Project Part 3: Classification Using Neural Networks and Deep Learning

Objective:

The main objective of this project is learning and understanding the concept of convolutional neural network in an experiment with different parameters and settings.

Dataset:

The dataset is the same as that of Project 1 – MNIST dataset and can be easily brought into the workspace using libraries in Keras.

Tasks:

The major 3 tasks to be implemented are:

1. Run the baseline code.

2. Change the kernel size to 5 X 5 and redo the experiment again and plot the learning errors vs epoch and report the testing error and testing accuracy.

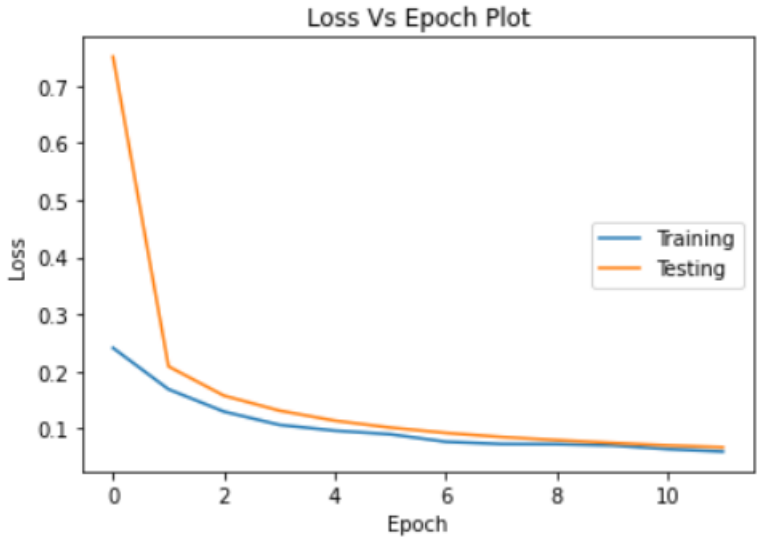
3. Change the number of feature maps in the first and second convolutional layers and repeat the experiment. Plot the learning errors vs epoch and report the testing error and testing accuracy.

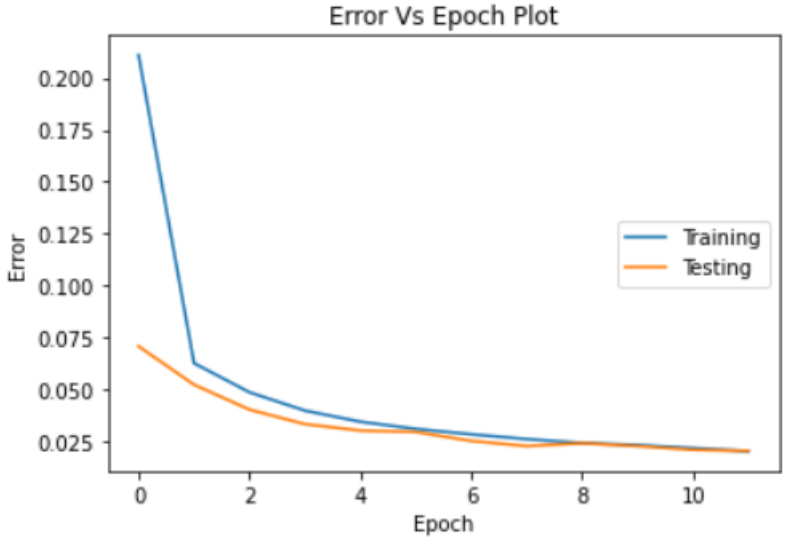
Results:

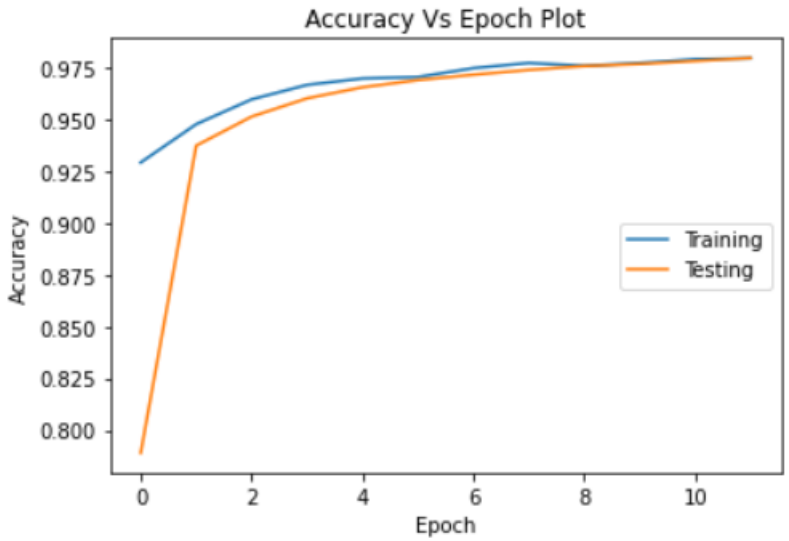
The plot of learning error vs epoch and accuracy are plotted at each stage of the experiment conducted.

Task 1:

Kernel size = 3 X 3. The test loss and accuracy of the experiment with the given (default) parameters are:


Loss Vs Epoch Plot
Loss
Epoch
Training
Testing
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8
10


Error Vs Epoch Plot
Error
Epoch
Training
Testing
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
2
4
6
8
10




```
Test Error: 0.020299971103668213
Test Loss: 0.05996604121867567
Test Accuracy: 0.9797000288963318
Train on 60000 samples, validate on 10000 samples
```

Task 2:

Change the kernel size to 5X5 and redo the experiment. The test loss and accuracy of the experiment with the changed parameters are:

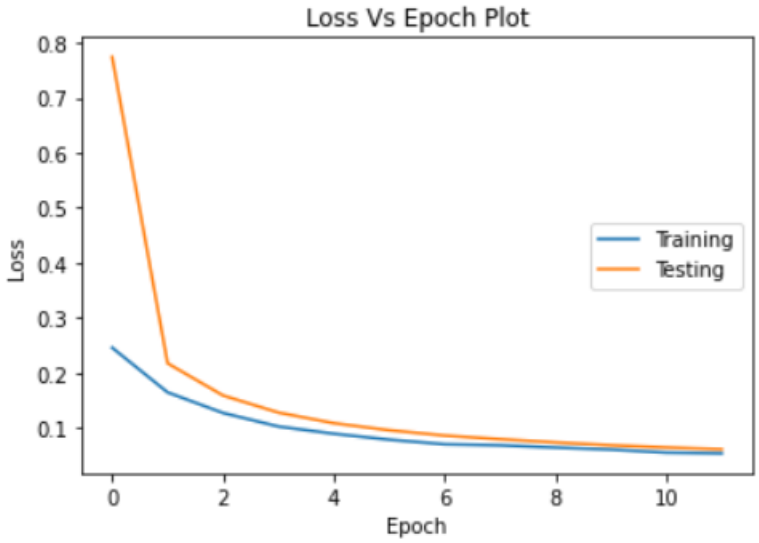
```
model1 = Sequential()
model1.add(Conv2D(6, kernel_size=(5, 5),
                  activation='relu',
                  input_shape=input_shape))
model1.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"))
model1.add(Conv2D(16, (5, 5), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"))
model1.add(Flatten())
model1.add(Dense(120, activation='relu'))
model1.add(Dense(84, activation='relu'))

model1.add(Dense(num_classes, activation='softmax'))
```

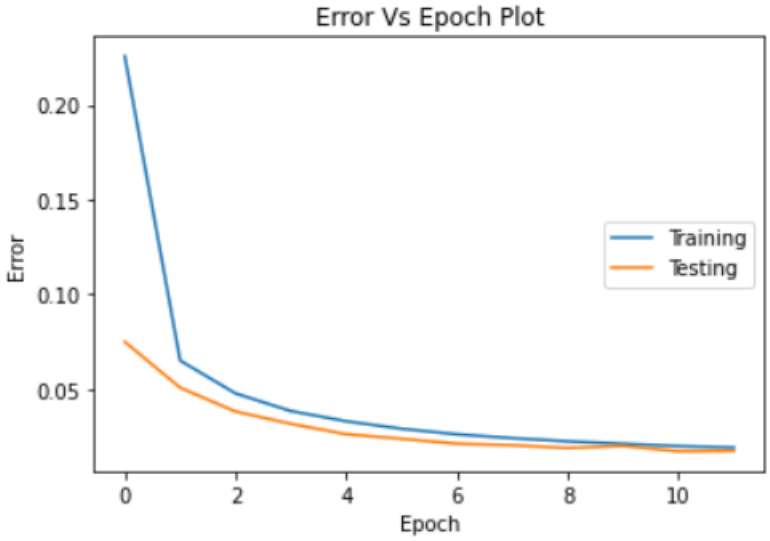
Because the Project Overview said: The pooling is 2x2 with stride 1

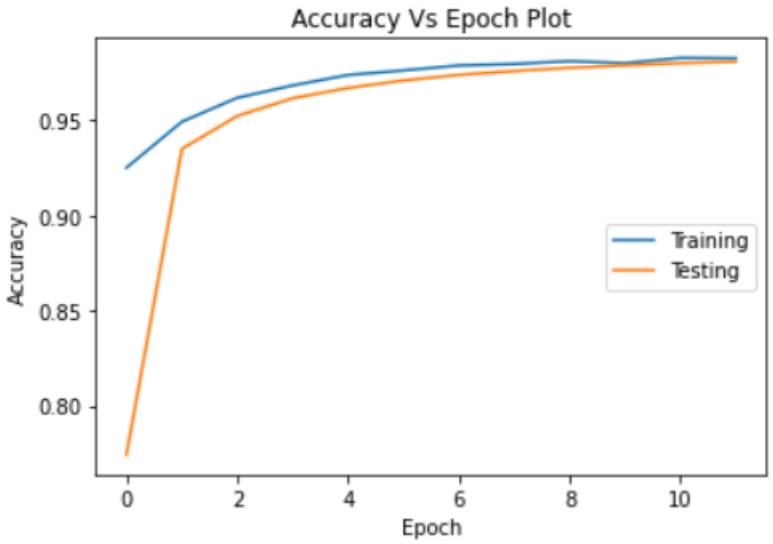
So, I add the strides= (1,1) and padding=" same", I made a comparison between adding and not adding, and finally the accuracy of adding strides is higher.

Result without strides:

```
#Change the kernel size to 5*5, redo the experiment, p
model1 = Sequential()
model1.add(Conv2D(6, kernel_size=(5, 5),
                  activation='relu',
                  input_shape=input_shape))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Conv2D(16, (5, 5), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Flatten())
model1.add(Dense(120, activation='relu'))
model1.add(Dense(84, activation='relu'))
```





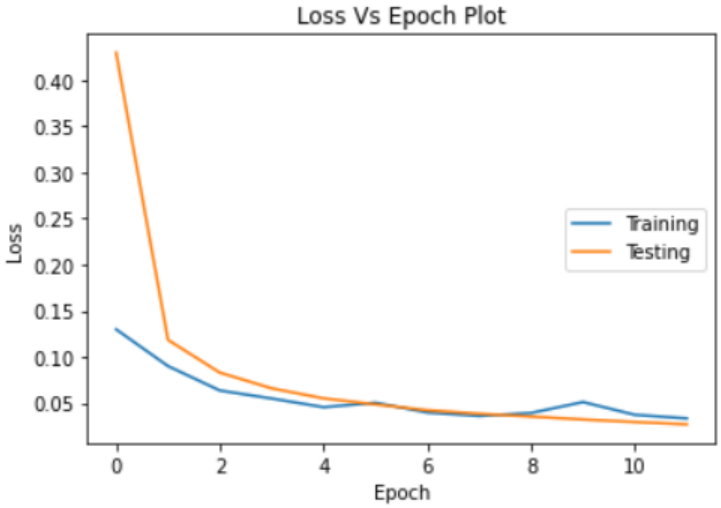


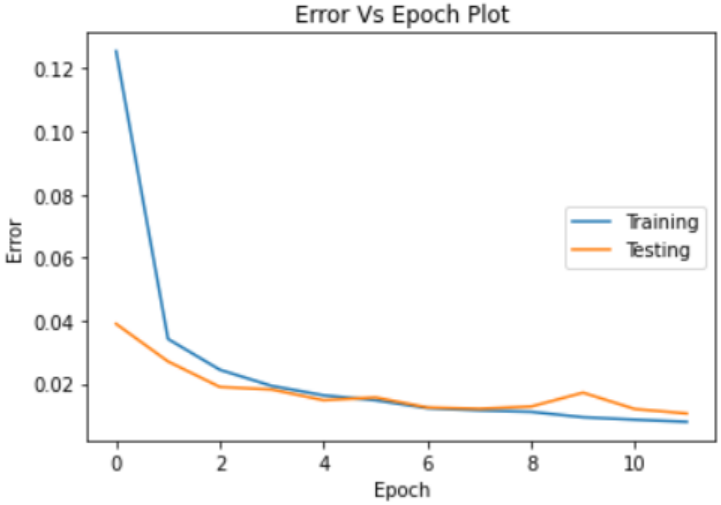

```
Test Error: 0.017199993133544922
Test Loss: 0.053491598547017204
Test Accuracy: 0.9828000068664551
Train on 60000 samples, validate on 10000 samples
```

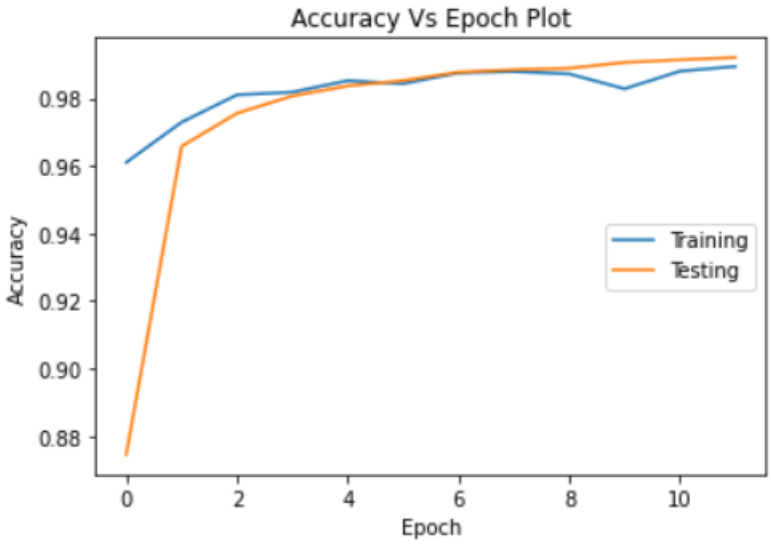
The accuracy is 98.2%,when the kernel size increased.

With strides:

```
model1 = Sequential()
model1.add(Conv2D(6, kernel_size=(5, 5),
                  activation='relu',
                  input_shape=input_shape))
model1.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"))
model1.add(Conv2D(16, (5, 5), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"))
model1.add(Flatten())
model1.add(Dense(120, activation='relu'))
model1.add(Dense(84, activation='relu'))

model1.add(Dense(num_classes, activation='softmax'))
```


Loss Vs Epoch Plot
Loss
Epoch
Training
Testing
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
2
4
6
8
10


Error Vs Epoch Plot
Error
Epoch
Training
Testing
0.12
0.10
0.08
0.06
0.04
0.02
0
2
4
6
8
10




```
Test Error: 0.010599970817565918
Test Loss: 0.033465440757456236
Test Accuracy: 0.9894000291824341
Train on 60000 samples, validate on 10000 samples
```

The accuracy seems to increase to 98.9% when the kernel size is increased and add strides.

Task 3:

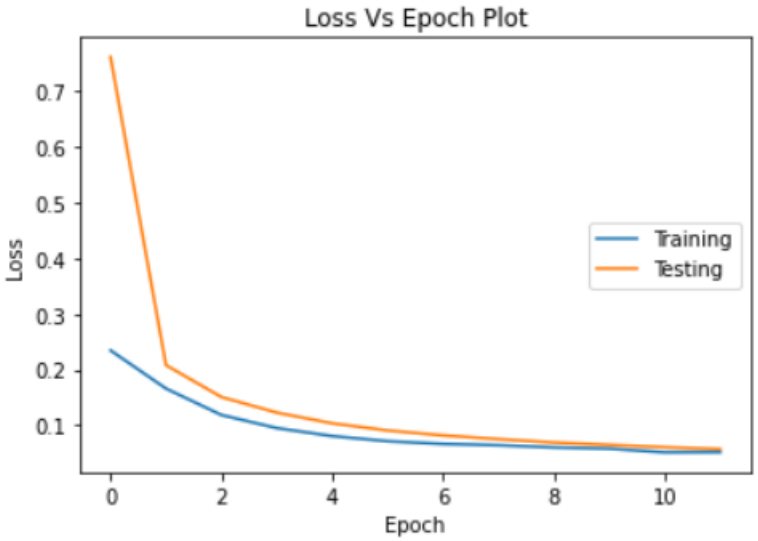
Change the number of feature maps in the first and second convolutional layers and redo the experiment. [# feature maps for 1st layer – 8, 2nd layer – 32]

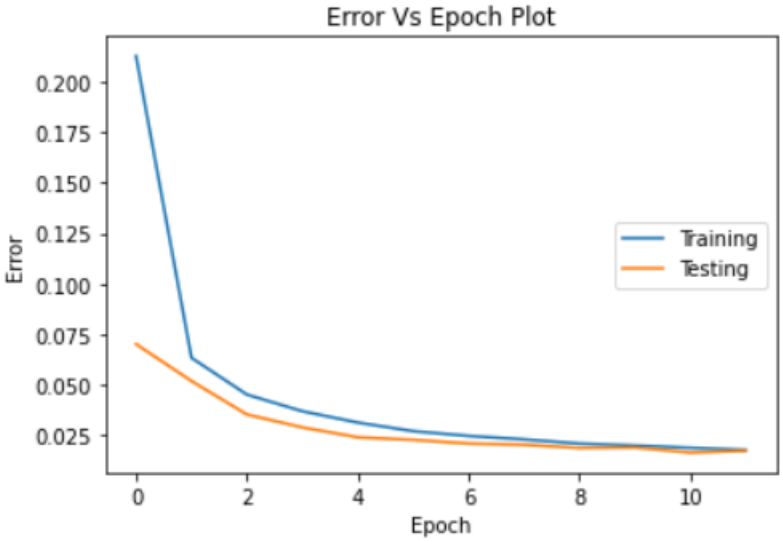
```
#Question 3:
#CNN with kernel-size as 3*3 and feature maps as 8,32 for 2 convolution layers respectively
model2 = Sequential()
model2.add(Conv2D(8, kernel_size=(3, 3),
                  activation='relu',
                  input_shape=input_shape))
model2.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1), padding="same"))
model2.add(Conv2D(32, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1), padding="same"))
model2.add(Flatten())
model2.add(Dense(120, activation='relu'))
model2.add(Dense(84, activation='relu'))
```

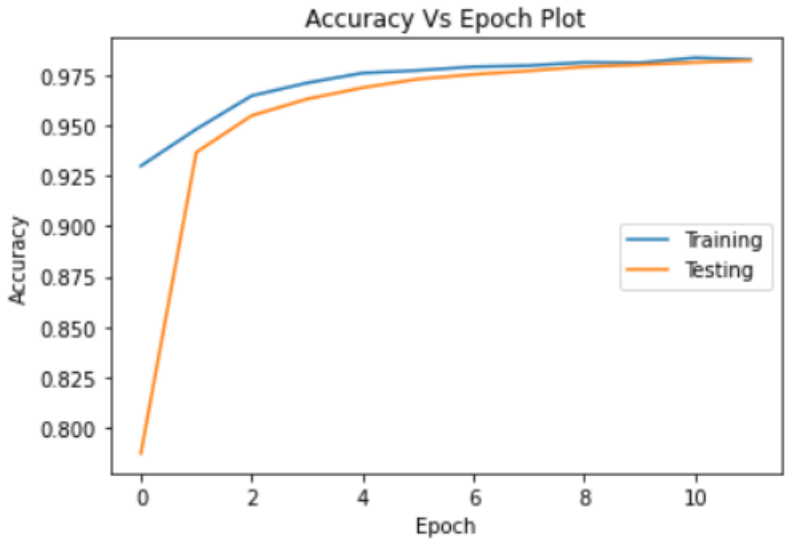
I also add strides to increase the accuracy and do the compare.

Result without strides:

```
model2 = Sequential()
model2.add(Conv2D(8, kernel_size=(3, 3),
                  activation='relu',
                  input_shape=input_shape))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Conv2D(32, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Flatten())
model2.add(Dense(120, activation='relu'))
model2.add(Dense(84, activation='relu'))
```


Loss Vs Epoch Plot
Loss
Epoch
Training
Testing
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8
10


Error Vs Epoch Plot
Error
Epoch
Training
Testing
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
2
4
6
8
10

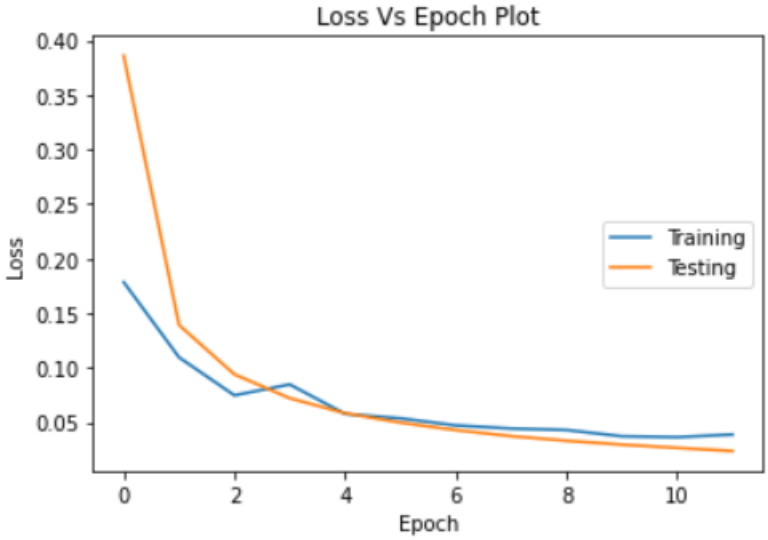


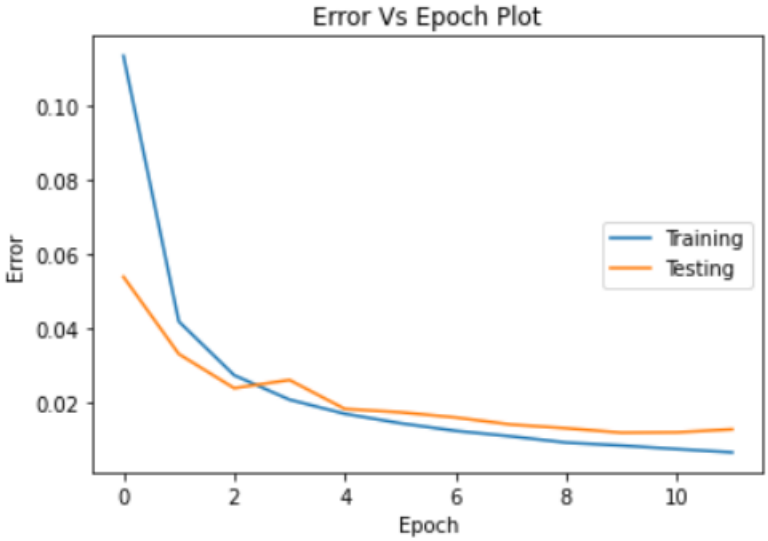

```
Test Error: 0.017099976539611816
Test Loss: 0.051556601767987015
Test Accuracy: 0.9829000234603882
```

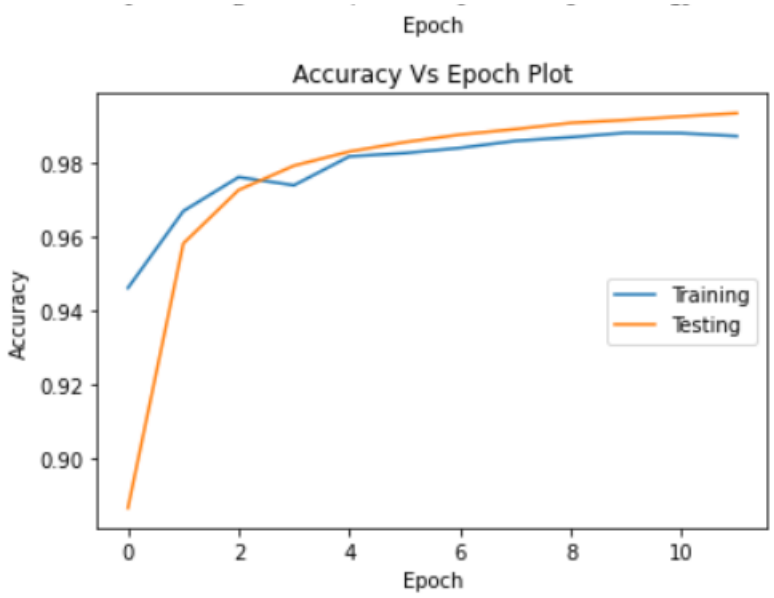
The accuracy seems to increase to 98.3% when the feature maps are increased in the first and second layers.

Result with strides:

```
#Question 3:
#CNN with kernel-size as 3*3 and feature maps as 8,32 for 2 convolution layers respectively
model2 = Sequential()
model2.add(Conv2D(8, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model2.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1), padding="same"))
model2.add(Conv2D(32, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1), padding="same"))
model2.add(Flatten())
model2.add(Dense(120, activation='relu'))
model2.add(Dense(84, activation='relu'))
```


Loss Vs Epoch Plot
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Loss
0
2
4
6
8
10
Epoch
Training
Testing


Error Vs Epoch Plot
0.10
0.08
0.06
0.04
0.02
Error
0
2
4
6
8
10
Epoch
Training
Testing




```
Test Error: 0.013000011444091797
Test Loss: 0.03893815435371362
Test Accuracy: 0.9869999885559082
```

The accuracy seems to increase to 98.7% when the feature maps are increased in the first and second layers and add strides.

Conclusion:

The overall observation is that the model performs better when increasing the kernel size and the number of feature maps and is evident in the above-mentioned obtained test accuracies.

And add strides also can increase the accuracy obviously.