1) Test the number of epochs:

- **Number of epochs =15:**

0- The code:

```
import tensorflow as tf
import matplotlib.pyplot as plt
%matplotlib inline
from tensorflow import keras
from tensorflow.keras import models
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
from mlxtend.data import loadlocal_mnist
import matplotlib.pyplot as plt
from scipy import ndimage as img
from scipy.ndimage.measurements import center_of_mass
import math
import warnings
warnings.filterwarnings('ignore')
import math
import time
import random
from sklearn import preprocessing
from matplotlib import pyplot as plt
from tensorflow.keras import layers
from tensorflow.keras.optimizers import Adam,Nadam, SGD
from keras.models import Sequential
from keras.layers import MaxPooling2D
from keras.layers import Conv2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import Dropout
```

```
(x_train,y_train) , (x_test ,y_test) = keras.datasets.mnist.load_data()
#samples
#x_train=x_train[:10000,:,:]
#y_train=y_train[:10000]
#x_test=x_test[:1000,:,:]
#y_test=y_test[:1000]
#test=y_test
print(len(x_test))
print(len(x_train))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
10000
60000
```

```
# Define the One-hot Encoder
ohe = preprocessing.OneHotEncoder()# Reshape data
y_train = y_train.reshape(-1, 1)
y_test = y_test.reshape(-1, 1)

# Fit and transform training data
ohe.fit(y_train)
y_train = ohe.transform(y_train).toarray()

# Fit and transform testing data
ohe.fit(y_test)
y_test = ohe.transform(y_test).toarray()

# Print results
print(f'Value with encoding: {y_test[1]}')
```

```
Value with encoding: [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```

```
def define_model():
  model = Sequential()
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))

  model.add(Flatten())

  model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005,momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
model.summary()
```

```
Model: "sequential_4"
```

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 10.695022344589233
Test accuracy: 98.29000234603882
```

- Accuracy in the first 5 epochs in train and test data:`

```
[100] model = define_model()
      model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.2603 - accuracy: 0.9394 - val_loss: 0.0973 - val_accuracy: 0.9697
Epoch 2/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0691 - accuracy: 0.9786 - val_loss: 0.0875 - val_accuracy: 0.9749
Epoch 3/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0432 - accuracy: 0.9862 - val_loss: 0.0784 - val_accuracy: 0.9783
Epoch 4/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0268 - accuracy: 0.9913 - val_loss: 0.0777 - val_accuracy: 0.9790
Epoch 5/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0168 - accuracy: 0.9944 - val_loss: 0.1022 - val_accuracy: 0.9762
Epoch 6/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0121 - accuracy: 0.9958 - val_loss: 0.0993 - val_accuracy: 0.9797
Epoch 7/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0086 - accuracy: 0.9970 - val_loss: 0.0938 - val_accuracy: 0.9795
Epoch 8/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0058 - accuracy: 0.9981 - val_loss: 0.1015 - val_accuracy: 0.9813
Epoch 9/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0032 - accuracy: 0.9991 - val_loss: 0.1007 - val_accuracy: 0.9825
Epoch 10/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0018 - accuracy: 0.9994 - val_loss: 0.0975 - val_accuracy: 0.9829
Epoch 11/15
3000/3000 [==============================] - 10s 3ms/step - loss: 4.6986e-04 - accuracy: 0.9999 - val_loss: 0.1029 - val_accuracy: 0.9822
Epoch 12/15
3000/3000 [==============================] - 10s 3ms/step - loss: 1.4454e-04 - accuracy: 1.0000 - val_loss: 0.1038 - val_accuracy: 0.9829
Epoch 13/15
3000/3000 [==============================] - 10s 3ms/step - loss: 8.0550e-05 - accuracy: 1.0000 - val_loss: 0.1049 - val_accuracy: 0.9829
Epoch 14/15
3000/3000 [==============================] - 10s 3ms/step - loss: 6.4504e-05 - accuracy: 1.0000 - val_loss: 0.1060 - val_accuracy: 0.9828
Epoch 15/15
3000/3000 [==============================] - 10s 3ms/step - loss: 5.5202e-05 - accuracy: 1.0000 - val_loss: 0.1070 - val_accuracy: 0.9829
<keras.callbacks.History at 0x7f43f87aced0>
```

2- The number of parameters in the model:

```
[14] model.summary()

Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 27, 27, 32)        160

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 5408)              0

 dense_8 (Dense)             (None, 100)               540900

 dense_9 (Dense)             (None, 10)                1010

=================================================================
Total params: 542,070
Trainable params: 542,070
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 9.86 ~ 10 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 100 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- ## Number of epochs= 12 :

0- The code:

```
[103] model = define_model()
      model.fit(x_train, y_train, epochs=12, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 12.751582264900208
Test accuracy: 97.78000116348267
```

- Accuracy in the first 5 epochs in train data:

```
[10] model = define_model()
     model.fit(x_train, y_train, epochs=12, batch_size=20, validation_data=(x_test, y_test),shuffle=True)

Epoch 1/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.3793 - accuracy: 0.9246 - val_loss: 0.1019 - val_accuracy: 0.9683
Epoch 2/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0829 - accuracy: 0.9746 - val_loss: 0.1003 - val_accuracy: 0.9664
Epoch 3/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0514 - accuracy: 0.9840 - val_loss: 0.1141 - val_accuracy: 0.9656
Epoch 4/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0367 - accuracy: 0.9879 - val_loss: 0.0866 - val_accuracy: 0.9760
Epoch 5/12
3000/3000 [==============================] - 8s 3ms/step - loss: 0.0262 - accuracy: 0.9914 - val_loss: 0.1007 - val_accuracy: 0.9771
Epoch 6/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0207 - accuracy: 0.9932 - val_loss: 0.1004 - val_accuracy: 0.9760
Epoch 7/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0166 - accuracy: 0.9943 - val_loss: 0.1102 - val_accuracy: 0.9763
Epoch 8/12
3000/3000 [==============================] - 8s 3ms/step - loss: 0.0127 - accuracy: 0.9956 - val_loss: 0.1068 - val_accuracy: 0.9787
Epoch 9/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0103 - accuracy: 0.9964 - val_loss: 0.1203 - val_accuracy: 0.9760
Epoch 10/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0115 - accuracy: 0.9960 - val_loss: 0.1270 - val_accuracy: 0.9755
Epoch 11/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0090 - accuracy: 0.9969 - val_loss: 0.1356 - val_accuracy: 0.9765
Epoch 12/12
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0075 - accuracy: 0.9975 - val_loss: 0.1275 - val_accuracy: 0.9778
<keras.callbacks.History at 0x7f02c620eed0>
```

2- The number of parameters in the model:

```
[106] model.summary()

Model: "sequential_31"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_51 (Conv2D)          (None, 27, 27, 32)        160

 max_pooling2d_51 (MaxPoolin  (None, 13, 13, 32)       0
 g2D)

 flatten_31 (Flatten)        (None, 5408)              0

 dense_62 (Dense)            (None, 100)               540900

 dense_63 (Dense)            (None, 10)                1010

=================================================================
Total params: 542,070
Trainable params: 542,070
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 8.83 ~ 9 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 100 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

• Number of epochs=10 :

0- The code:

```
model = define_model()
model.fit(x_train, y_train, epochs=10, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

➢ Final accuracy:

```
[15] score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)

     Test loss: 15.170176327228546
     Test accuracy: 97.14000225067139
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=10, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.3517 - accuracy: 0.9180 - val_loss: 0.1238 - val_accuracy: 0.9623
Epoch 2/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0987 - accuracy: 0.9697 - val_loss: 0.1053 - val_accuracy: 0.9671
Epoch 3/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0649 - accuracy: 0.9795 - val_loss: 0.1076 - val_accuracy: 0.9680
Epoch 4/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0500 - accuracy: 0.9836 - val_loss: 0.1195 - val_accuracy: 0.9660
Epoch 5/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0377 - accuracy: 0.9875 - val_loss: 0.0984 - val_accuracy: 0.9738
Epoch 6/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0283 - accuracy: 0.9900 - val_loss: 0.1244 - val_accuracy: 0.9703
Epoch 7/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0233 - accuracy: 0.9923 - val_loss: 0.1212 - val_accuracy: 0.9721
Epoch 8/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0210 - accuracy: 0.9930 - val_loss: 0.1370 - val_accuracy: 0.9723
Epoch 9/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0180 - accuracy: 0.9937 - val_loss: 0.1376 - val_accuracy: 0.9729
Epoch 10/10
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0176 - accuracy: 0.9935 - val_loss: 0.1517 - val_accuracy: 0.9714
<keras.callbacks.History at 0x7f02b03c6710>
```

`

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 27, 27, 32)        160

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 5408)              0

 dense_8 (Dense)             (None, 100)               540900

 dense_9 (Dense)             (None, 10)                1010

=================================================================
Total params: 542,070
Trainable params: 542,070
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 9 sec

4- The average test time in each epoch:

- 0-1 sec

5- **Layers of the model:**
    - One convolution layer with channels=32, filters=(2,2)
    - Relu layer
    - Pooling layer
    - Flatten layer
    - Hidden layer (fully connected) with 100 neuron
    - Relu layer
    - o/p layer with 10 neuron
    - softmax layer

6- **The learning rate used and configuration of the optimizers:**
    - Learning rate=0.005
7- **The optimizer used with its configuration:**
    - SGD with momentum= 0.1

- **So the best case is when the number of epochs=15**
- **Observation: When the number of epochs increase so the accuracy increases**

2) Test the learning rate:

- **Lr=0.005:**

0- The code:

```
[6]  def define_model():
       model = Sequential()
       model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
       model.add(MaxPooling2D(strides=(2, 2)))

       model.add(Flatten())

       model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
       model.add(Dense(10, activation='softmax'))
       opt = SGD(lr=0.005,momentum=0.1)
       model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
       return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
[102] score = model.evaluate(x_test, y_test, verbose=0)
      print("Test loss:", score[0]*100)
      print("Test accuracy:", score[1]*100)

      Test loss: 10.695022344589233
      Test accuracy: 98.29000234603882
```

- Accuracy in the first 5 epochs in train and test data:

```
[100] model = define_model()
      model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.2603 - accuracy: 0.9394 - val_loss: 0.0973 - val_accuracy: 0.9697
Epoch 2/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0691 - accuracy: 0.9786 - val_loss: 0.0875 - val_accuracy: 0.9749
Epoch 3/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0432 - accuracy: 0.9862 - val_loss: 0.0784 - val_accuracy: 0.9783
Epoch 4/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0268 - accuracy: 0.9913 - val_loss: 0.0777 - val_accuracy: 0.9790
Epoch 5/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0168 - accuracy: 0.9944 - val_loss: 0.1022 - val_accuracy: 0.9762
Epoch 6/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0121 - accuracy: 0.9958 - val_loss: 0.0993 - val_accuracy: 0.9797
Epoch 7/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0086 - accuracy: 0.9970 - val_loss: 0.0938 - val_accuracy: 0.9795
Epoch 8/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0058 - accuracy: 0.9981 - val_loss: 0.1015 - val_accuracy: 0.9813
Epoch 9/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0032 - accuracy: 0.9991 - val_loss: 0.1007 - val_accuracy: 0.9825
Epoch 10/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0018 - accuracy: 0.9994 - val_loss: 0.0975 - val_accuracy: 0.9829
Epoch 11/15
3000/3000 [==============================] - 10s 3ms/step - loss: 4.6986e-04 - accuracy: 0.9999 - val_loss: 0.1029 - val_accuracy: 0.9822
Epoch 12/15
3000/3000 [==============================] - 10s 3ms/step - loss: 1.4454e-04 - accuracy: 1.0000 - val_loss: 0.1038 - val_accuracy: 0.9829
Epoch 13/15
3000/3000 [==============================] - 10s 3ms/step - loss: 8.0550e-05 - accuracy: 1.0000 - val_loss: 0.1049 - val_accuracy: 0.9829
Epoch 14/15
3000/3000 [==============================] - 10s 3ms/step - loss: 6.4504e-05 - accuracy: 1.0000 - val_loss: 0.1060 - val_accuracy: 0.9828
Epoch 15/15
3000/3000 [==============================] - 10s 3ms/step - loss: 5.5202e-05 - accuracy: 1.0000 - val_loss: 0.1070 - val_accuracy: 0.9829
<keras.callbacks.History at 0x7f43f87aced0>
```

2- The number of parameters in the model:

```
[14] model.summary()

Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 27, 27, 32)        160

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 5408)              0

 dense_8 (Dense)             (None, 100)               540900

 dense_9 (Dense)             (None, 10)                1010

=================================================================
Total params: 542,070
Trainable params: 542,070
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 9.86 ~ 10 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 100 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Lr=0.001:**

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))

  model.add(Flatten())

  model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.001,momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
[21] score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)

     Test loss: 9.328395128250122
     Test accuracy: 98.2200026512146
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 13s 4ms/step - loss: 0.3470 - accuracy: 0.9309 - val_loss: 0.1122 - val_accuracy: 0.9676
Epoch 2/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0845 - accuracy: 0.9746 - val_loss: 0.0854 - val_accuracy: 0.9743
Epoch 3/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0571 - accuracy: 0.9829 - val_loss: 0.0797 - val_accuracy: 0.9775
Epoch 4/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0404 - accuracy: 0.9880 - val_loss: 0.0809 - val_accuracy: 0.9770
Epoch 5/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0293 - accuracy: 0.9913 - val_loss: 0.0777 - val_accuracy: 0.9788
Epoch 6/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0213 - accuracy: 0.9944 - val_loss: 0.0782 - val_accuracy: 0.9786
Epoch 7/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0147 - accuracy: 0.9961 - val_loss: 0.0812 - val_accuracy: 0.9794
Epoch 8/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0104 - accuracy: 0.9976 - val_loss: 0.0808 - val_accuracy: 0.9804
Epoch 9/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0069 - accuracy: 0.9988 - val_loss: 0.0845 - val_accuracy: 0.9805
Epoch 10/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0049 - accuracy: 0.9994 - val_loss: 0.0870 - val_accuracy: 0.9801
Epoch 11/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0035 - accuracy: 0.9998 - val_loss: 0.0872 - val_accuracy: 0.9808
Epoch 12/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0026 - accuracy: 0.9998 - val_loss: 0.0877 - val_accuracy: 0.9817
Epoch 13/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0020 - accuracy: 0.9999 - val_loss: 0.0911 - val_accuracy: 0.9810
Epoch 14/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.0930 - val_accuracy: 0.9813
Epoch 15/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0933 - val_accuracy: 0.9822
<keras.callbacks.History at 0x7f2a906ded90>
```

2- The number of parameters in the model:

```
[14] model.summary()

Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 27, 27, 32)        160

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 5408)              0

 dense_8 (Dense)             (None, 100)               540900

 dense_9 (Dense)             (None, 10)                1010

=================================================================
Total params: 542,070
Trainable params: 542,070
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 10.2 ~ 10 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 100 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.001

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Lr= 0.01:**

0- The code:

```
[26] def define_model():
        model = Sequential()
        model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
        model.add(MaxPooling2D(strides=(2, 2)))

        model.add(Flatten())

        model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
        model.add(Dense(10, activation='softmax'))
        opt = SGD(lr=0.01,momentum=0.1)
        model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
        return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 19.60545778274536
Test accuracy: 94.30000185966492
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 10s 3ms/step - loss: 1.2368 - accuracy: 0.6409 - val_loss: 0.3661 - val_accuracy: 0.8883
Epoch 2/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.3720 - accuracy: 0.8863 - val_loss: 0.3220 - val_accuracy: 0.9021
Epoch 3/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.3100 - accuracy: 0.9063 - val_loss: 0.2709 - val_accuracy: 0.9170
Epoch 4/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.2739 - accuracy: 0.9153 - val_loss: 0.2499 - val_accuracy: 0.9260
Epoch 5/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.2437 - accuracy: 0.9249 - val_loss: 0.2603 - val_accuracy: 0.9214
Epoch 6/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.2207 - accuracy: 0.9312 - val_loss: 0.2153 - val_accuracy: 0.9319
Epoch 7/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.2022 - accuracy: 0.9366 - val_loss: 0.1873 - val_accuracy: 0.9413
Epoch 8/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.1866 - accuracy: 0.9424 - val_loss: 0.1965 - val_accuracy: 0.9416
Epoch 9/15
3000/3000 [==============================] - 8s 3ms/step - loss: 0.1814 - accuracy: 0.9440 - val_loss: 0.1762 - val_accuracy: 0.9446
Epoch 10/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.1701 - accuracy: 0.9473 - val_loss: 0.1657 - val_accuracy: 0.9493
Epoch 11/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.1662 - accuracy: 0.9484 - val_loss: 0.1699 - val_accuracy: 0.9474
Epoch 12/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.1620 - accuracy: 0.9498 - val_loss: 0.1751 - val_accuracy: 0.9471
Epoch 13/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.1582 - accuracy: 0.9499 - val_loss: 0.1886 - val_accuracy: 0.9435
Epoch 14/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.1525 - accuracy: 0.9514 - val_loss: 0.1590 - val_accuracy: 0.9534
Epoch 15/15
3000/3000 [==============================] - 8s 3ms/step - loss: 0.1446 - accuracy: 0.9549 - val_loss: 0.1961 - val_accuracy: 0.9430
<keras.callbacks.History at 0x7f02a87bad50>
```

2- The number of parameters in the model:

```
[14] model.summary()

Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 27, 27, 32)        160

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 5408)              0

 dense_8 (Dense)             (None, 100)               540900

 dense_9 (Dense)             (None, 10)                1010

=================================================================
Total params: 542,070
Trainable params: 542,070
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 8.9 ~ 9 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 100 neuron
- Relu layer
- o/p layer with 10 neorun
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.01

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Lr= 0.008:**

0- **The code:**

```
[35] def define_model():
        model = Sequential()
        model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
        model.add(MaxPooling2D(strides=(2, 2)))

        model.add(Flatten())

        model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
        model.add(Dense(10, activation='softmax'))
        opt = SGD(lr=0.008,momentum=0.1)
        model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
        return model
```

1- **Final accuracy of the model and the accuracy in the first 5 epoch:**

- Final accuracy:

```
[38] score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)

     Test loss: 21.307839453220367
     Test accuracy: 96.48000001907349
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.4104 - accuracy: 0.9260 - val_loss: 0.1169 - val_accuracy: 0.9617
Epoch 2/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1002 - accuracy: 0.9688 - val_loss: 0.0985 - val_accuracy: 0.9701
Epoch 3/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0709 - accuracy: 0.9781 - val_loss: 0.1092 - val_accuracy: 0.9653
Epoch 4/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0524 - accuracy: 0.9833 - val_loss: 0.0974 - val_accuracy: 0.9740
Epoch 5/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0385 - accuracy: 0.9875 - val_loss: 0.1187 - val_accuracy: 0.9729
Epoch 6/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0371 - accuracy: 0.9879 - val_loss: 0.1369 - val_accuracy: 0.9694
Epoch 7/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0284 - accuracy: 0.9905 - val_loss: 0.1130 - val_accuracy: 0.9729
Epoch 8/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0269 - accuracy: 0.9910 - val_loss: 0.1184 - val_accuracy: 0.9707
Epoch 9/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0273 - accuracy: 0.9912 - val_loss: 0.1455 - val_accuracy: 0.9720
Epoch 10/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0200 - accuracy: 0.9934 - val_loss: 0.1494 - val_accuracy: 0.9725
Epoch 11/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0256 - accuracy: 0.9918 - val_loss: 0.1442 - val_accuracy: 0.9720
Epoch 12/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0197 - accuracy: 0.9937 - val_loss: 0.1462 - val_accuracy: 0.9705
Epoch 13/15
3000/3000 [==============================] - 14s 5ms/step - loss: 0.0172 - accuracy: 0.9945 - val_loss: 0.1517 - val_accuracy: 0.9733
Epoch 14/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0154 - accuracy: 0.9952 - val_loss: 0.1843 - val_accuracy: 0.9737
Epoch 15/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0249 - accuracy: 0.9927 - val_loss: 0.2131 - val_accuracy: 0.9648
<keras.callbacks.History at 0x7f2a886ca050>
```

2- **The number of parameters in the model:**

```
model.summary()

Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 27, 27, 32)        160

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 5408)              0

 dense_8 (Dense)             (None, 100)               540900

 dense_9 (Dense)             (None, 10)                1010

=================================================================
Total params: 542,070
Trainable params: 542,070
Non-trainable params: 0
_________________________________________________________________
```

3- **The average time to train in each epoch:**
   - 10.2 ~ 10 sec

4- **The average test time in each epoch:**
   - 0-1 sec

5- **Layers of the model:**
   - One convolution layer with channels=32, filters=(2,2)
   - Relu layer
   - Pooling layer
   - Flatten layer
   - Hidden layer (fully connected) with 100 neuron
   - Relu layer
   - o/p layer with 10 neuron
   - softmax layer

6- **The learning rate used and configuration of the optimizers:**
   - Learning rate=0.008

7- **The optimizer used with its configuration:**
   - SGD with momentum= 0.1

❖ **So the best case is when the learning rate=0.005**

❖ **Observation: When the learning rate be smaller or greater than 0.005 the accuracy decreases.**

3) Test changing the number of CNN:

- **First model:**

0- The code:

```
[43] def define_model():
        model = Sequential()
        model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
        model.add(MaxPooling2D(strides=(2, 2)))

        model.add(Flatten())

        model.add(Dense(20, activation='relu', kernel_initializer='he_uniform'))
        model.add(Dense(10, activation='softmax'))
        opt = SGD(lr=0.005,momentum=0.1)
        model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
        return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
[43] score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)

     Test loss: 12.255346775054932
     Test accuracy: 97.42000102996826
```

- Accuracy in the first 5 epochs in train and test data:

```
[41] model = define_model()
     model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)

Epoch 1/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.3228 - accuracy: 0.9170 - val_loss: 0.1247 - val_accuracy: 0.9644
Epoch 2/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1148 - accuracy: 0.9669 - val_loss: 0.0933 - val_accuracy: 0.9718
Epoch 3/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0848 - accuracy: 0.9747 - val_loss: 0.1037 - val_accuracy: 0.9689
Epoch 4/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0698 - accuracy: 0.9791 - val_loss: 0.0925 - val_accuracy: 0.9728
Epoch 5/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0565 - accuracy: 0.9826 - val_loss: 0.0941 - val_accuracy: 0.9727
Epoch 6/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0510 - accuracy: 0.9840 - val_loss: 0.0778 - val_accuracy: 0.9790
Epoch 7/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0447 - accuracy: 0.9861 - val_loss: 0.1065 - val_accuracy: 0.9729
Epoch 8/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0406 - accuracy: 0.9873 - val_loss: 0.0999 - val_accuracy: 0.9739
Epoch 9/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0341 - accuracy: 0.9890 - val_loss: 0.1000 - val_accuracy: 0.9770
Epoch 10/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0329 - accuracy: 0.9896 - val_loss: 0.0881 - val_accuracy: 0.9775
Epoch 11/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0307 - accuracy: 0.9901 - val_loss: 0.0867 - val_accuracy: 0.9793
Epoch 12/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0252 - accuracy: 0.9921 - val_loss: 0.0955 - val_accuracy: 0.9784
Epoch 13/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.0252 - accuracy: 0.9914 - val_loss: 0.1052 - val_accuracy: 0.9777
Epoch 14/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0246 - accuracy: 0.9919 - val_loss: 0.1082 - val_accuracy: 0.9774
Epoch 15/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0195 - accuracy: 0.9937 - val_loss: 0.1226 - val_accuracy: 0.9742
<keras.callbacks.History at 0x7f2a884c2550>
```

2- The number of parameters in the model:

```
[45] model.summary()

Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_13 (Conv2D)          (None, 27, 27, 32)        160

 max_pooling2d_13 (MaxPoolin  (None, 13, 13, 32)       0
 g2D)

 flatten_13 (Flatten)        (None, 5408)              0

 dense_26 (Dense)            (None, 20)                108180

 dense_27 (Dense)            (None, 10)                210

=================================================================
Total params: 108,550
Trainable params: 108,550
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 9.6 ~ 10 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Second model:**

0- The code:

```
[47] def define_model():
        model = Sequential()
        model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
        model.add(MaxPooling2D(strides=(2, 2)))

        model.add(Flatten())

        model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
        model.add(Dense(10, activation='softmax'))
        opt = SGD(lr=0.005,momentum=0.1)
        model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
        return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
[47] score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)
```

```
Test loss: 34.032753109931946
Test accuracy: 90.39999842643738
```

- Accuracy in the first 5 epochs in train and test data:

```
[45] model = define_model()
     model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 10s 3ms/step - loss: 2.3111 - accuracy: 0.1134 - val_loss: 2.2684 - val_accuracy: 0.1292
Epoch 2/15
3000/3000 [==============================] - 9s 3ms/step - loss: 1.8104 - accuracy: 0.2864 - val_loss: 1.3670 - val_accuracy: 0.4237
Epoch 3/15
3000/3000 [==============================] - 9s 3ms/step - loss: 1.2265 - accuracy: 0.5156 - val_loss: 1.1069 - val_accuracy: 0.5591
Epoch 4/15
3000/3000 [==============================] - 9s 3ms/step - loss: 1.0744 - accuracy: 0.5640 - val_loss: 1.0439 - val_accuracy: 0.5488
Epoch 5/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.8413 - accuracy: 0.7042 - val_loss: 0.7799 - val_accuracy: 0.7687
Epoch 6/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.5621 - accuracy: 0.8369 - val_loss: 0.4932 - val_accuracy: 0.8690
Epoch 7/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.4927 - accuracy: 0.8599 - val_loss: 0.4764 - val_accuracy: 0.8737
Epoch 8/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.4575 - accuracy: 0.8710 - val_loss: 0.4259 - val_accuracy: 0.8864
Epoch 9/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.4424 - accuracy: 0.8749 - val_loss: 0.4196 - val_accuracy: 0.8910
Epoch 10/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.4338 - accuracy: 0.8793 - val_loss: 0.4886 - val_accuracy: 0.8600
Epoch 11/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.4278 - accuracy: 0.8793 - val_loss: 0.4295 - val_accuracy: 0.8843
Epoch 12/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.4213 - accuracy: 0.8817 - val_loss: 0.4315 - val_accuracy: 0.8851
Epoch 13/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.4116 - accuracy: 0.8845 - val_loss: 0.4452 - val_accuracy: 0.8889
Epoch 14/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.4046 - accuracy: 0.8858 - val_loss: 0.3610 - val_accuracy: 0.9003
Epoch 15/15
3000/3000 [==============================] - 9s 3ms/step - loss: 0.3326 - accuracy: 0.9065 - val_loss: 0.3403 - val_accuracy: 0.9040
<keras.callbacks.History at 0x7f2a8841e990>
```

2- The number of parameters in the model:

```
[51] model.summary()

Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_14 (Conv2D)          (None, 27, 27, 32)        160

 max_pooling2d_14 (MaxPoolin  (None, 13, 13, 32)       0
 g2D)

 flatten_14 (Flatten)        (None, 5408)              0

 dense_28 (Dense)            (None, 10)                54090

 dense_29 (Dense)            (None, 10)                110

=================================================================
Total params: 54,360
Trainable params: 54,360
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 9.06 ~ 9 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 10 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Third model:**

0- The code:

```
[57] def define_model():
        model = Sequential()
        model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
        model.add(MaxPooling2D(strides=(2, 2)))
        model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
        model.add(MaxPooling2D(strides=(2, 2)))

        model.add(Flatten())

        model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
        model.add(Dense(10, activation='softmax'))
        opt = SGD(lr=0.005,momentum=0.1)
        model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
        return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
[10] score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)
```

```
Test loss: 7.0219337940216064
Test accuracy: 98.00000190734863
```

- Accuracy in the first 5 epochs in train and test data:

```
[8] model = define_model()
    model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 15s 5ms/step - loss: 1.3837 - accuracy: 0.5071 - val_loss: 0.5570 - val_accuracy: 0.8020
Epoch 2/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.2377 - accuracy: 0.9301 - val_loss: 0.1398 - val_accuracy: 0.9566
Epoch 3/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1225 - accuracy: 0.9639 - val_loss: 0.0955 - val_accuracy: 0.9685
Epoch 4/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0965 - accuracy: 0.9712 - val_loss: 0.0949 - val_accuracy: 0.9724
Epoch 5/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0818 - accuracy: 0.9749 - val_loss: 0.0796 - val_accuracy: 0.9740
Epoch 6/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0745 - accuracy: 0.9769 - val_loss: 0.0818 - val_accuracy: 0.9754
Epoch 7/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0674 - accuracy: 0.9791 - val_loss: 0.0888 - val_accuracy: 0.9719
Epoch 8/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0625 - accuracy: 0.9807 - val_loss: 0.0700 - val_accuracy: 0.9769
Epoch 9/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0584 - accuracy: 0.9819 - val_loss: 0.0655 - val_accuracy: 0.9793
Epoch 10/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0566 - accuracy: 0.9824 - val_loss: 0.0733 - val_accuracy: 0.9779
Epoch 11/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0532 - accuracy: 0.9829 - val_loss: 0.0640 - val_accuracy: 0.9812
Epoch 12/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0496 - accuracy: 0.9840 - val_loss: 0.0697 - val_accuracy: 0.9783
Epoch 13/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0472 - accuracy: 0.9851 - val_loss: 0.0684 - val_accuracy: 0.9808
Epoch 14/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0459 - accuracy: 0.9855 - val_loss: 0.0771 - val_accuracy: 0.9765
Epoch 15/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0454 - accuracy: 0.9853 - val_loss: 0.0702 - val_accuracy: 0.9800
<keras.callbacks.History at 0x7f20c00361d0>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_16 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_16 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_17 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_17 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 flatten_16 (Flatten)        (None, 1152)              0

 dense_32 (Dense)            (None, 10)                11530

 dense_33 (Dense)            (None, 10)                110

=================================================================
Total params: 20,184
Trainable params: 20,184
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 10.53 ~ 11 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- convolution layer with channels=64, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 10 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Fourth model:**

0- The code:

```
def define_model():
    model = Sequential()
    model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(strides=(2, 2)))
    model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(strides=(2, 2)))
    model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(strides=(2, 2)))
    model.add(Flatten())

    model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    opt = SGD(lr=0.005,momentum=0.1)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
[7] score = model.evaluate(x_test, y_test, verbose=0)
    print("Test loss:", score[0]*100)
    print("Test accuracy:", score[1]*100)
```

```
Test loss: 10.102955251932144
Test accuracy: 97.08999991416931
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 18s 4ms/step - loss: 0.7784 - accuracy: 0.7517 - val_loss: 0.3040 - val_accuracy: 0.9084
Epoch 2/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.2753 - accuracy: 0.9166 - val_loss: 0.2368 - val_accuracy: 0.9314
Epoch 3/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.2011 - accuracy: 0.9385 - val_loss: 0.1785 - val_accuracy: 0.9458
Epoch 4/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1661 - accuracy: 0.9485 - val_loss: 0.1551 - val_accuracy: 0.9538
Epoch 5/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1456 - accuracy: 0.9549 - val_loss: 0.1378 - val_accuracy: 0.9603
Epoch 6/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1324 - accuracy: 0.9589 - val_loss: 0.1299 - val_accuracy: 0.9594
Epoch 7/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1211 - accuracy: 0.9631 - val_loss: 0.1081 - val_accuracy: 0.9648
Epoch 8/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1130 - accuracy: 0.9646 - val_loss: 0.1156 - val_accuracy: 0.9654
Epoch 9/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1046 - accuracy: 0.9684 - val_loss: 0.1171 - val_accuracy: 0.9658
Epoch 10/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0980 - accuracy: 0.9696 - val_loss: 0.0980 - val_accuracy: 0.9704
Epoch 11/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0939 - accuracy: 0.9712 - val_loss: 0.1083 - val_accuracy: 0.9671
Epoch 12/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0886 - accuracy: 0.9723 - val_loss: 0.1109 - val_accuracy: 0.9672
Epoch 13/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0865 - accuracy: 0.9726 - val_loss: 0.0900 - val_accuracy: 0.9734
Epoch 14/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0827 - accuracy: 0.9745 - val_loss: 0.1204 - val_accuracy: 0.9645
Epoch 15/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0810 - accuracy: 0.9746 - val_loss: 0.1010 - val_accuracy: 0.9709
<keras.callbacks.History at 0x7fd80a371350>
```

2- The number of parameters in the model:

```
[75] model.summary()

Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_28 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_28 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_29 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_30 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_30 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_22 (Flatten)        (None, 64)                0

 dense_44 (Dense)            (None, 10)                650

 dense_45 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 11.46 ~ 12 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- convolution layer with channels=64, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 10 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Fifth model:**

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005,momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
[94] score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)

     Test loss: 8.798278868198395
     Test accuracy: 97.39000201225281
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 12s 4ms/step - loss: 1.2816 - accuracy: 0.6113 - val_loss: 0.2343 - val_accuracy: 0.9337
Epoch 2/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.2126 - accuracy: 0.9386 - val_loss: 0.1394 - val_accuracy: 0.9600
Epoch 3/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1491 - accuracy: 0.9559 - val_loss: 0.1157 - val_accuracy: 0.9645
Epoch 4/15
3000/3000 [==============================] - 12s 4ms/step - loss: 0.1233 - accuracy: 0.9638 - val_loss: 0.1134 - val_accuracy: 0.9678
Epoch 5/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1123 - accuracy: 0.9675 - val_loss: 0.1311 - val_accuracy: 0.9636
Epoch 6/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1009 - accuracy: 0.9695 - val_loss: 0.0819 - val_accuracy: 0.9748
Epoch 7/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0928 - accuracy: 0.9724 - val_loss: 0.0912 - val_accuracy: 0.9727
Epoch 8/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0884 - accuracy: 0.9734 - val_loss: 0.0940 - val_accuracy: 0.9721
Epoch 9/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0836 - accuracy: 0.9744 - val_loss: 0.0859 - val_accuracy: 0.9745
Epoch 10/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0788 - accuracy: 0.9768 - val_loss: 0.1160 - val_accuracy: 0.9670
Epoch 11/15
3000/3000 [==============================] - 12s 4ms/step - loss: 0.0756 - accuracy: 0.9771 - val_loss: 0.0729 - val_accuracy: 0.9779
Epoch 12/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0733 - accuracy: 0.9778 - val_loss: 0.0781 - val_accuracy: 0.9764
Epoch 13/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0697 - accuracy: 0.9785 - val_loss: 0.0737 - val_accuracy: 0.9767
Epoch 14/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0668 - accuracy: 0.9796 - val_loss: 0.0910 - val_accuracy: 0.9727
Epoch 15/15
3000/3000 [==============================] - 12s 4ms/step - loss: 0.0655 - accuracy: 0.9800 - val_loss: 0.0880 - val_accuracy: 0.9739
<keras.callbacks.History at 0x7f44123f1510>
```

2- The number of parameters in the model:

```
[93] model.summary()

Model: "sequential_28"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_46 (Conv2D)          (None, 26, 26, 64)        640

 max_pooling2d_46 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_47 (Conv2D)          (None, 11, 11, 32)        18464

 max_pooling2d_47 (MaxPoolin  (None, 5, 5, 32)         0
 g2D)

 conv2d_48 (Conv2D)          (None, 4, 4, 16)          2064

 max_pooling2d_48 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_28 (Flatten)        (None, 64)                0

 dense_56 (Dense)            (None, 10)                650

 dense_57 (Dense)            (None, 10)                110

=================================================================
Total params: 21,928
Trainable params: 21,928
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 11.2 ~ 11 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- convolution layer with channels=64, filters=(3,3)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(3,3)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 10 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- **The learning rate used and configuration of the optimizers:**
    - Learning rate=0.005
7- **The optimizer used with its configuration:**
    - SGD with momentum= 0.1

- **So the best case is the fourth model such as the number of parameters= 11,368 and the accuracy= 97.08%**
- **Observation: when we increase number of neurons in fc layer and filters the number of parameters increase, and the accuracy does not increase too much.**

4) Test batch size:

- **Batch size=20:**

  0- The code:

```
[100] model = define_model()
      model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

➢ Final accuracy:

```
[7]  score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)

     Test loss: 10.102955251932144
     Test accuracy: 97.08999991416931
```

➢ Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)

Epoch 1/15
3000/3000 [==============================] - 18s 4ms/step - loss: 0.7784 - accuracy: 0.7517 - val_loss: 0.3040 - val_accuracy: 0.9084
Epoch 2/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.2753 - accuracy: 0.9166 - val_loss: 0.2368 - val_accuracy: 0.9314
Epoch 3/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.2011 - accuracy: 0.9385 - val_loss: 0.1785 - val_accuracy: 0.9458
Epoch 4/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1661 - accuracy: 0.9485 - val_loss: 0.1551 - val_accuracy: 0.9538
Epoch 5/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1456 - accuracy: 0.9549 - val_loss: 0.1378 - val_accuracy: 0.9603
Epoch 6/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1324 - accuracy: 0.9589 - val_loss: 0.1299 - val_accuracy: 0.9594
Epoch 7/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1211 - accuracy: 0.9631 - val_loss: 0.1081 - val_accuracy: 0.9648
Epoch 8/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1130 - accuracy: 0.9646 - val_loss: 0.1156 - val_accuracy: 0.9654
Epoch 9/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1046 - accuracy: 0.9684 - val_loss: 0.1171 - val_accuracy: 0.9658
Epoch 10/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0980 - accuracy: 0.9696 - val_loss: 0.0980 - val_accuracy: 0.9704
Epoch 11/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0939 - accuracy: 0.9712 - val_loss: 0.1083 - val_accuracy: 0.9671
Epoch 12/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0886 - accuracy: 0.9723 - val_loss: 0.1109 - val_accuracy: 0.9672
Epoch 13/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0865 - accuracy: 0.9726 - val_loss: 0.0900 - val_accuracy: 0.9734
Epoch 14/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0827 - accuracy: 0.9745 - val_loss: 0.1204 - val_accuracy: 0.9645
Epoch 15/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0810 - accuracy: 0.9746 - val_loss: 0.1010 - val_accuracy: 0.9709
<keras.callbacks.History at 0x7fd80a371350>
```

2- The number of parameters in the model:

```
[75] model.summary()

Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_28 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_28 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_29 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_30 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_30 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_22 (Flatten)        (None, 64)                0

 dense_44 (Dense)            (None, 10)                650

 dense_45 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 11.46 ~ 12 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- convolution layer with channels=64, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 10 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Batch size=40:**

0- The code:

```
[11] model = define_model()
     model.fit(x_train, y_train, epochs=15, batch_size=40, validation_data=(x_test, y_test),shuffle=True)
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
[18] score = model.evaluate(x_test, y_test, verbose=0)
     print("Test loss:", score[0]*100)
     print("Test accuracy:", score[1]*100)

     Test loss: 18.082310259342194
     Test accuracy: 95.5299973487854
```

- Accuracy in the first 5 epochs in train and test data:

```
[16] model = define_model()
     model.fit(x_train, y_train, epochs=15, batch_size=40, validation_data=(x_test, y_test),shuffle=True)

Epoch 1/15
1500/1500 [==============================] - 6s 4ms/step - loss: 1.8958 - accuracy: 0.3006 - val_loss: 1.4441 - val_accuracy: 0.4376
Epoch 2/15
1500/1500 [==============================] - 6s 4ms/step - loss: 1.2867 - accuracy: 0.5031 - val_loss: 1.1357 - val_accuracy: 0.5492
Epoch 3/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.8994 - accuracy: 0.6894 - val_loss: 0.6150 - val_accuracy: 0.8093
Epoch 4/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.5758 - accuracy: 0.8117 - val_loss: 0.4965 - val_accuracy: 0.8311
Epoch 5/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.4848 - accuracy: 0.8291 - val_loss: 0.4644 - val_accuracy: 0.8363
Epoch 6/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.4206 - accuracy: 0.8551 - val_loss: 0.4053 - val_accuracy: 0.9215
Epoch 7/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.3580 - accuracy: 0.9242 - val_loss: 0.3300 - val_accuracy: 0.9282
Epoch 8/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2888 - accuracy: 0.9320 - val_loss: 0.2715 - val_accuracy: 0.9380
Epoch 9/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2508 - accuracy: 0.9377 - val_loss: 0.2567 - val_accuracy: 0.9371
Epoch 10/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2233 - accuracy: 0.9441 - val_loss: 0.2329 - val_accuracy: 0.9451
Epoch 11/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2068 - accuracy: 0.9467 - val_loss: 0.2055 - val_accuracy: 0.9513
Epoch 12/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1906 - accuracy: 0.9506 - val_loss: 0.2390 - val_accuracy: 0.9368
Epoch 13/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1796 - accuracy: 0.9527 - val_loss: 0.1888 - val_accuracy: 0.9523
Epoch 14/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1696 - accuracy: 0.9552 - val_loss: 0.2119 - val_accuracy: 0.9463
Epoch 15/15
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1617 - accuracy: 0.9573 - val_loss: 0.1808 - val_accuracy: 0.9553
<keras.callbacks.History at 0x7f205617c710>
```

2- The number of parameters in the model:

```
[75] model.summary()

Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_28 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_28 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_29 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_30 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_30 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_22 (Flatten)        (None, 64)                0

 dense_44 (Dense)            (None, 10)                650

 dense_45 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 6 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- convolution layer with channels=64, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 10 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Batch size=60:**

0- The code:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=60, validation_data=(x_test, y_test),shuffle=True)
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

➢ Final accuracy:

```
[7] score = model.evaluate(x_test, y_test, verbose=0)
    print("Test loss:", score[0]*100)
    print("Test accuracy:", score[1]*100)

    Test loss: 22.055603563785553
    Test accuracy: 93.44000220298767
```

➢ Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=60, validation_data=(x_test, y_test),shuffle=True)

Epoch 1/15
1000/1000 [==============================] - 11s 6ms/step - loss: 2.2527 - accuracy: 0.1825 - val_loss: 2.0272 - val_accuracy: 0.2157
Epoch 2/15
1000/1000 [==============================] - 4s 4ms/step - loss: 1.9628 - accuracy: 0.2272 - val_loss: 1.8996 - val_accuracy: 0.2378
Epoch 3/15
1000/1000 [==============================] - 4s 4ms/step - loss: 1.8695 - accuracy: 0.2414 - val_loss: 1.8263 - val_accuracy: 0.2374
Epoch 4/15
1000/1000 [==============================] - 4s 4ms/step - loss: 1.8097 - accuracy: 0.2465 - val_loss: 1.7824 - val_accuracy: 0.2435
Epoch 5/15
1000/1000 [==============================] - 4s 4ms/step - loss: 1.7187 - accuracy: 0.2914 - val_loss: 1.4939 - val_accuracy: 0.4290
Epoch 6/15
1000/1000 [==============================] - 4s 4ms/step - loss: 1.4384 - accuracy: 0.4489 - val_loss: 1.3611 - val_accuracy: 0.4800
Epoch 7/15
1000/1000 [==============================] - 4s 4ms/step - loss: 1.2643 - accuracy: 0.4950 - val_loss: 1.0569 - val_accuracy: 0.6138
Epoch 8/15
1000/1000 [==============================] - 4s 4ms/step - loss: 1.0228 - accuracy: 0.6287 - val_loss: 0.9196 - val_accuracy: 0.6764
Epoch 9/15
1000/1000 [==============================] - 4s 4ms/step - loss: 0.8618 - accuracy: 0.6882 - val_loss: 0.7215 - val_accuracy: 0.7522
Epoch 10/15
1000/1000 [==============================] - 4s 4ms/step - loss: 0.6457 - accuracy: 0.7905 - val_loss: 0.5557 - val_accuracy: 0.8185
Epoch 11/15
1000/1000 [==============================] - 4s 4ms/step - loss: 0.5137 - accuracy: 0.8344 - val_loss: 0.4455 - val_accuracy: 0.8647
Epoch 12/15
1000/1000 [==============================] - 4s 4ms/step - loss: 0.4046 - accuracy: 0.9009 - val_loss: 0.3481 - val_accuracy: 0.9144
Epoch 13/15
1000/1000 [==============================] - 4s 4ms/step - loss: 0.3172 - accuracy: 0.9247 - val_loss: 0.3124 - val_accuracy: 0.9221
Epoch 14/15
1000/1000 [==============================] - 4s 4ms/step - loss: 0.2605 - accuracy: 0.9317 - val_loss: 0.2181 - val_accuracy: 0.9381
Epoch 15/15
1000/1000 [==============================] - 4s 4ms/step - loss: 0.2156 - accuracy: 0.9377 - val_loss: 0.2206 - val_accuracy: 0.9344
<keras.callbacks.History at 0x7f35a049ec90>
```

2- The number of parameters in the model:

```
[75] model.summary()

Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_28 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_28 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_29 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_30 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_30 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_22 (Flatten)        (None, 64)                0

 dense_44 (Dense)            (None, 10)                650

 dense_45 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 4.46 ~ 5 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- convolution layer with channels=64, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 10 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- **The learning rate used and configuration of the optimizers:**
    - Learning rate=0.005
7- **The optimizer used with its configuration:**
    - SGD with momentum= 0.1

- **The best case is when the batch size=20**
- **Observation: when the batch size increases, the accuracy decreases.**

5) changing all the activations:

- **Relu:**

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 8.51719081401825
Test accuracy: 97.39000201225281
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 11s 3ms/step - loss: 0.7961 - accuracy: 0.7483 - val_loss: 0.3217 - val_accuracy: 0.9008
Epoch 2/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.2710 - accuracy: 0.9187 - val_loss: 0.1906 - val_accuracy: 0.9470
Epoch 3/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.2011 - accuracy: 0.9392 - val_loss: 0.1720 - val_accuracy: 0.9486
Epoch 4/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1639 - accuracy: 0.9502 - val_loss: 0.1555 - val_accuracy: 0.9534
Epoch 5/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1427 - accuracy: 0.9564 - val_loss: 0.1251 - val_accuracy: 0.9616
Epoch 6/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1305 - accuracy: 0.9597 - val_loss: 0.1161 - val_accuracy: 0.9653
Epoch 7/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1199 - accuracy: 0.9630 - val_loss: 0.1227 - val_accuracy: 0.9643
Epoch 8/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1118 - accuracy: 0.9655 - val_loss: 0.1180 - val_accuracy: 0.9632
Epoch 9/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1056 - accuracy: 0.9680 - val_loss: 0.0977 - val_accuracy: 0.9692
Epoch 10/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1015 - accuracy: 0.9698 - val_loss: 0.0968 - val_accuracy: 0.9707
Epoch 11/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0964 - accuracy: 0.9701 - val_loss: 0.0994 - val_accuracy: 0.9706
Epoch 12/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0921 - accuracy: 0.9714 - val_loss: 0.1081 - val_accuracy: 0.9687
Epoch 13/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0883 - accuracy: 0.9723 - val_loss: 0.1011 - val_accuracy: 0.9689
Epoch 14/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0844 - accuracy: 0.9739 - val_loss: 0.1026 - val_accuracy: 0.9677
Epoch 15/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0817 - accuracy: 0.9742 - val_loss: 0.0852 - val_accuracy: 0.9739
<keras.callbacks.History at 0x7f25a4697a10>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 27, 27, 64)        320

 max_pooling2d_9 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 conv2d_10 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_10 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_11 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_11 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_3 (Flatten)         (None, 64)                0

 dense_6 (Dense)             (None, 10)                650

 dense_7 (Dense)             (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 10.4 ~ 11 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- convolution layer with channels=64, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 10 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

• Sigmoid:

The difference in (loss, accuracy) for each epoch, Test loss and Test accuracy.

Loss increased and accuracy decreased.

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='sigmoid', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='sigmoid', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='sigmoid', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Flatten())

  model.add(Dense(10, activation='sigmoid', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 81.35636448860168
Test accuracy: 80.15000224113464
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 20s 7ms/step - loss: 2.3098 - accuracy: 0.1102 - val_loss: 2.2988 - val_accuracy: 0.1135
Epoch 2/15
3000/3000 [==============================] - 19s 6ms/step - loss: 2.2978 - accuracy: 0.1156 - val_loss: 2.2963 - val_accuracy: 0.1136
Epoch 3/15
3000/3000 [==============================] - 19s 6ms/step - loss: 2.2951 - accuracy: 0.1223 - val_loss: 2.2937 - val_accuracy: 0.1135
Epoch 4/15
3000/3000 [==============================] - 19s 6ms/step - loss: 2.2916 - accuracy: 0.1163 - val_loss: 2.2886 - val_accuracy: 0.1135
Epoch 5/15
3000/3000 [==============================] - 19s 6ms/step - loss: 2.2855 - accuracy: 0.1287 - val_loss: 2.2806 - val_accuracy: 0.1603
Epoch 6/15
3000/3000 [==============================] - 20s 7ms/step - loss: 2.2738 - accuracy: 0.1658 - val_loss: 2.2642 - val_accuracy: 0.2042
Epoch 7/15
3000/3000 [==============================] - 20s 7ms/step - loss: 2.2482 - accuracy: 0.2647 - val_loss: 2.2245 - val_accuracy: 0.3107
Epoch 8/15
3000/3000 [==============================] - 20s 7ms/step - loss: 2.1811 - accuracy: 0.3780 - val_loss: 2.1191 - val_accuracy: 0.3937
Epoch 9/15
3000/3000 [==============================] - 20s 7ms/step - loss: 2.0240 - accuracy: 0.4292 - val_loss: 1.9165 - val_accuracy: 0.4708
Epoch 10/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.8093 - accuracy: 0.4917 - val_loss: 1.7068 - val_accuracy: 0.5320
Epoch 11/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.6110 - accuracy: 0.5634 - val_loss: 1.5183 - val_accuracy: 0.6137
Epoch 12/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.4200 - accuracy: 0.6385 - val_loss: 1.3203 - val_accuracy: 0.6677
Epoch 13/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.2180 - accuracy: 0.6949 - val_loss: 1.1154 - val_accuracy: 0.7183
Epoch 14/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.0308 - accuracy: 0.7434 - val_loss: 0.9443 - val_accuracy: 0.7605
Epoch 15/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.8824 - accuracy: 0.7795 - val_loss: 0.8136 - val_accuracy: 0.8015
<keras.callbacks.History at 0x7fadc0bbdb50>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_15 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_16 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_16 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_17 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_17 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_5 (Flatten)         (None, 64)                0

 dense_10 (Dense)            (None, 10)                650

 dense_11 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 19.6 ~ 20 sec

4- The average test time in each epoch:

- 0-1 sec

5- **Layers of the model:**
    - One convolution layer with channels=32, filters=(2,2)
    - Relu layer
    - Pooling layer
    - convolution layer with channels=32, filters=(2,2)
    - Relu layer
    - Pooling layer
    - convolution layer with channels=16, filters=(2,2)
    - Relu layer
    - Pooling layer
    - Flatten layer
    - Hidden layer (fully connected) with 20 neuron
    - Relu layer
    - o/p layer with 10 neuron
    - softmax layer

6- **The learning rate used and configuration of the optimizers:**
    - Learning rate=0.005

7- **The optimizer used with its configuration:**
    - SGD with momentum= 0.1

- ## Softplus:

  The difference in (loss, accuracy) for each epoch, Test loss and Test accuracy.

  Loss increased and accuracy decreased.

  0- **The code:**

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='softplus', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='softplus', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='softplus', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Flatten())

  model.add(Dense(10, activation='softplus', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 17.455923557281494
Test accuracy: 95.05000114440918
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 20s 6ms/step - loss: 1.7620 - accuracy: 0.3800 - val_loss: 1.3646 - val_accuracy: 0.4690
Epoch 2/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.1290 - accuracy: 0.5961 - val_loss: 0.7737 - val_accuracy: 0.7784
Epoch 3/15
3000/3000 [==============================] - 21s 7ms/step - loss: 0.5386 - accuracy: 0.8362 - val_loss: 0.3265 - val_accuracy: 0.9019
Epoch 4/15
3000/3000 [==============================] - 21s 7ms/step - loss: 0.3302 - accuracy: 0.9013 - val_loss: 0.3661 - val_accuracy: 0.8926
Epoch 5/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.2732 - accuracy: 0.9196 - val_loss: 0.2938 - val_accuracy: 0.9178
Epoch 6/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.2393 - accuracy: 0.9288 - val_loss: 0.2292 - val_accuracy: 0.9335
Epoch 7/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.2193 - accuracy: 0.9359 - val_loss: 0.2054 - val_accuracy: 0.9395
Epoch 8/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.2066 - accuracy: 0.9388 - val_loss: 0.1938 - val_accuracy: 0.9453
Epoch 9/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1947 - accuracy: 0.9432 - val_loss: 0.2400 - val_accuracy: 0.9353
Epoch 10/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1855 - accuracy: 0.9456 - val_loss: 0.2074 - val_accuracy: 0.9436
Epoch 11/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1791 - accuracy: 0.9480 - val_loss: 0.1842 - val_accuracy: 0.9485
Epoch 12/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1717 - accuracy: 0.9496 - val_loss: 0.1658 - val_accuracy: 0.9543
Epoch 13/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1667 - accuracy: 0.9508 - val_loss: 0.1637 - val_accuracy: 0.9561
Epoch 14/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1623 - accuracy: 0.9524 - val_loss: 0.1821 - val_accuracy: 0.9480
Epoch 15/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1593 - accuracy: 0.9534 - val_loss: 0.1746 - val_accuracy: 0.9505
<keras.callbacks.History at 0x7fae38e2b890>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_18 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_18 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_19 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_19 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_20 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_20 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_6 (Flatten)         (None, 64)                0

 dense_12 (Dense)            (None, 10)                650

 dense_13 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 19.8 ~ 20 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- **Tanh:**

The difference in (loss, accuracy, validation_accuracy and validation_loss) for each epoch, Test loss and Test accuracy.

Loss increased and accuracy decreased.

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='tanh', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='tanh', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='tanh', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Flatten())

  model.add(Dense(10, activation='tanh', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 9.699803590774536
Test accuracy: 97.24000096321106
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.1670 - accuracy: 0.7093 - val_loss: 0.6404 - val_accuracy: 0.8624
Epoch 2/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.4819 - accuracy: 0.8907 - val_loss: 0.3509 - val_accuracy: 0.9224
Epoch 3/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.3139 - accuracy: 0.9232 - val_loss: 0.2516 - val_accuracy: 0.9414
Epoch 4/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.2432 - accuracy: 0.9397 - val_loss: 0.2155 - val_accuracy: 0.9454
Epoch 5/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.2016 - accuracy: 0.9483 - val_loss: 0.1762 - val_accuracy: 0.9555
Epoch 6/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1742 - accuracy: 0.9548 - val_loss: 0.1559 - val_accuracy: 0.9602
Epoch 7/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1566 - accuracy: 0.9585 - val_loss: 0.1439 - val_accuracy: 0.9630
Epoch 8/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1401 - accuracy: 0.9637 - val_loss: 0.1308 - val_accuracy: 0.9647
Epoch 9/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1301 - accuracy: 0.9658 - val_loss: 0.1234 - val_accuracy: 0.9679
Epoch 10/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1228 - accuracy: 0.9675 - val_loss: 0.1138 - val_accuracy: 0.9696
Epoch 11/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1166 - accuracy: 0.9686 - val_loss: 0.1093 - val_accuracy: 0.9694
Epoch 12/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1064 - accuracy: 0.9713 - val_loss: 0.1153 - val_accuracy: 0.9681
Epoch 13/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1054 - accuracy: 0.9714 - val_loss: 0.1025 - val_accuracy: 0.9703
Epoch 14/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1011 - accuracy: 0.9723 - val_loss: 0.0993 - val_accuracy: 0.9737
Epoch 15/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.0982 - accuracy: 0.9732 - val_loss: 0.0970 - val_accuracy: 0.9724
<keras.callbacks.History at 0x7fae38d5f0d0>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_21 (Conv2D)           (None, 27, 27, 64)        320

 max_pooling2d_21 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_22 (Conv2D)           (None, 12, 12, 32)        8224

 max_pooling2d_22 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_23 (Conv2D)           (None, 5, 5, 16)          2064

 max_pooling2d_23 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_7 (Flatten)          (None, 64)                0

 dense_14 (Dense)             (None, 10)                650

 dense_15 (Dense)             (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 19.46 ~ 20 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- **The learning rate used and configuration of the optimizers:**
    - Learning rate=0.005

7- **The optimizer used with its configuration:**
    - SGD with momentum= 0.1

- **The best activation function is Relu with Learning rate=0.005(best value of parameter of SGD optimizer).**
- **Observation: when we changed the activation function into any function except relu, the accuracy decreases and the time of training increases.**

6) Change in optimizers:

- **SGD (At learning rate=0.005, momentum=0.1):**

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 8.51719081401825
Test accuracy: 97.39000201225281
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 11s 3ms/step - loss: 0.7961 - accuracy: 0.7483 - val_loss: 0.3217 - val_accuracy: 0.9008
Epoch 2/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.2710 - accuracy: 0.9187 - val_loss: 0.1906 - val_accuracy: 0.9470
Epoch 3/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.2011 - accuracy: 0.9392 - val_loss: 0.1720 - val_accuracy: 0.9486
Epoch 4/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1639 - accuracy: 0.9502 - val_loss: 0.1555 - val_accuracy: 0.9534
Epoch 5/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1427 - accuracy: 0.9564 - val_loss: 0.1251 - val_accuracy: 0.9616
Epoch 6/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1305 - accuracy: 0.9597 - val_loss: 0.1161 - val_accuracy: 0.9653
Epoch 7/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1199 - accuracy: 0.9630 - val_loss: 0.1227 - val_accuracy: 0.9643
Epoch 8/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1118 - accuracy: 0.9655 - val_loss: 0.1180 - val_accuracy: 0.9632
Epoch 9/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.1056 - accuracy: 0.9680 - val_loss: 0.0977 - val_accuracy: 0.9692
Epoch 10/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.1015 - accuracy: 0.9698 - val_loss: 0.0968 - val_accuracy: 0.9707
Epoch 11/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0964 - accuracy: 0.9701 - val_loss: 0.0994 - val_accuracy: 0.9706
Epoch 12/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0921 - accuracy: 0.9714 - val_loss: 0.1081 - val_accuracy: 0.9687
Epoch 13/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0883 - accuracy: 0.9723 - val_loss: 0.1011 - val_accuracy: 0.9689
Epoch 14/15
3000/3000 [==============================] - 10s 3ms/step - loss: 0.0844 - accuracy: 0.9739 - val_loss: 0.1026 - val_accuracy: 0.9677
Epoch 15/15
3000/3000 [==============================] - 11s 4ms/step - loss: 0.0817 - accuracy: 0.9742 - val_loss: 0.0852 - val_accuracy: 0.9739
<keras.callbacks.History at 0x7f25a4697a10>
```

2- The number of parameters in the model:

```
model.summary()

Model: "sequential_9"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_27 (Conv2D)           (None, 27, 27, 64)        320

 max_pooling2d_27 (MaxPoolin  (None, 13, 13, 64)        0
 g2D)

 conv2d_28 (Conv2D)           (None, 12, 12, 32)        8224

 max_pooling2d_28 (MaxPoolin  (None, 6, 6, 32)          0
 g2D)

 conv2d_29 (Conv2D)           (None, 5, 5, 16)          2064

 max_pooling2d_29 (MaxPoolin  (None, 2, 2, 16)          0
 g2D)

 flatten_9 (Flatten)          (None, 64)                0

 dense_18 (Dense)             (None, 10)                650

 dense_19 (Dense)             (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 10.4 ~ 11 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with momentum= 0.1

- ## Adadelta(At Learning rate=0.005):

The difference in (loss, accuracy) for each epoch, Test loss and Test accuracy.

Loss increased and accuracy decreased.

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = Adadelta(lr=0.005)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 61.74129247665405
Test accuracy: 82.22000002861023
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 18s 6ms/step - loss: 5.7495 - accuracy: 0.2044 - val_loss: 2.5770 - val_accuracy: 0.2654
Epoch 2/15
3000/3000 [==============================] - 18s 6ms/step - loss: 2.2226 - accuracy: 0.2930 - val_loss: 1.9856 - val_accuracy: 0.3478
Epoch 3/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.8719 - accuracy: 0.3859 - val_loss: 1.7279 - val_accuracy: 0.4408
Epoch 4/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.6516 - accuracy: 0.4651 - val_loss: 1.5312 - val_accuracy: 0.5113
Epoch 5/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.4736 - accuracy: 0.5303 - val_loss: 1.3594 - val_accuracy: 0.5704
Epoch 6/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.3160 - accuracy: 0.5895 - val_loss: 1.2096 - val_accuracy: 0.6254
Epoch 7/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.1824 - accuracy: 0.6361 - val_loss: 1.0855 - val_accuracy: 0.6673
Epoch 8/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.0727 - accuracy: 0.6731 - val_loss: 0.9860 - val_accuracy: 0.7011
Epoch 9/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.9833 - accuracy: 0.7035 - val_loss: 0.9035 - val_accuracy: 0.7249
Epoch 10/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.9081 - accuracy: 0.7282 - val_loss: 0.8345 - val_accuracy: 0.7489
Epoch 11/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.8449 - accuracy: 0.7491 - val_loss: 0.7789 - val_accuracy: 0.7643
Epoch 12/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.7908 - accuracy: 0.7647 - val_loss: 0.7271 - val_accuracy: 0.7872
Epoch 13/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.7439 - accuracy: 0.7790 - val_loss: 0.6859 - val_accuracy: 0.8007
Epoch 14/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.7024 - accuracy: 0.7926 - val_loss: 0.6497 - val_accuracy: 0.8126
Epoch 15/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.6654 - accuracy: 0.8043 - val_loss: 0.6174 - val_accuracy: 0.8222
<keras.callbacks.History at 0x7f2db2121490>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_39 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_39 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_40 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_40 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_41 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_41 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_13 (Flatten)        (None, 64)                0

 dense_26 (Dense)            (None, 10)                650

 dense_27 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 17.93 ~ 18 sec

4- The average test time in each epoch:

- 0-1 sec

5- **Layers of the model:**
   - One convolution layer with channels=32, filters=(2,2)
   - Relu layer
   - Pooling layer
   - convolution layer with channels=32, filters=(2,2)
   - Relu layer
   - Pooling layer
   - convolution layer with channels=16, filters=(2,2)
   - Relu layer
   - Pooling layer
   - Flatten layer
   - Hidden layer (fully connected) with 20 neuron
   - Relu layer
   - o/p layer with 10 neuron
   - softmax layer

6- **The learning rate used and configuration of the optimizers:**
   - Learning rate=0.005

7- **The optimizer used with its configuration:**
   - Adadelta with Learning rate=0.005

- ## Adagrad(At Learning rate=0.005):

  The difference in (loss, accuracy) for each epoch, Test loss and Test accuracy.

  Loss increased and accuracy decreased.

  ### 0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = Adagrad(lr=0.005)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 17.618240416049957
Test accuracy: 94.52000260353088
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 130s 6ms/step - loss: 1.5924 - accuracy: 0.4966 - val_loss: 1.1152 - val_accuracy: 0.6665
Epoch 2/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.8699 - accuracy: 0.7425 - val_loss: 0.6101 - val_accuracy: 0.8319
Epoch 3/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.5166 - accuracy: 0.8570 - val_loss: 0.4131 - val_accuracy: 0.8860
Epoch 4/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.3873 - accuracy: 0.8896 - val_loss: 0.3391 - val_accuracy: 0.9036
Epoch 5/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.3227 - accuracy: 0.9071 - val_loss: 0.2878 - val_accuracy: 0.9153
Epoch 6/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.2834 - accuracy: 0.9160 - val_loss: 0.2669 - val_accuracy: 0.9209
Epoch 7/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.2574 - accuracy: 0.9237 - val_loss: 0.2450 - val_accuracy: 0.9254
Epoch 8/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.2378 - accuracy: 0.9297 - val_loss: 0.2287 - val_accuracy: 0.9307
Epoch 9/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.2229 - accuracy: 0.9343 - val_loss: 0.2150 - val_accuracy: 0.9358
Epoch 10/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.2103 - accuracy: 0.9379 - val_loss: 0.2060 - val_accuracy: 0.9360
Epoch 11/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.2003 - accuracy: 0.9413 - val_loss: 0.1951 - val_accuracy: 0.9373
Epoch 12/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.1919 - accuracy: 0.9426 - val_loss: 0.1893 - val_accuracy: 0.9408
Epoch 13/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.1848 - accuracy: 0.9450 - val_loss: 0.1832 - val_accuracy: 0.9422
Epoch 14/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.1781 - accuracy: 0.9472 - val_loss: 0.1816 - val_accuracy: 0.9416
Epoch 15/15
3000/3000 [==============================] - 17s 6ms/step - loss: 0.1724 - accuracy: 0.9483 - val_loss: 0.1762 - val_accuracy: 0.9452
<keras.callbacks.History at 0x7f2db1c5c750>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_42 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_42 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_43 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_43 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_44 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_44 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_14 (Flatten)        (None, 64)                0

 dense_28 (Dense)            (None, 10)                650

 dense_29 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 17 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- Adagrad with Learning rate=0.005

- **The best Optimizer is SGD with Learning rate=0.005 and with momentum= 0.1.**
- **Observation: when we changed the optimizer into any optimizer except SGD, the accuracy decreases and the time of training increases.**

7) Effect of dropout layer

- **Before flatten layer (at rate=0.2):**

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))

  model.add(Dropout(rate = 0.2, noise_shape=None, seed=None))

  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 8.942525088787079
Test accuracy: 97.15999960899353
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.1535 - accuracy: 0.6195 - val_loss: 0.2958 - val_accuracy: 0.9076
Epoch 2/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.3776 - accuracy: 0.8807 - val_loss: 0.1791 - val_accuracy: 0.9456
Epoch 3/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.2898 - accuracy: 0.9094 - val_loss: 0.1438 - val_accuracy: 0.9564
Epoch 4/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.2464 - accuracy: 0.9229 - val_loss: 0.1393 - val_accuracy: 0.9585
Epoch 5/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.2203 - accuracy: 0.9305 - val_loss: 0.1168 - val_accuracy: 0.9650
Epoch 6/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.2123 - accuracy: 0.9344 - val_loss: 0.1299 - val_accuracy: 0.9601
Epoch 7/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.2007 - accuracy: 0.9383 - val_loss: 0.1177 - val_accuracy: 0.9654
Epoch 8/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1955 - accuracy: 0.9393 - val_loss: 0.1124 - val_accuracy: 0.9665
Epoch 9/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1838 - accuracy: 0.9429 - val_loss: 0.1115 - val_accuracy: 0.9688
Epoch 10/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1783 - accuracy: 0.9449 - val_loss: 0.0958 - val_accuracy: 0.9728
Epoch 11/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.1720 - accuracy: 0.9466 - val_loss: 0.1036 - val_accuracy: 0.9682
Epoch 12/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1683 - accuracy: 0.9478 - val_loss: 0.0925 - val_accuracy: 0.9722
Epoch 13/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1619 - accuracy: 0.9496 - val_loss: 0.1040 - val_accuracy: 0.9682
Epoch 14/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1606 - accuracy: 0.9492 - val_loss: 0.0934 - val_accuracy: 0.9702
Epoch 15/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1579 - accuracy: 0.9507 - val_loss: 0.0894 - val_accuracy: 0.9716
<keras.callbacks.History at 0x7f5c16a9c710>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_18 (Conv2D)           (None, 27, 27, 64)        320

 max_pooling2d_18 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_19 (Conv2D)           (None, 12, 12, 32)        8224

 max_pooling2d_19 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_20 (Conv2D)           (None, 5, 5, 16)          2064

 max_pooling2d_20 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 dropout_6 (Dropout)          (None, 2, 2, 16)          0

 flatten_6 (Flatten)          (None, 64)                0

 dense_12 (Dense)             (None, 10)                650

 dense_13 (Dense)             (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 19.06 ~ 19 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Dropout layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron

- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with Learning rate=0.005 and with momentum= 0.1.

- **Before flatten layer (at** rate=0.85**):**

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))

  model.add(Dropout(rate = 0.85, noise_shape=None, seed=None))

  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 59.054332971572876
Test accuracy: 88.37000131607056
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 19s 6ms/step - loss: 2.3228 - accuracy: 0.1679 - val_loss: 2.0465 - val_accuracy: 0.3914
Epoch 2/15
3000/3000 [==============================] - 18s 6ms/step - loss: 2.1013 - accuracy: 0.2191 - val_loss: 1.7683 - val_accuracy: 0.5608
Epoch 3/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.9823 - accuracy: 0.2728 - val_loss: 1.6124 - val_accuracy: 0.6574
Epoch 4/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.8523 - accuracy: 0.3221 - val_loss: 1.3789 - val_accuracy: 0.7088
Epoch 5/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.7183 - accuracy: 0.3778 - val_loss: 1.1726 - val_accuracy: 0.7725
Epoch 6/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.6349 - accuracy: 0.4085 - val_loss: 1.1188 - val_accuracy: 0.8125
Epoch 7/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.5671 - accuracy: 0.4381 - val_loss: 0.9370 - val_accuracy: 0.8114
Epoch 8/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.4960 - accuracy: 0.4611 - val_loss: 0.7934 - val_accuracy: 0.8551
Epoch 9/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.4492 - accuracy: 0.4785 - val_loss: 0.7648 - val_accuracy: 0.8689
Epoch 10/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.4077 - accuracy: 0.4963 - val_loss: 0.7237 - val_accuracy: 0.8787
Epoch 11/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.3778 - accuracy: 0.5098 - val_loss: 0.7085 - val_accuracy: 0.8657
Epoch 12/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.3542 - accuracy: 0.5166 - val_loss: 0.7277 - val_accuracy: 0.8795
Epoch 13/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.3255 - accuracy: 0.5273 - val_loss: 0.6546 - val_accuracy: 0.8619
Epoch 14/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.3077 - accuracy: 0.5363 - val_loss: 0.6660 - val_accuracy: 0.8715
Epoch 15/15
3000/3000 [==============================] - 18s 6ms/step - loss: 1.2845 - accuracy: 0.5436 - val_loss: 0.5905 - val_accuracy: 0.8837
<keras.callbacks.History at 0x7f5c1673c8d0>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_24 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_24 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_25 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_25 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_26 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_26 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 dropout_8 (Dropout)         (None, 2, 2, 16)          0

 flatten_8 (Flatten)         (None, 64)                0

 dense_16 (Dense)            (None, 10)                650

 dense_17 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 19 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer

- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Dropout layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with Learning rate=0.005 and with momentum= 0.1.

❖ **When dropout layer is before flatten layer, the best rate is 0.2 because it has the most accuracy.**

- **After flatten layer (at** rate=0.001)

0- The code:

```
def define_model():
   model = Sequential()
   model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
   model.add(MaxPooling2D(strides=(2, 2)))
   model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
   model.add(MaxPooling2D(strides=(2, 2)))
   model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
   model.add(MaxPooling2D(strides=(2, 2)))

   model.add(Flatten())

   model.add(Dropout(rate = 0.001, noise_shape=None, seed=None))

   model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
   model.add(Dense(10, activation='softmax'))
   opt = SGD(lr=0.005, momentum=0.1)
   model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
   return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 10.358086228370667
Test accuracy: 96.97999954223633
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.6758 - accuracy: 0.7943 - val_loss: 0.2703 - val_accuracy: 0.9217
Epoch 2/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.2690 - accuracy: 0.9172 - val_loss: 0.1901 - val_accuracy: 0.9407
Epoch 3/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.2078 - accuracy: 0.9353 - val_loss: 0.1602 - val_accuracy: 0.9516
Epoch 4/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1749 - accuracy: 0.9453 - val_loss: 0.1455 - val_accuracy: 0.9555
Epoch 5/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1547 - accuracy: 0.9524 - val_loss: 0.1236 - val_accuracy: 0.9630
Epoch 6/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.1414 - accuracy: 0.9558 - val_loss: 0.1324 - val_accuracy: 0.9585
Epoch 7/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1322 - accuracy: 0.9586 - val_loss: 0.1350 - val_accuracy: 0.9586
Epoch 8/15
3000/3000 [==============================] - 20s 6ms/step - loss: 0.1245 - accuracy: 0.9610 - val_loss: 0.1242 - val_accuracy: 0.9628
Epoch 9/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1160 - accuracy: 0.9646 - val_loss: 0.1040 - val_accuracy: 0.9674
Epoch 10/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1102 - accuracy: 0.9657 - val_loss: 0.1336 - val_accuracy: 0.9622
Epoch 11/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1054 - accuracy: 0.9673 - val_loss: 0.1726 - val_accuracy: 0.9474
Epoch 12/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1014 - accuracy: 0.9690 - val_loss: 0.1144 - val_accuracy: 0.9657
Epoch 13/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.0974 - accuracy: 0.9692 - val_loss: 0.1365 - val_accuracy: 0.9574
Epoch 14/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.0921 - accuracy: 0.9713 - val_loss: 0.0976 - val_accuracy: 0.9716
Epoch 15/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.0897 - accuracy: 0.9723 - val_loss: 0.1036 - val_accuracy: 0.9698
<keras.callbacks.History at 0x7f5c10c15e10>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_27 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_27 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_28 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_28 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_29 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_29 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_9 (Flatten)         (None, 64)                0

 dropout_9 (Dropout)         (None, 64)                0

 dense_18 (Dense)            (None, 10)                650

 dense_19 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 20 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Dropout layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with Learning rate=0.005 and with momentum= 0.1.

- **After flatten layer (at** rate=0.5)

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))

  model.add(Flatten())

  model.add(Dropout(rate = 0.5, noise_shape=None, seed=None))

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 14.278300106525421
Test accuracy: 96.03999853134155
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 21s 7ms/step - loss: 2.0079 - accuracy: 0.3015 - val_loss: 1.0731 - val_accuracy: 0.7254
Epoch 2/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.2041 - accuracy: 0.5980 - val_loss: 0.4923 - val_accuracy: 0.8828
Epoch 3/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.8774 - accuracy: 0.7100 - val_loss: 0.3607 - val_accuracy: 0.9022
Epoch 4/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.7001 - accuracy: 0.7716 - val_loss: 0.2829 - val_accuracy: 0.9288
Epoch 5/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.6064 - accuracy: 0.8038 - val_loss: 0.2259 - val_accuracy: 0.9373
Epoch 6/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.5468 - accuracy: 0.8228 - val_loss: 0.2196 - val_accuracy: 0.9364
Epoch 7/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.5014 - accuracy: 0.8392 - val_loss: 0.1926 - val_accuracy: 0.9445
Epoch 8/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.4713 - accuracy: 0.8497 - val_loss: 0.1706 - val_accuracy: 0.9474
Epoch 9/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.4414 - accuracy: 0.8591 - val_loss: 0.1793 - val_accuracy: 0.9465
Epoch 10/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.4220 - accuracy: 0.8660 - val_loss: 0.1541 - val_accuracy: 0.9544
Epoch 11/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.4002 - accuracy: 0.8722 - val_loss: 0.1544 - val_accuracy: 0.9547
Epoch 12/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.3902 - accuracy: 0.8764 - val_loss: 0.1388 - val_accuracy: 0.9569
Epoch 13/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.3813 - accuracy: 0.8795 - val_loss: 0.1788 - val_accuracy: 0.9457
Epoch 14/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.3666 - accuracy: 0.8840 - val_loss: 0.1579 - val_accuracy: 0.9499
Epoch 15/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.3623 - accuracy: 0.8850 - val_loss: 0.1428 - val_accuracy: 0.9604
<keras.callbacks.History at 0x7f5c101f1590>
```

2- The number of parameters in the model:

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_30 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_30 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_31 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_31 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_32 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_32 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_10 (Flatten)        (None, 64)                0

 dropout_10 (Dropout)        (None, 64)                0

 dense_20 (Dense)            (None, 10)                650

 dense_21 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 20 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Dropout layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with Learning rate=0.005 and with momentum= 0.1.

❖ **When dropout layer is after flatten layer, the best rate is 0.001 because it has the most accuracy.**

- **After first Dense layer(hidden layer) (at rate=0.001):**

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))

  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))

  model.add(Dropout(rate = 0.001, noise_shape=None, seed=None))

  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

➢ Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 8.147069066762924
Test accuracy: 97.45000004768372
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 21s 7ms/step - loss: 0.6247 - accuracy: 0.8162 - val_loss: 0.2217 - val_accuracy: 0.9333
Epoch 2/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.2050 - accuracy: 0.9382 - val_loss: 0.1510 - val_accuracy: 0.9563
Epoch 3/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1557 - accuracy: 0.9531 - val_loss: 0.1309 - val_accuracy: 0.9619
Epoch 4/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1358 - accuracy: 0.9585 - val_loss: 0.1084 - val_accuracy: 0.9671
Epoch 5/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1238 - accuracy: 0.9621 - val_loss: 0.1007 - val_accuracy: 0.9681
Epoch 6/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1140 - accuracy: 0.9648 - val_loss: 0.1086 - val_accuracy: 0.9652
Epoch 7/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.1081 - accuracy: 0.9665 - val_loss: 0.1057 - val_accuracy: 0.9687
Epoch 8/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.1044 - accuracy: 0.9676 - val_loss: 0.0893 - val_accuracy: 0.9710
Epoch 9/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.0990 - accuracy: 0.9694 - val_loss: 0.1063 - val_accuracy: 0.9655
Epoch 10/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.0956 - accuracy: 0.9705 - val_loss: 0.0826 - val_accuracy: 0.9755
Epoch 11/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.0928 - accuracy: 0.9704 - val_loss: 0.0915 - val_accuracy: 0.9719
Epoch 12/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.0886 - accuracy: 0.9724 - val_loss: 0.0950 - val_accuracy: 0.9720
Epoch 13/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.0838 - accuracy: 0.9737 - val_loss: 0.0935 - val_accuracy: 0.9728
Epoch 14/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.0844 - accuracy: 0.9738 - val_loss: 0.0970 - val_accuracy: 0.9690
Epoch 15/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.0804 - accuracy: 0.9749 - val_loss: 0.0815 - val_accuracy: 0.9745
<keras.callbacks.History at 0x7f5c0feb2ad0>
```

2- The number of parameters in the model:

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_36 (Conv2D)          (None, 27, 27, 64)        320

 max_pooling2d_36 (MaxPoolin  (None, 13, 13, 64)       0
 g2D)

 conv2d_37 (Conv2D)          (None, 12, 12, 32)        8224

 max_pooling2d_37 (MaxPoolin  (None, 6, 6, 32)         0
 g2D)

 conv2d_38 (Conv2D)          (None, 5, 5, 16)          2064

 max_pooling2d_38 (MaxPoolin  (None, 2, 2, 16)         0
 g2D)

 flatten_12 (Flatten)        (None, 64)                0

 dense_24 (Dense)            (None, 10)                650

 dropout_13 (Dropout)        (None, 10)                0

 dense_25 (Dense)            (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 19 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- Dropout layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with Learning rate=0.005 and with momentum= 0.1.

- **After first Dense layer(hidden layer) (at rate=0.3):**

0- The code:

```
def define_model():
  model = Sequential()
  model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))
  model.add(Conv2D(16, (2, 2), activation='relu', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D(strides=(2, 2)))

  model.add(Flatten())

  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform'))

  model.add(Dropout(rate = 0.3, noise_shape=None, seed=None))

  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.005, momentum=0.1)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

1- Final accuracy of the model and the accuracy in the first 5 epoch:

- Final accuracy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0]*100)
print("Test accuracy:", score[1]*100)
```

```
Test loss: 26.383966207504272
Test accuracy: 94.17999982833862
```

- Accuracy in the first 5 epochs in train and test data:

```
model = define_model()
model.fit(x_train, y_train, epochs=15, batch_size=20, validation_data=(x_test, y_test),shuffle=True)
```

```
Epoch 1/15
3000/3000 [==============================] - 22s 7ms/step - loss: 1.8303 - accuracy: 0.3300 - val_loss: 1.2706 - val_accuracy: 0.6008
Epoch 2/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.3444 - accuracy: 0.5104 - val_loss: 0.8987 - val_accuracy: 0.7265
Epoch 3/15
3000/3000 [==============================] - 20s 7ms/step - loss: 1.1895 - accuracy: 0.5647 - val_loss: 0.7480 - val_accuracy: 0.7868
Epoch 4/15
3000/3000 [==============================] - 19s 6ms/step - loss: 1.0530 - accuracy: 0.6411 - val_loss: 0.5282 - val_accuracy: 0.9126
Epoch 5/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.9648 - accuracy: 0.6843 - val_loss: 0.4846 - val_accuracy: 0.9071
Epoch 6/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.9451 - accuracy: 0.6897 - val_loss: 0.4602 - val_accuracy: 0.9203
Epoch 7/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.8950 - accuracy: 0.7078 - val_loss: 0.4355 - val_accuracy: 0.9241
Epoch 8/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.8575 - accuracy: 0.7206 - val_loss: 0.3905 - val_accuracy: 0.9344
Epoch 9/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.8313 - accuracy: 0.7317 - val_loss: 0.3733 - val_accuracy: 0.9329
Epoch 10/15
3000/3000 [==============================] - 18s 6ms/step - loss: 0.7724 - accuracy: 0.7518 - val_loss: 0.3009 - val_accuracy: 0.9441
Epoch 11/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.7183 - accuracy: 0.7663 - val_loss: 0.2981 - val_accuracy: 0.9430
Epoch 12/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.6863 - accuracy: 0.7762 - val_loss: 0.2706 - val_accuracy: 0.9464
Epoch 13/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.6575 - accuracy: 0.7886 - val_loss: 0.2824 - val_accuracy: 0.9378
Epoch 14/15
3000/3000 [==============================] - 20s 7ms/step - loss: 0.6482 - accuracy: 0.7908 - val_loss: 0.2611 - val_accuracy: 0.9453
Epoch 15/15
3000/3000 [==============================] - 19s 6ms/step - loss: 0.6101 - accuracy: 0.8076 - val_loss: 0.2638 - val_accuracy: 0.9418
<keras.callbacks.History at 0x7efc673575d0>
```

2- The number of parameters in the model:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 27, 27, 64)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 12, 12, 32)        8224

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 5, 5, 16)          2064

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 16)         0
 2D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 10)                650

 dropout (Dropout)           (None, 10)                0

 dense_1 (Dense)             (None, 10)                110

=================================================================
Total params: 11,368
Trainable params: 11,368
Non-trainable params: 0
_________________________________________________________________
```

3- The average time to train in each epoch:

- 19 sec

4- The average test time in each epoch:

- 0-1 sec

5- Layers of the model:

- One convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=32, filters=(2,2)
- Relu layer
- Pooling layer
- convolution layer with channels=16, filters=(2,2)
- Relu layer
- Pooling layer
- Flatten layer
- Hidden layer (fully connected) with 20 neuron
- Relu layer
- Dropout layer
- o/p layer with 10 neuron
- softmax layer

6- The learning rate used and configuration of the optimizers:

- Learning rate=0.005

7- The optimizer used with its configuration:

- SGD with Learning rate=0.005 and with momentum= 0.1.

❖ **When dropout layer is after first Dense layer (hidden layer), the best rate is 0.001 because it has the most accuracy.**

❖ **The best model with adding Dropout layer is when dropout layer is after first Dense layer (hidden layer) with rate 0.001.**

Conclusion:

Due to the diversity of materials, learning about materials using supervised learning is a very challenging task that has received a lot of attention in the past decades. This project aims specifically classifies materials and provides an accurate and observational evaluation of five different CNN models in MNIST dataset. Since image segmentation and understanding are some of the primary challenges that computer vision systems attempt to address, this project took additional look at approaches such as patch segmentation and transfer learning and how they affect the way a network learns features at different layers.

Through empirical tests, the understanding of the real-world scene was examined by considering contextual modeling among the various components of the created system. The pipeline consists of training and test sets that are fed as inputs to the pre-trained network on the ImageNet dataset. The network will then extract the features into the feature vector fed into the classifier which will compute the score map. The mean average precision will rank each image in the data set and output the results.

The collected results demonstrate that a recent accuracy of 97.08% is achieved using a deep neural network on the MNIST dataset when using patch segmentation and transfer learning as methods. All results obtained from training the network on material classification showed an improvement in overall performance. To ensure that the results capture the full complexity of the tests, all classes were run from the data set. What is new in the scene is the analysis of how fast the architectures are as well as observing the factors that can influence the system. The number of images each database provides affects the amount of time it takes for architectures to evaluate all architectures differently with respect to the size of each data set. The average time of training the data is differ from 4 seconds to 20 seconds and average test time is from 0 second to 1 second.