= 0.029495740309357643
Epoch=2 Batch=10 Loss= 0.05976662375032902
Epoch=2 Batch=15 Loss= 0.03505545714870095
Epoch=2 Batch=20 Loss= 0.02985305115580559
Epoch=2 Batch=25 Loss= 0.028352854866534473
Epoch=2 Batch=30 Loss= 0.006425504048820585
Epoch=3 Batch=5 Loss= 0.01427673497237265
Epoch=3 Batch=10 Loss= 0.019744636467657985
Epoch=3 Batch=15 Loss= 0.06296591572463513
Epoch=3 Batch=20 Loss= 0.04074167087674141
Epoch=3 Batch=25 Loss= 0.02195752691477537
Epoch=3 Batch=30 Loss= 0.021957881375760734
Epoch=4 Batch=5 Loss= 0.050832238886505364
Epoch=4 Batch=10 Loss= 0.006763005338143557
Epoch=4 Batch=15 Loss= 0.010284369857981802
Epoch=4 Batch=20 Loss= 0.01142562720924616
Epoch=4 Batch=25 Loss= 0.004739635900477879
Epoch=4 Batch=30 Loss= 0.001920543142068709
Epoch=5 Batch=5 Loss= 0.0040952633833512666
Epoch=5 Batch=10 Loss= 0.01242464315946563
Epoch=5 Batch=15 Loss= 0.01981421603122726
Epoch=5 Batch=20 Loss= 0.046549115800007715
Epoch=5 Batch=25 Loss= 0.015750737860798836
Epoch=5 Batch=30 Loss= 0.012358448840677027
####Finished Training######
```

```
0:01:42.299164
Accuracy of the network on validation images: 98 %
Params: (5, 0.01),validation_accu:0.98, time=102.2992 secs
----------------------------------------------------
Started training for parametrs: (7, 0.001)
Epoch=1 Batch=5 Loss= 0.012506514647975564
Epoch=1 Batch=10 Loss= 0.007403862942010164
Epoch=1 Batch=15 Loss= 0.0038767669000662865
Epoch=1 Batch=20 Loss= 0.01725656297057867
Epoch=1 Batch=25 Loss= 0.009686843492090702
Epoch=1 Batch=30 Loss= 0.007528938887116965
Epoch=2 Batch=5 Loss= 0.007346650445833802
Epoch=2 Batch=10 Loss= 0.0037590278894640504
Epoch=2 Batch=15 Loss= 0.006923525978345424
Epoch=2 Batch=20 Loss= 0.006258218991570174
Epoch=2 Batch=25 Loss= 0.004290932114236057
Epoch=2 Batch=30 Loss= 0.002420494075931856
Epoch=3 Batch=5 Loss= 0.0025759661104530094
Epoch=3 Batch=10 Loss= 0.002806492825038731
Epoch=3 Batch=15 Loss= 0.004695069891749881
Epoch=3 Batch=20 Loss= 0.004706210130825639
Epoch=3 Batch=25 Loss= 0.002550392330158502
Epoch=3 Batch=30 Loss= 0.003154250152874738
Epoch=4 Batch=5 Loss= 0.0028825498884543777
Epoch=4 Batch=10 Loss= 0.002521451422944665
Epoch=4 Batch=15 Loss= 0.002775299776112661
Epoch=4 Batch=20 Loss= 0.0010072451084852218
Epoch=4 Batch=25 Loss= 0.002330156788229942
Epoch=4 Batch=30 Loss= 0.0038080394908320157
Epoch=5 Batch=5 Loss= 0.0022046197031158955
Epoch=5 Batch=10 Loss= 0.0020050166756846012
Epoch=5 Batch=15 Loss= 0.002305570081807673
Epoch=5 Batch=20 Loss= 0.0026724117808043955
Epoch=5 Batch=25 Loss= 0.0010475117167516145
Epoch=5 Batch=30 Loss= 0.0019758030597586186
Epoch=6 Batch=5 Loss= 0.001667990069836378
Epoch=6 Batch=10 Loss= 0.002743887429824099
Epoch=6 Batch=15 Loss= 0.000979958797688596
Epoch=6 Batch=20 Loss= 0.002666951762512326
Epoch=6 Batch=25 Loss= 0.0006397850462235511
Epoch=6 Batch=30 Loss= 0.0019233613624237479
Epoch=7 Batch=5 Loss= 0.0010669141134712844
Epoch=7 Batch=10 Loss= 0.0011768118944019078
Epoch=7 Batch=15 Loss= 0.002460748376324773
Epoch=7 Batch=20 Loss= 0.002693029955844395
Epoch=7 Batch=25 Loss= 0.0008398581936489791
Epoch=7 Batch=30 Loss= 0.0004994444934254716
####Finished Training######
```

```
0:02:22.627103
Accuracy of the network on validation images: 98 %
Params: (7, 0.001),validation_accu:0.985, time=142.6271 secs

----------------------------------------------------
Started training for parametrs: (7, 0.01)
Epoch=1 Batch=5 Loss= 0.0011505623144330457
Epoch=1 Batch=10 Loss= 0.0010124007996637375
Epoch=1 Batch=15 Loss= 0.0007216810059617274
Epoch=1 Batch=20 Loss= 0.002235731761902571
Epoch=1 Batch=25 Loss= 0.00040881025634007526
Epoch=1 Batch=30 Loss= 0.0018952044634261255
Epoch=2 Batch=5 Loss= 0.001133093354292214
Epoch=2 Batch=10 Loss= 0.0009668315069575328
Epoch=2 Batch=15 Loss= 0.0003226513923436869
Epoch=2 Batch=20 Loss= 0.0004296330531360582
Epoch=2 Batch=25 Loss= 0.0006676849092400516
Epoch=2 Batch=30 Loss= 0.00023884935799216577
Epoch=3 Batch=5 Loss= 0.0008467499283142388
Epoch=3 Batch=10 Loss= 0.00020242480932211037
Epoch=3 Batch=15 Loss= 0.00023261224159796257
Epoch=3 Batch=20 Loss= 0.0003052640258829342
Epoch=3 Batch=25 Loss= 0.0005474966921610758
Epoch=3 Batch=30 Loss= 0.0001639828347833827
Epoch=4 Batch=5 Loss= 0.00017513335078547242
Epoch=4 Batch=10 Loss= 0.0003755812325834995
Epoch=4 Batch=15 Loss= 0.0005177910832571797
Epoch=4 Batch=20 Loss= 0.00013996557840982858
Epoch=4 Batch=25 Loss= 0.0002823359485773835
Epoch=4 Batch=30 Loss= 0.0002501755740695444
Epoch=5 Batch=5 Loss= 0.0003756277819775278
Epoch=5 Batch=10 Loss= 0.00010481426888873103
Epoch=5 Batch=15 Loss= 0.00024174734817279387
Epoch=5 Batch=20 Loss= 0.00018175811565015464
Epoch=5 Batch=25 Loss= 0.00015408305007440503
Epoch=5 Batch=30 Loss= 0.00019281440363485557
Epoch=6 Batch=5 Loss= 0.00023945627544890157
Epoch=6 Batch=10 Loss= 0.0001172480835521128
Epoch=6 Batch=15 Loss= 0.00018206329896202077
Epoch=6 Batch=20 Loss= 0.0001268359508685535
Epoch=6 Batch=25 Loss= 0.00034865587804233655
Epoch=6 Batch=30 Loss= 4.192890628473833e-05
Epoch=7 Batch=5 Loss= 0.00018145609046769096
Epoch=7 Batch=10 Loss= 0.00028334902017377315
Epoch=7 Batch=15 Loss= 0.00014522990736622887
Epoch=7 Batch=20 Loss= 5.1253687706775965e-05
Epoch=7 Batch=25 Loss= 4.231749044265598e-05
Epoch=7 Batch=30 Loss= 0.0001680383320490364
####Finished Training######
```

```
0:02:22.827581
Accuracy of the network on validation images: 98 %
Params: (7, 0.01),validation_accu:0.9825, time=142.8276 secs
----------------------------------------------------


Model Layers    Parameters(Epochs,LR)      Validation Accu    Running
Time(seconds)
--------------  ----------------------  -----------------
----------------------
4 layers        (1, 0.001)                          0.8025
20.6156
4 layers        (1, 0.01)                           0.4825
20.407
4 layers        (3, 0.001)                          0.8425
60.817
4 layers        (3, 0.01)                           0.975
61.1468
4 layers        (5, 0.001)                          0.9775
102.019
4 layers        (5, 0.01)                           0.98
102.299
4 layers        (7, 0.001)                          0.985
142.627
4 layers        (7, 0.01)                           0.9825
142.828

Best Parameters: (7, 0.001), Validation Acuuracy: 0.985, Running Time: 142.6271

Based on these parameters, the test accuracy is:

Epoch=1 Batch=5 Loss= 6.517626421214118e-05
Epoch=1 Batch=10 Loss= 0.00021404675153462448
Epoch=1 Batch=15 Loss= 0.0001547794327052543
Epoch=1 Batch=20 Loss= 7.076230613165536e-05
Epoch=1 Batch=25 Loss= 9.226006950484589e-05
Epoch=1 Batch=30 Loss= 0.00017270003701241876
Epoch=2 Batch=5 Loss= 0.00015975440328475088
Epoch=2 Batch=10 Loss= 5.999789536872413e-05
Epoch=2 Batch=15 Loss= 0.0001967525431609829
Epoch=2 Batch=20 Loss= 9.408122568856924e-05
Epoch=2 Batch=25 Loss= 0.00013969274432383826
Epoch=2 Batch=30 Loss= 0.00010921109333139611
Epoch=3 Batch=5 Loss= 7.752868077659514e-05
Epoch=3 Batch=10 Loss= 0.0002157480121240951
Epoch=3 Batch=15 Loss= 7.050490889923822e-05
Epoch=3 Batch=20 Loss= 0.00016003704404283782
Epoch=3 Batch=25 Loss= 9.497640567133204e-05
```

```
Epoch=3 Batch=30 Loss= 0.00018086953787133098
Epoch=4 Batch=5 Loss= 0.0001792347575246822
Epoch=4 Batch=10 Loss= 0.00010913591436292335
Epoch=4 Batch=15 Loss= 0.00012821899290429428
Epoch=4 Batch=20 Loss= 0.00012443312426739793
Epoch=4 Batch=25 Loss= 0.00010681348558136961
Epoch=4 Batch=30 Loss= 8.998089251690544e-05
Epoch=5 Batch=5 Loss= 0.00014320173559099204
Epoch=5 Batch=10 Loss= 9.809511111598112e-05
Epoch=5 Batch=15 Loss= 0.0001744861619954463
Epoch=5 Batch=20 Loss= 7.927484039100818e-05
Epoch=5 Batch=25 Loss= 0.00018849076732294635
Epoch=5 Batch=30 Loss= 4.151145094439812e-05
Epoch=6 Batch=5 Loss= 0.0001566025857755 4487
Epoch=6 Batch=10 Loss= 0.00019236720208937186
Epoch=6 Batch=15 Loss= 4.527691417024471e-05
Epoch=6 Batch=20 Loss= 9.417474993824725e-05
Epoch=6 Batch=25 Loss= 0.0002104485291056335
Epoch=6 Batch=30 Loss= 1.9266695380792954e-05
Epoch=7 Batch=5 Loss= 0.00015706719136687752
Epoch=7 Batch=10 Loss= 9.154806266451488e-05
Epoch=7 Batch=15 Loss= 9.19858780434879e-05
Epoch=7 Batch=20 Loss= 9.020557554322295e-05
Epoch=7 Batch=25 Loss= 0.00021219178088358604
Epoch=7 Batch=30 Loss= 6.667216500773065e-05
####Finished Training######
Accuracy of the network on test images: 96 %
0.9697885196374623
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Overall Performance

Model Layers    Test Accu for Best Parameters    Parameters(Epochs,LR)
Validation Accu    Running Time(seconds)
--------------  ------------------------------  ----------------------
----------------  ----------------------
2 layers                                         (1, 0.001)
0.8975               14.7693
2 layers                                         (1, 0.01)
0.5325               14.8554
2 layers                                         (3, 0.001)
0.865                44.4693
2 layers                                         (3, 0.01)
0.4825               43.9481
2 layers                                         (5, 0.001)
0.9275               73.6818
2 layers                                         (5, 0.01)
```

```
0.955                    73.4141
2 layers                                        (7, 0.001)
0.955                   102.658
2 layers          95.166                        (7, 0.01)
0.96                    102.585
3 layers                                        (1, 0.001)
0.8825                   17.7488
3 layers                                        (1, 0.01)
0.9225                   18.2211
3 layers                                        (3, 0.001)
0.9475                   53.3484
3 layers                                        (3, 0.01)
0.77                     53.5255
3 layers                                        (5, 0.001)
0.9625                   88.5956
3 layers                                        (5, 0.01)
0.9275                   88.6793
3 layers                                        (7, 0.001)
0.9725                  123.851
3 layers          96.677                        (7, 0.01)
0.975                   123.874
4 layers                                        (1, 0.001)
0.8025                   20.6156
4 layers                                        (1, 0.01)
0.4825                   20.407
4 layers                                        (3, 0.001)
0.8425                   60.817
4 layers                                        (3, 0.01)
0.975                    61.1468
4 layers                                        (5, 0.001)
0.9775                  102.019
4 layers                                        (5, 0.01)
0.98                    102.299
4 layers          96.979                        (7, 0.001)
0.985                   142.627
4 layers                                        (7, 0.01)
0.9825                  142.828
```

Conclusion: Based on our experiments with 3 different models we observe that with 4 layers model we got accuracy of 96% on test dataset with best parameters combination of number of epochos=7 and learning rate =0.001

References:

1. CS231n: Deep Learning for Computer Vision - Stanford - Spring 2022
2. PyTorch Tutorials
3. PyTorch Conv2D Explained with Examples

[ ]:
```
# print('Actual:{} Predicted:{}'.format(testData.dataset.
 ↪classes[labels[7]],testData.dataset.classes[predicted[7]]))
# plt.imshow(images[7].cpu().permute(1,2,0))
```

[ ]:
```
# for i in testDataLoader:
#   print('Images:{} --> Lables:{}'.format(len(i[0]),len(i[1])))
```