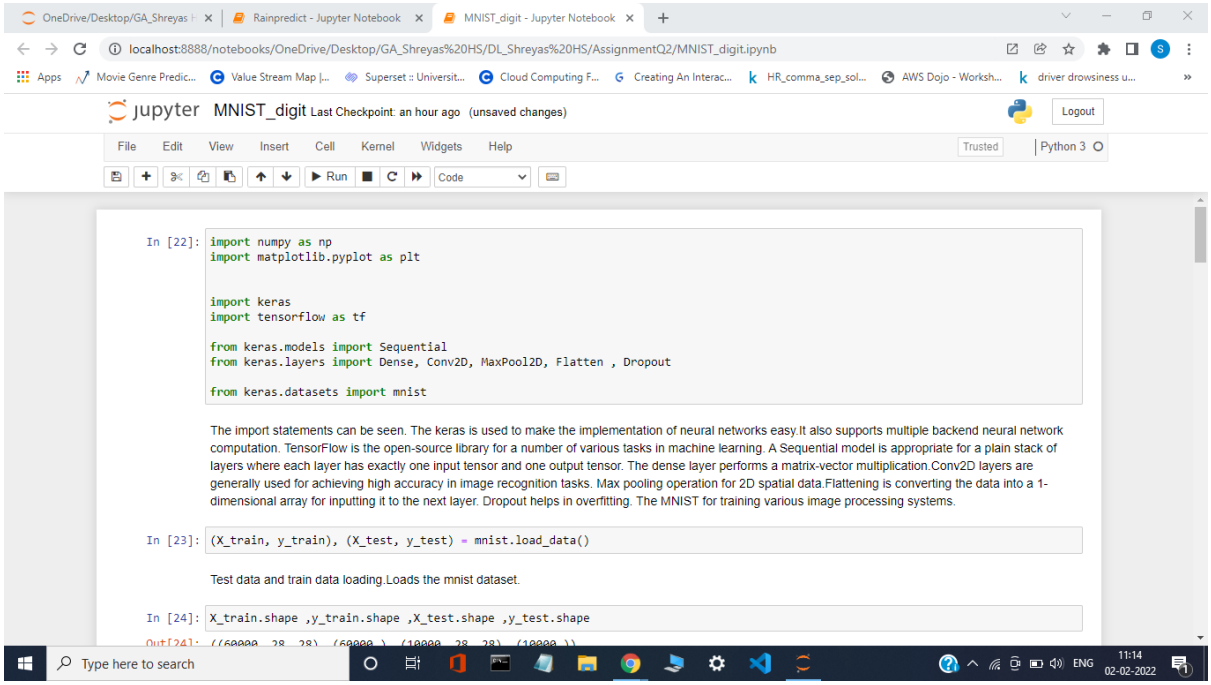
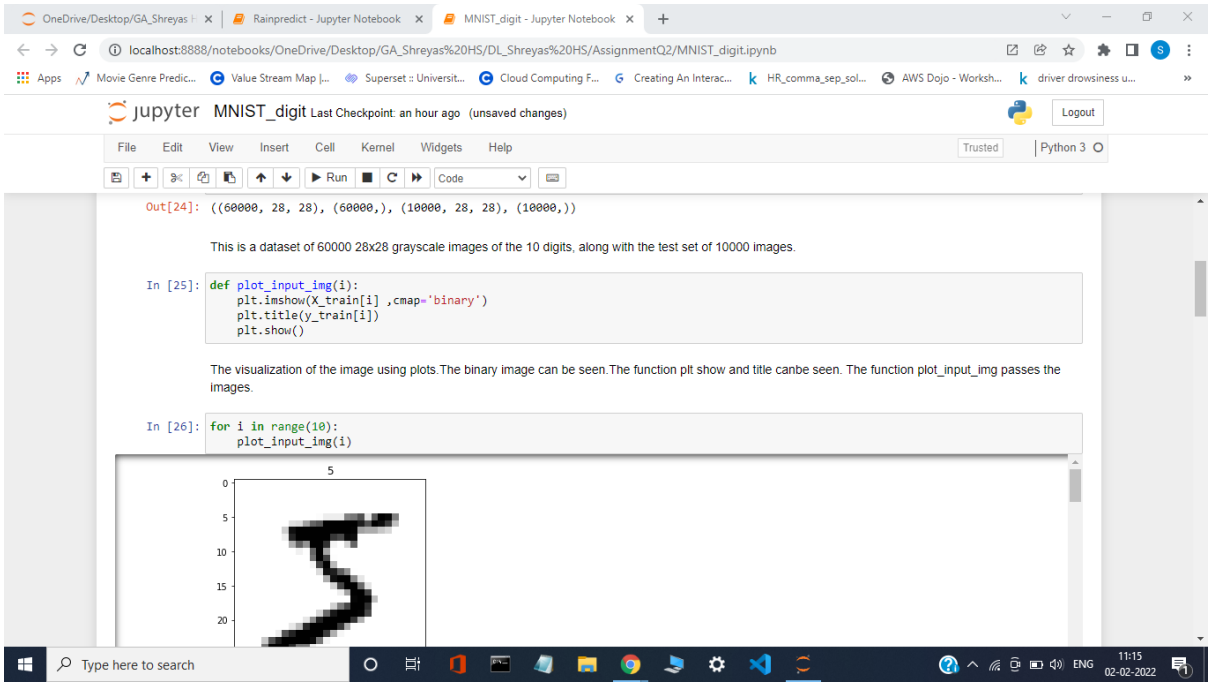
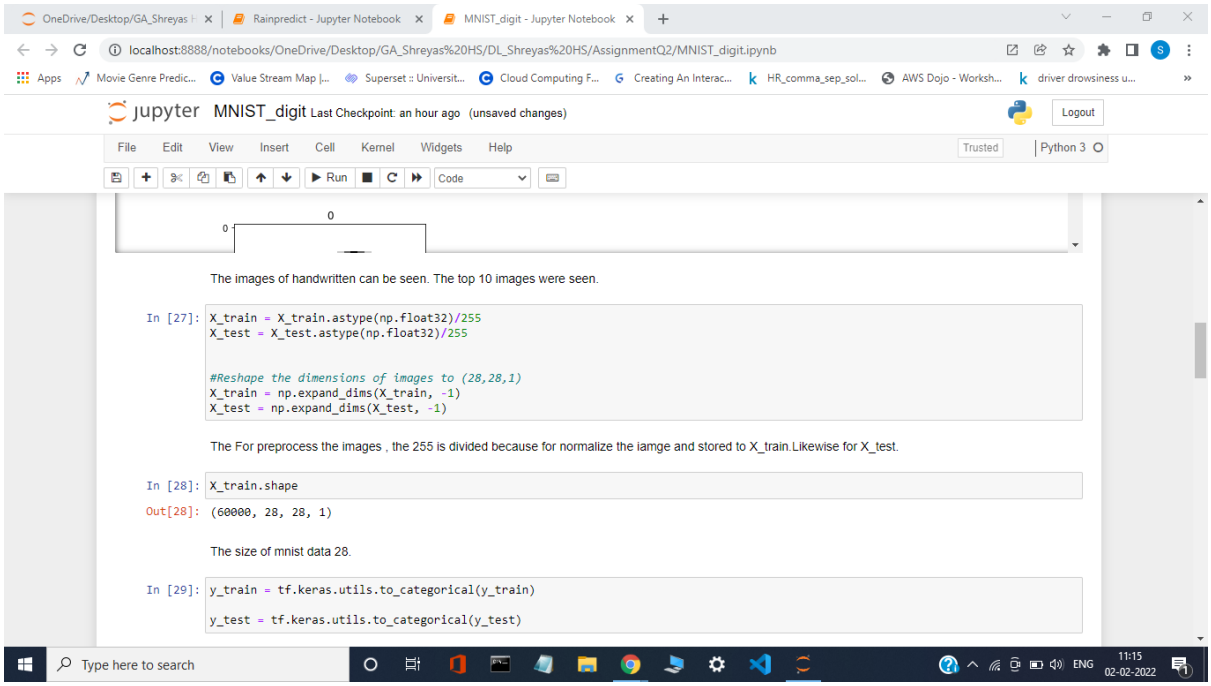
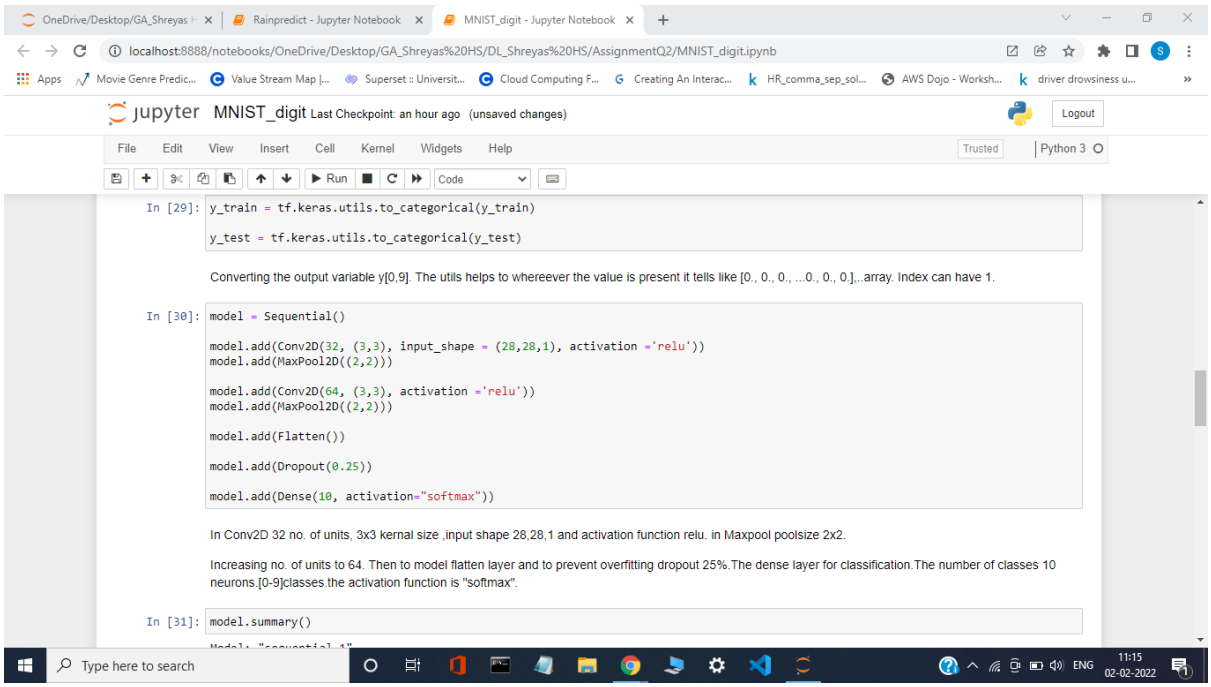
Q2. Train a deep learning classification model for classes in **MNIST** *Handwriting* dataset.

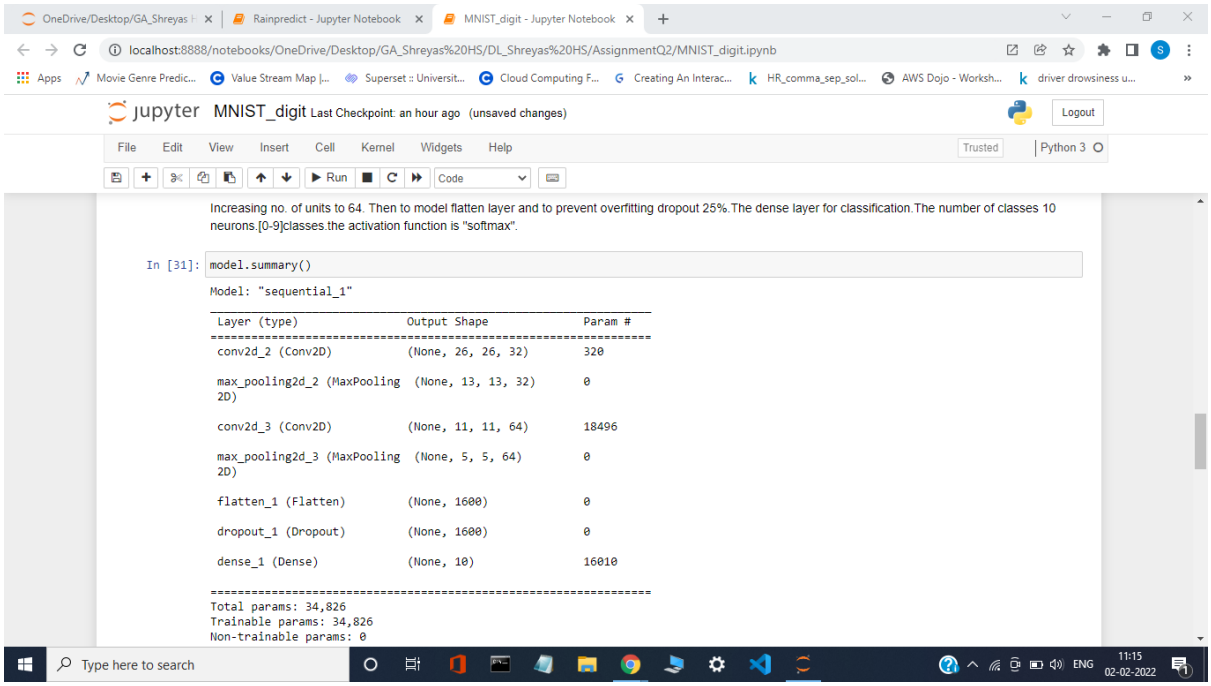
Screenshots:

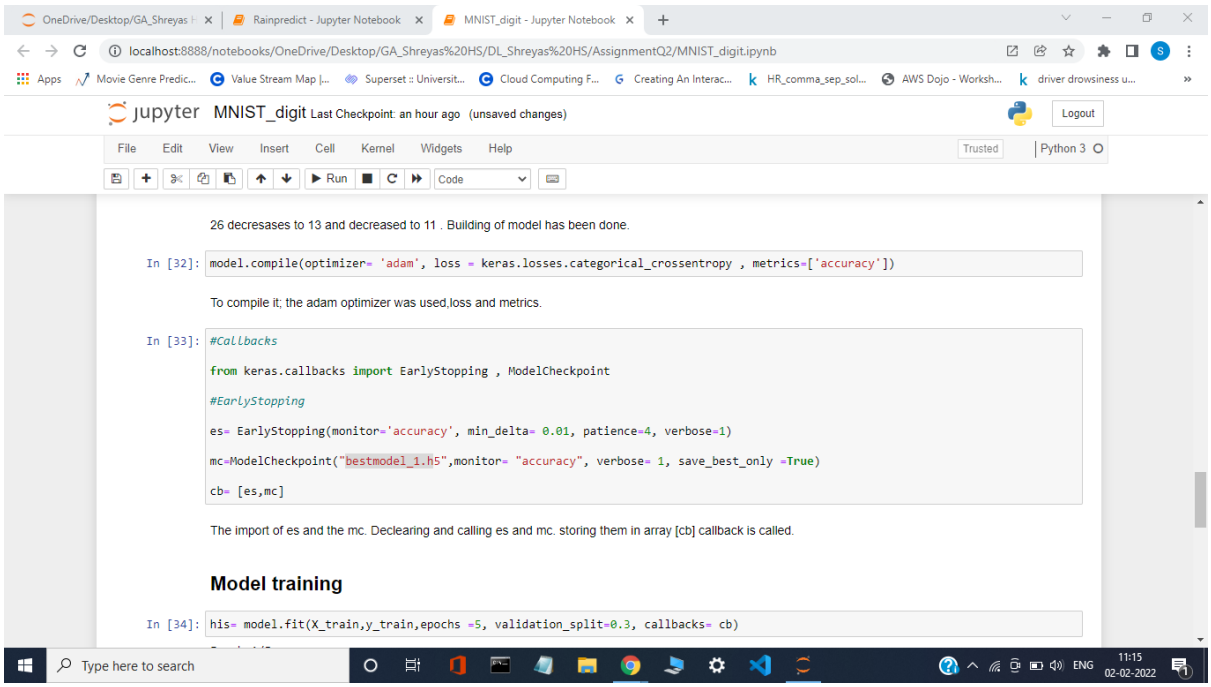


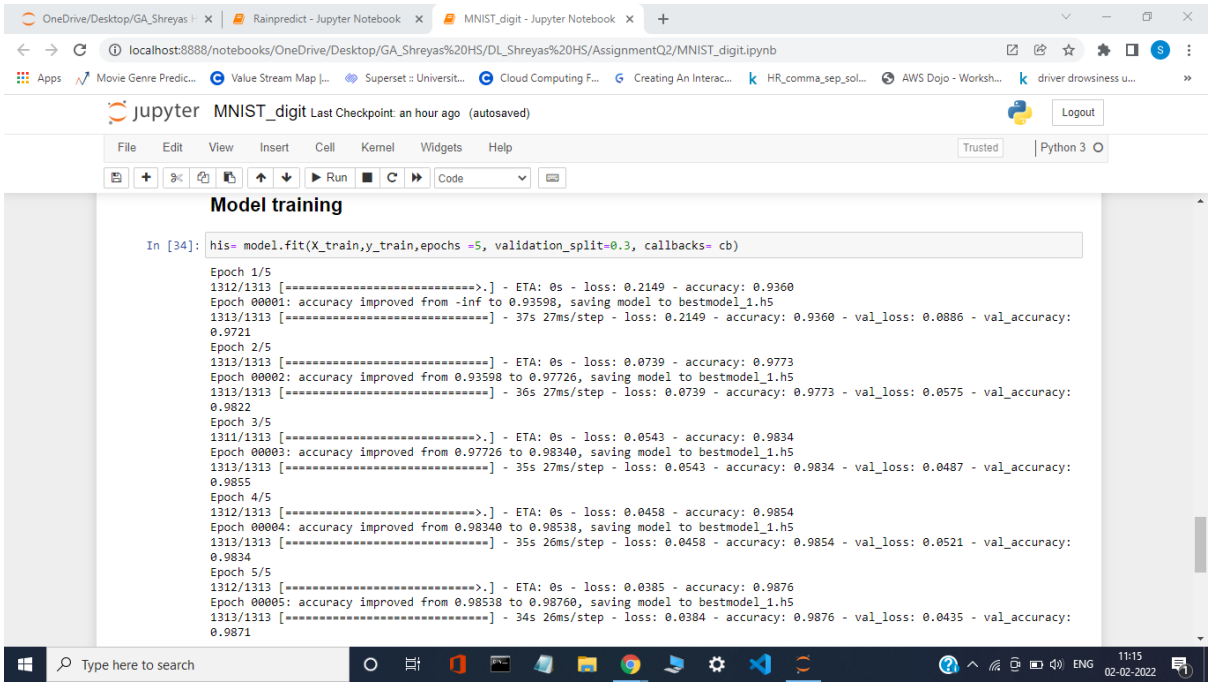


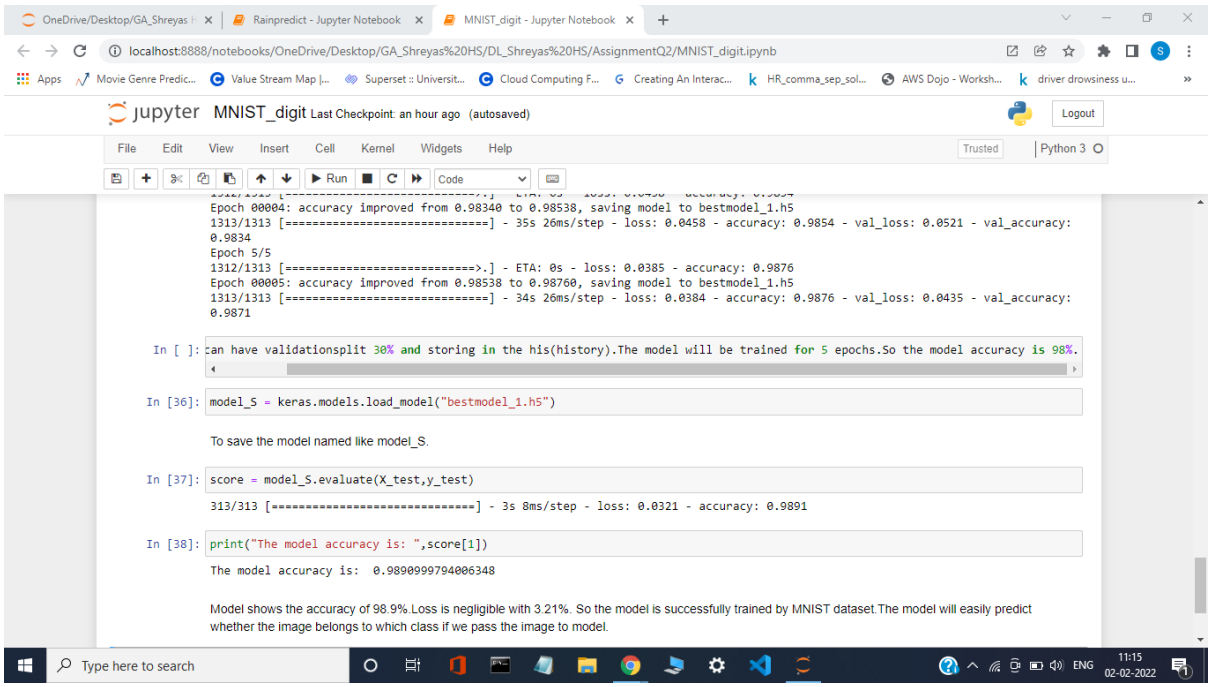

OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/DL_Shreyas%20HS/AssignmentQ2/MNIST_digit.ipynb
jupyter MNIST_digit Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
0
0
The images of handwritten can be seen. The top 10 images were seen.
In [27]:
X_train = X_train.astype(np.float32)/255
X_test = X_test.astype(np.float32)/255
#Reshape the dimensions of images to (28,28,1)
X_train = np.expand_dims(X_train, -1)
X_test = np.expand_dims(X_test, -1)
The For preprocess the images , the 255 is divided because for normalize the iamge and stored to X_train.Likewise for X_test.
In [28]:
X_train.shape
Out[28]:
(60000, 28, 28, 1)
The size of mnist data 28.
In [29]:
y_train = tf.keras.utils.to_categorical(y_train)
y_test = tf.keras.utils.to_categorical(y_test)
Type here to search
ENG
11:15
02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/DL_Shreyas%20HS/AssignmentQ2/MNIST_digit.ipynb
jupyter MNIST_digit Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [29]:
y_train = tf.keras.utils.to_categorical(y_train)
y_test = tf.keras.utils.to_categorical(y_test)
Converting the output variable y[0,9]. The utils helps to whereever the value is present it tells like [0., 0., 0., ...0., 0., 0.]...array. Index can have 1.
In [30]:
model = Sequential()
model.add(Conv2D(32, (3,3), input_shape = (28,28,1), activation ='relu'))
model.add(MaxPool2D((2,2)))
model.add(Conv2D(64, (3,3), activation ='relu'))
model.add(MaxPool2D((2,2)))
model.add(Flatten())
model.add(Dropout(0.25))
model.add(Dense(10, activation="softmax"))
In Conv2D 32 no. of units, 3x3 kernal size ,input shape 28,28,1 and activation function relu. in Maxpool poolsize 2x2.
Increasing no. of units to 64. Then to model flatten layer and to prevent overfitting dropout 25%.The dense layer for classification.The number of classes 10 neurons.[0-9]classes.the activation function is "softmax".
In [31]:
model.summary()
Type here to search
ENG
11:15
02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/DL_Shreyas%20HS/AssignmentQ2/MNIST_digit.ipynb
jupyter MNIST_digit Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Increasing no. of units to 64. Then to model flatten layer and to prevent overfitting dropout 25%.The dense layer for classification.The number of classes 10 neurons.[0-9]classes.the activation function is "softmax".
In [31]:
model.summary()
Model: "sequential_1"
Layer (type) Output Shape Param #
conv2d_2 (Conv2D) (None, 26, 26, 32) 320
max_pooling2d_2 (MaxPooling 2D) (None, 13, 13, 32) 0
conv2d_3 (Conv2D) (None, 11, 11, 64) 18496
max_pooling2d_3 (MaxPooling 2D) (None, 5, 5, 64) 0
flatten_1 (Flatten) (None, 1600) 0
dropout_1 (Dropout) (None, 1600) 0
dense_1 (Dense) (None, 10) 16010
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
Type here to search
ENG
11:15
02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/DL_Shreyas%20HS/AssignmentQ2/MNIST_digit.ipynb
jupyter MNIST_digit Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
26 decresases to 13 and decreased to 11 . Building of model has been done.
In [32]:
model.compile(optimizer= 'adam', loss = keras.losses.categorical_crossentropy , metrics=['accuracy'])
To compile it; the adam optimizer was used,loss and metrics.
In [33]:
#Callbacks
from keras.callbacks import EarlyStopping , ModelCheckpoint
#EarlyStopping
es= EarlyStopping(monitor='accuracy', min_delta= 0.01, patience=4, verbose=1)
mc=ModelCheckpoint("bestmodel_1.h5",monitor= "accuracy", verbose= 1, save_best_only =True)
cb= [es,mc]
The import of es and the mc. Declearing and calling es and mc. storing them in array [cb] callback is called.
Model training
In [34]:
his= model.fit(X_train,y_train,epochs =5, validation_split=0.3, callbacks= cb)
Type here to search
ENG
11:15
02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/DL_Shreyas%20HS/AssignmentQ2/MNIST_digit.ipynb
jupyter MNIST_digit Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Model training
In [34]:
his= model.fit(X_train,y_train,epochs =5, validation_split=0.3, callbacks= cb)
Epoch 1/5
1312/1313 [==============================>.] - ETA: 0s - loss: 0.2149 - accuracy: 0.9360
Epoch 00001: accuracy improved from -inf to 0.93598, saving model to bestmodel_1.h5
1313/1313 [==============================] - 37s 27ms/step - loss: 0.2149 - accuracy: 0.9360 - val_loss: 0.0886 - val_accuracy: 0.9721
Epoch 2/5
1313/1313 [==============================] - ETA: 0s - loss: 0.0739 - accuracy: 0.9773
Epoch 00002: accuracy improved from 0.93598 to 0.97726, saving model to bestmodel_1.h5
1313/1313 [==============================] - 36s 27ms/step - loss: 0.0739 - accuracy: 0.9773 - val_loss: 0.0575 - val_accuracy: 0.9822
Epoch 3/5
1311/1313 [==============================>.] - ETA: 0s - loss: 0.0543 - accuracy: 0.9834
Epoch 00003: accuracy improved from 0.97726 to 0.98340, saving model to bestmodel_1.h5
1313/1313 [==============================] - 35s 27ms/step - loss: 0.0543 - accuracy: 0.9834 - val_loss: 0.0487 - val_accuracy: 0.9855
Epoch 4/5
1312/1313 [==============================>.] - ETA: 0s - loss: 0.0458 - accuracy: 0.9854
Epoch 00004: accuracy improved from 0.98340 to 0.98538, saving model to bestmodel_1.h5
1313/1313 [==============================] - 35s 26ms/step - loss: 0.0458 - accuracy: 0.9854 - val_loss: 0.0521 - val_accuracy: 0.9834
Epoch 5/5
1312/1313 [==============================>.] - ETA: 0s - loss: 0.0385 - accuracy: 0.9876
Epoch 00005: accuracy improved from 0.98538 to 0.98760, saving model to bestmodel_1.h5
1313/1313 [==============================] - 34s 26ms/step - loss: 0.0384 - accuracy: 0.9876 - val_loss: 0.0435 - val_accuracy: 0.9871
Type here to search
ENG
11:15
02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/DL_Shreyas%20HS/AssignmentQ2/MNIST_digit.ipynb
jupyter MNIST_digit Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Epoch 00004: accuracy improved from 0.98340 to 0.98538, saving model to bestmodel_1.h5
1313/1313 [==============================] - 35s 26ms/step - loss: 0.0458 - accuracy: 0.9854 - val_loss: 0.0521 - val_accuracy: 0.9834
Epoch 5/5
1312/1313 [==============================>.] - ETA: 0s - loss: 0.0385 - accuracy: 0.9876
Epoch 00005: accuracy improved from 0.98538 to 0.98760, saving model to bestmodel_1.h5
1313/1313 [==============================] - 34s 26ms/step - loss: 0.0384 - accuracy: 0.9876 - val_loss: 0.0435 - val_accuracy: 0.9871
In [ ]:
can have validationsplit 30% and storing in the his(history).The model will be trained for 5 epochs.So the model accuracy is 98%.
In [36]:
model_S = keras.models.load_model("bestmodel_1.h5")
To save the model named like model_S.
In [37]:
score = model_S.evaluate(X_test,y_test)
313/313 [==============================] - 3s 8ms/step - loss: 0.0321 - accuracy: 0.9891
In [38]:
print("The model accuracy is: ",score[1])
The model accuracy is: 0.9890999794006348
Model shows the accuracy of 98.9%.Loss is negligible with 3.21%. So the model is successfully trained by MNIST dataset.The model will easily predict whether the image belongs to which class if we pass the image to model.
Type here to search
ENG
11:15
02-02-2022

Code: and **Description:**

```
import numpy as np

import matplotlib.pyplot as plt


import keras

import tensorflow as tf


from keras.models import Sequential

from keras.layers import Dense, Conv2D, MaxPool2D, Flatten , Dropout


from keras.datasets import mnist
```

The import statements can be seen.

The keras is used to make the implementation of neural networks easy.It also supports multiple backend neural network computation.

TensorFlow is the open-source library for a number of various tasks in machine learning.

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

The dense layer performs a matrix-vector multiplication.Conv2D layers are generally used for achieving high accuracy in image recognition tasks.

Max pooling operation for 2D spatial data.Flattening is converting the data into a 1-dimensional array for inputting it to the next layer.

Dropout helps in overfitting.

The MNIST for training various image processing systems.

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Test data and train data loading.Loads the mnist dataset.

```
X_train.shape ,y_train.shape ,X_test.shape ,y_test.shape
```

This is a dataset of 60000 28x28 grayscale images of the 10 digits, along with the test set of 10000 images.

```
def plot_input_img(i):
    plt.imshow(X_train[i] ,cmap='binary')
    plt.title(y_train[i])
    plt.show()
```

The visualization of the image using plots.The binary image can be seen.The function plt show and title canbe seen. The function plot_input_img passes the images.

```
for i in range(10):
    plot_input_img(i)
```

The images of handwritten can be seen. The top 10 images were seen.

```
X_train = X_train.astype(np.float32)/255
X_test = X_test.astype(np.float32)/255
```

```
#Reshape the dimensions of images to (28,28,1)
X_train = np.expand_dims(X_train, -1)
X_test = np.expand_dims(X_test, -1)
```

The For preprocess the images , the 255 is divided because for normalize the iamge and stored to X_train.Likewise for X_test.

```
X_train.shape
```

The size of mnist data 28.

```
y_train = tf.keras.utils.to_categorical(y_train)

y_test = tf.keras.utils.to_categorical(y_test)
```

Converting the output variable y[0,9]. The utils helps to whereever the value is present it tells like [0., 0., 0., ...0., 0., 0.],..array. Index can have 1.

```
model = Sequential()

model.add(Conv2D(32, (3,3), input_shape = (28,28,1), activation ='relu'))

model.add(MaxPool2D((2,2)))

model.add(Conv2D(64, (3,3), activation ='relu'))

model.add(MaxPool2D((2,2)))

model.add(Flatten())

model.add(Dropout(0.25))

model.add(Dense(10, activation="softmax"))
```

In Conv2D 32 no. of units, 3x3 kernal size ,input shape 28,28,1 and activation function relu. in Maxpool poolsize 2x2.

Increasing no. of units to 64. Then to model flatten layer and to prevent overfitting dropout 25%.The dense layer for classification.The number of classes 10 neurons.[0-9]classes.the activation function is "softmax".

```
model.summary()
```

26 decresases to 13 and decreased to 11 . Building of model has been done.

```
model.compile(optimizer= 'adam', loss = keras.losses.categorical_crossentropy ,
metrics=['accuracy'])
```

To compile it; the adam optimizer was used,loss and metrics.

```
#Callbacks

from keras.callbacks import EarlyStopping , ModelCheckpoint


#EarlyStopping

es= EarlyStopping(monitor='accuracy', min_delta= 0.01, patience=4, verbose=1)

mc=ModelCheckpoint("bestmodel_1.h5",monitor= "accuracy", verbose= 1, save_best_only =True)


cb= [es,mc]
```

The import of es and the mc. Declearing and calling es and mc. storing them in array [cb] callback is called.

Model training

```
his= model.fit(X_train,y_train,epochs =5, validation_split=0.3, callbacks= cb)
```

The model can have validationsplit 30% and storing in the his(history).The model will be trained for 5 epochs.So the model accuracy is 98%

```
model_S = keras.models.load_model("bestmodel_1.h5")
```

To save the model named like model_S.

```
score = model_S.evaluate(X_test,y_test)

print("The model accuracy is: ",score[1])
```

```
The model accuracy is:  0.9890999794006348
```

Model shows the accuracy of 98.9%.Loss is negligible with 3.21%. So the model is successfully trained by MNIST dataset.The model will easily predict whether the image belongs to which class if we pass the image to model.

Inference:

Model shows the accuracy of 98.9%.

Loss is negligible with 3.21%.

So the model is successfully trained by MNIST dataset.

The model will easily predict whether the image belongs to which class if we pass the image to model.