Project Part 3 by Pravin12

Implementation 1

Results 2

Baseline 2

Experiment with kernel size: 5x5 and feature maps : 6 for 1st conv2D layer and 16 for 2nd con2D layer 3

Experiment with kernel size: 5x5 and feature maps : 20 for 1st conv2D layer and 50 for 2nd con2D layer 4

Analyzing Results 5

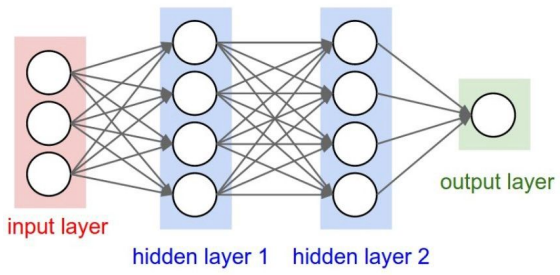
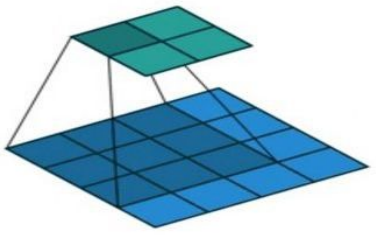
Code 6

Implementation

In this part of the project, we are using deep learning to classify images by building convolutional neural network(CNN). We are using Keras python library here. After reading the data from MNIST dataset, model of sequential type is built. We have added several layers as mentioned in the problem. Among all the layers, there are 2 conv2D layers. After adding the layers, model is compiled passing 3 parameters: loss, optimizer and metrics. we will use the 'accuracy' metric to see the accuracy score on the validation set when we train the model. Then the model is trained with train dataset using fit() method. At the end, the model is evaluated with test data and found the test accuracy and loss. At the end , model accuracy and loss is plotted for both train and test dataset.

The above algorithm is run in 3 different flavours:

1. Baseline : as it is given in the project i.e with kernel size: 3x3 and feature maps : 6 for 1st conv2D layer and 16 for 2nd con2D layer
2. Experiment with kernel size: 5x5 and feature maps : 6 for 1st conv2D layer and 16 for 2nd con2D layer
3. Experiment with kernel size: 5x5 and feature maps : 20 for 1st conv2D layer and 50 for 2nd con2D layer

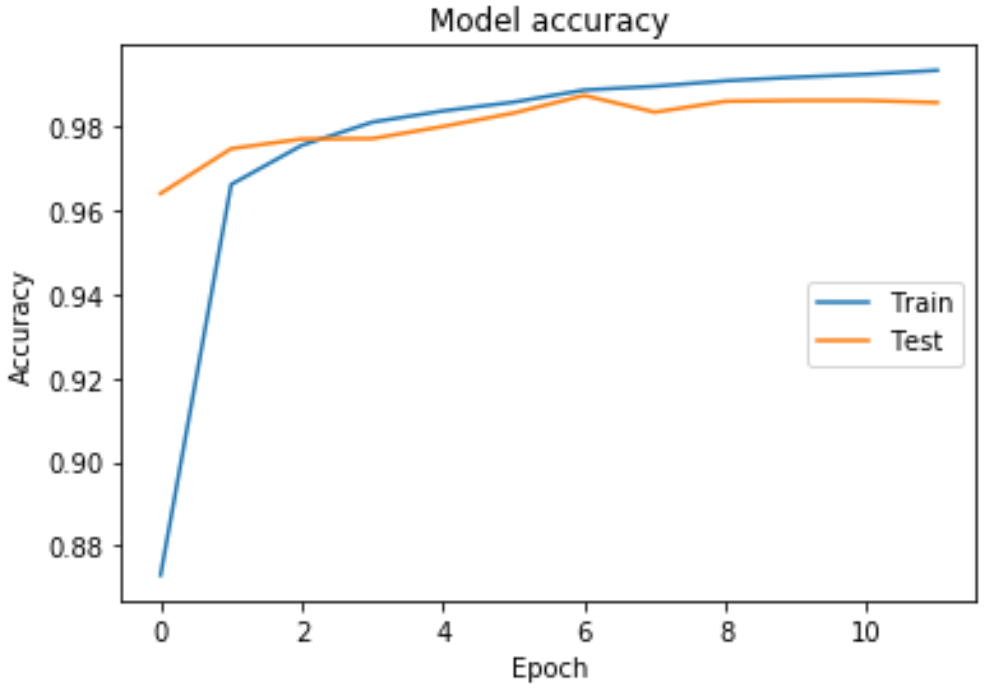
A simple neural network	**Convolutional Neural Network**
	

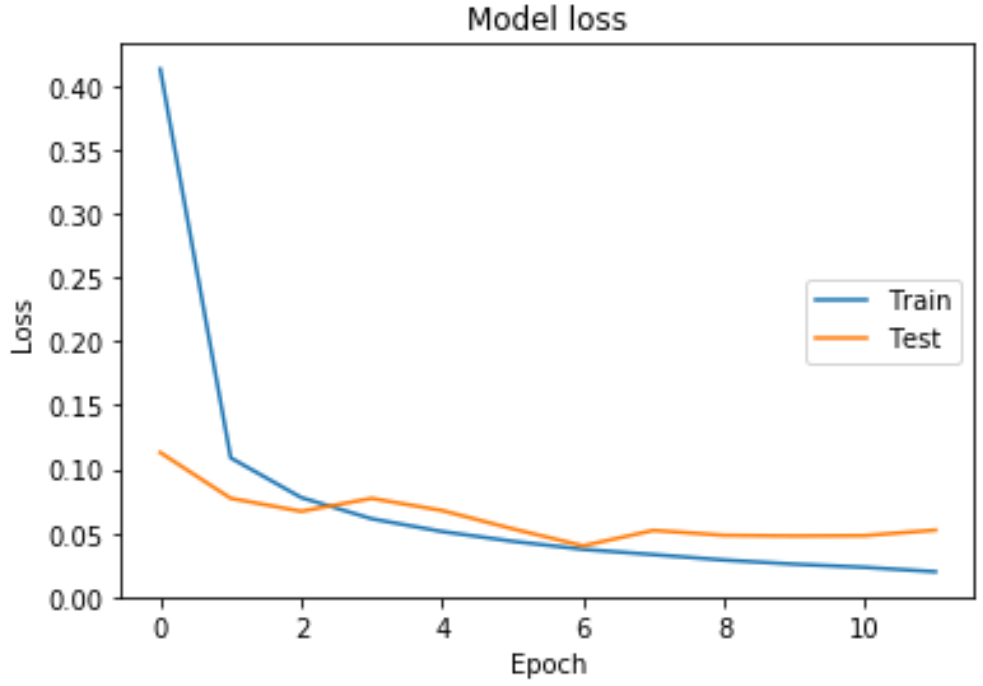
Results

Below are the results for all the 3 different flavours.

1. Baseline

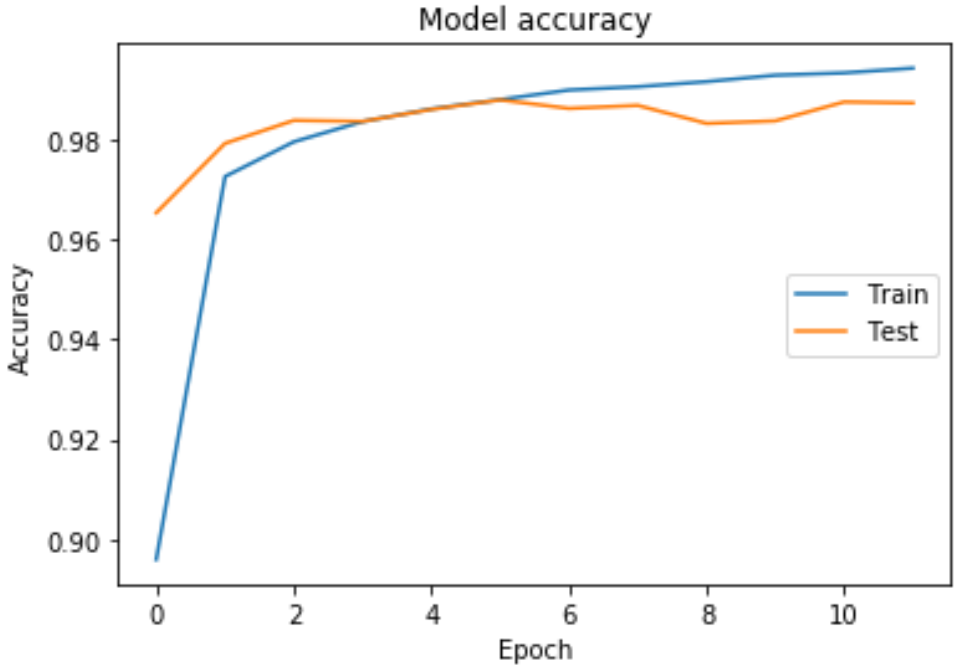
Test loss: 0.05222264671537505
Test accuracy: 0.9857

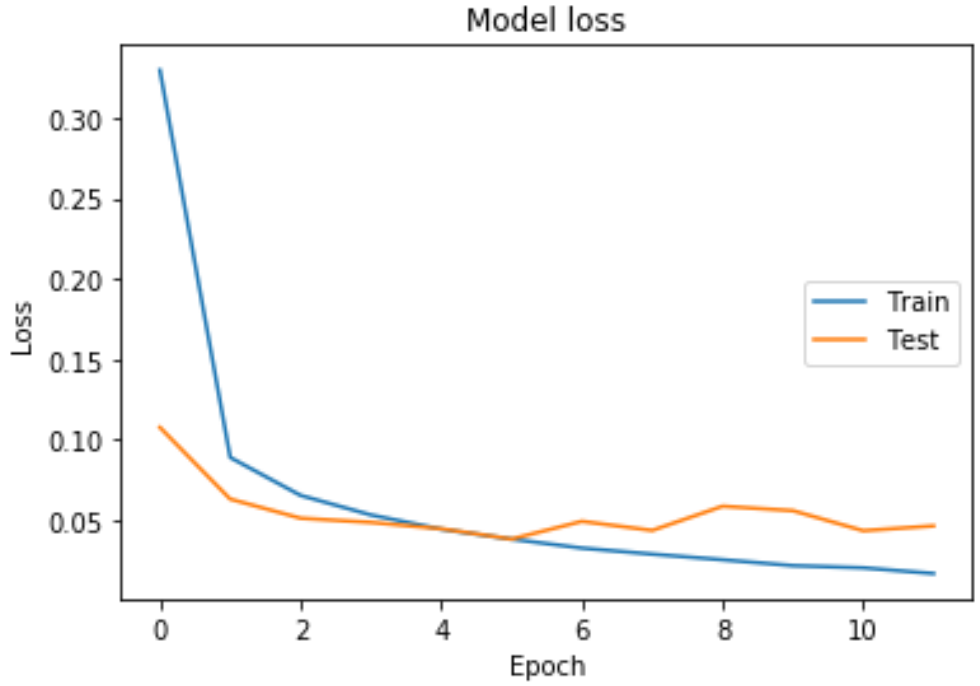




2. **Experiment with kernel size: 5x5 and feature maps : 6 for 1st conv2D layer and 16 for 2nd con2D layer**

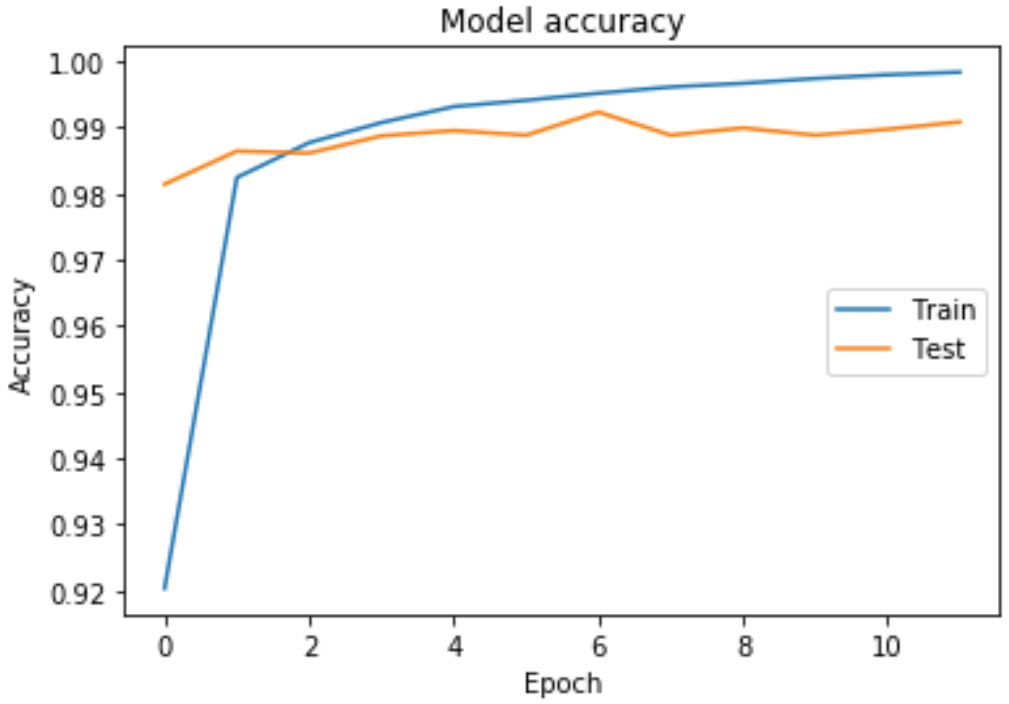
Test loss: 0.04650563598516392
Test accuracy: 0.9874

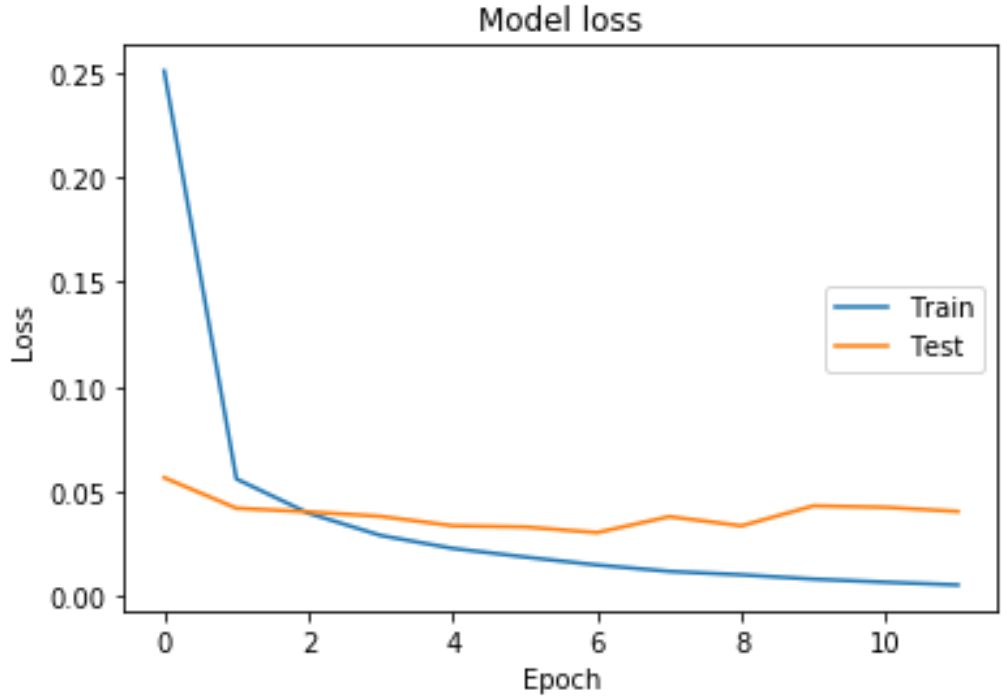




3. **Experiment with kernel size: 5x5 and feature maps : 20 for 1st conv2D layer and 50 for 2nd con2D layer**

Test loss: 0.040642606166668656
Test accuracy: 0.9908





Analyzing Results

1. As the kernel size increased from 3x3 to 5x5 keeping the same feature maps , the test accuracy has increased from 0.9857 to 0.9874 , while the test loss is decreased from 0.0522 to 0.0465.

2. As the feature maps is increased from 6 to 20 for 1st layer and 16 to 50 for 2nd layer, the test accuracy has increased from 0.9874 to 0.9908, while the test loss is decreased from 0.0465 to 0.0406.

3. As we increase the number of epochs, the time to fit and learn the model takes more time, but the train accuracy is increasing and the train loss is decreasing.

4. Where as for test data, as the epochs are increasing, we can say accuracy is increased in general but it is not a clear pattern. For example, test accuracy might not increase when epoch is increased from 5 to 6, but it is increased when epoch is changed from 1 to 12. Note: Here the number of epochs is the number of times the model will cycle through the data.

Code

```
import keras
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
#### Code structure #####
# 1. Method main has the baseline code. This method is parameterized with kernel size and feature maps.
#      This way i can call the same method with different parameters as asked in the question.
# 2. Method for plotting the graphs.
# 3. Call the above methods with appropriate parameters as per the requirement.
######## End #############

#  Returns model after learning from train dataset. And also reports the test loss and accuracy.
#  Parameters: 1. feature_map - An array of number of feature maps for each layer
#              2. kernel_zie - An array of kernel size for all dimention for example [3, 3] for 3x3
#              3. dispStr: Display string when reporting the test loss and accurace.
def main(feature_map, kernel_size, dispStr):
batch_size = 128
num_classes = 10
epochs = 12
# input image dimensions
img_rows, img_cols = 28, 28
# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
```

```
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
model = Sequential()
print ("kernel_sizes =  ", kernel_size)
print ("feature_map = ", feature_map)
model.add(Conv2D(feature_map[0], kernel_size=(kernel_size[0], kernel_size[1]),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(feature_map[1], (kernel_size[0], kernel_size[1]), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
# https://keras.io/optimizers/
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=None, decay=0.0),
              metrics=['accuracy'])
learn = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print ("\n", dispStr)
print ('      Test loss:', score[0])
print ('      Test accuracy:', score[1])
print ("################################################\n")

return learn
# Plots the Model accuracy and loss  of both train and test dataset.
# Parameters: 1. learn - model which is created from train dataset.
def plot(learn):
# Plot training & validation accuracy values
plt.plot(learn.history['acc'])
plt.plot(learn.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='center right')
plt.show()
plt.plot(learn.history['loss'])
plt.plot(learn.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='center right')
plt.show()
#main code, calling methods
print("###### BASELINE ########")
learn=main(feature_map = [6,16], kernel_size=[3,3], dispStr="######## BASELINE ERROR AND
ACCURACY ######")
plot(learn)
print("###### EXPERIMENT WITH kernel size to 5*5 #######")
```

```
learn=main(feature_map = [6,16], kernel_size=[5,5],dispStr="######## ERROR AND ACCURACY OF
EXPERIMENT WITH kernel size to 5*5 ######")
plot(learn)
print("###### EXPERIMENT WITH kernel size to 5*5 AND MODIFIED FEATURE MAPS #######")
learn=main(feature_map = [20,50],kernel_size=[5,5],dispStr="######## ERROR AND ACCURACY OF
EXPERIMENT WITH kernel size to 5*5 AND MODIFIED FEATURE MAPS######")
plot(learn)
```