Name : Esraa yazid Ahmed

ID : 20190093

Supervised learning

Group : 3AI-S1

Spring 2022

Assignment 3

INTRODUCTION:

Link to the full code :

https://colab.research.google.com/drive/1Z1kRe3YlYV_K5ZQCAUpxLGX5mx8pJdwe?usp=sharing

---

FIRST TEST: we will try testing the number of epochs:

1. Trying 10 epochs

```
from tensorflow.python.keras.activations import softmax
from tensorflow.python.keras.backend import conv2d
model1 =Sequential()
#first convolution layer
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28,
 28, 1)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

    model1.add(Flatten())
    model1.add(Dense(64, activation='relu'))
    model1.add(Dense(10, activation='softmax'))
    from tensorflow.keras.optimizers import SGD
    sgd = SGD(learning_rate=0.001, momentum=0.0)
    model1.compile(optimizer= sgd,
                   loss=tf.keras.losses.SparseCategoricalCrossent
ropy(),
                   metrics=['accuracy'])

    history = model1.fit(train_X, train_labels, epochs=10,
                         validation_data=(test_X, test_labels),sh
uffle = True,batch_size = 32)
    test_loss, test_acc = model1.evaluate(test_X,  test_labels,
verbose=1)
```

1. final accuracy if the model :95.40%
2. accuracy of the first 5 epochs :
    1. Epoch 1/10 :loss: 2.1509 - accuracy: 0.3941 - val_loss: 1.7631 - val_accuracy: 0.7074
    2. Epoch 2/10 :loss: 0.9498 - accuracy: 0.7906 - val_loss: 0.5131 - val_accuracy: 0.8688
    3. Epoch 3/10 : loss: 0.4294 - accuracy: 0.8808 - val_loss: 0.3451 - val_accuracy: 0.9044
    4. Epoch 4/10 :loss: 0.3248 - accuracy: 0.9079 - val_loss: 0.2746 - val_accuracy: 0.9217
    5. Epoch 5/10 :loss: 0.2718 - accuracy: 0.9218 - val_loss: 0.2330 - val_accuracy: 0.9329

3. Average training time for each epoch : 16 s

4. Average training time for each epoch :16 s
5. Learning rate = 0.001
6. Number of parameters : 121,930
7. Used optimizer : SGD , LR = 0.001 , momentum = 0.0
8. Layers :
    1. Cnn layer followed by relu activation layer and a 2D maxpooling layer with kernel 2*2 and stride 2*2
    2. Another CNN layer followed by relu activation and a 2D maxpooling layer with kernel 2*2 and stride 2*2
    3. Flattening layer
    4. Fully connected layer with output 64 and relu activation
    5. Fully connected layer with output 10 and softmax activation

---

23. Trying 25 epochs :

```
24.      model1 =Sequential()
25.      model1.add(Conv2D(32, (3, 3), activation='relu', input_shape
   =(28, 28, 1)))
26.      model1.add(MaxPooling2D((2, 2),strides=(2,2)))
27.      model1.add(Conv2D(64, (3, 3), activation='relu'))
28.      model1.add(MaxPooling2D((2, 2),strides=(2,2)))
29.
30.      model1.add(Flatten())
31.      model1.add(Dense(64, activation='relu'))
32.      model1.add(Dense(10, activation='softmax'))
33.
34.      sgd = SGD(learning_rate=0.001, momentum=0.0)
35.
36.      model1.compile(optimizer= sgd,
37.                     loss=tf.keras.losses.SparseCategoricalCrossent
   ropy(),
38.                     metrics=['accuracy'])
39.
40.      history = model1.fit(train_X, train_labels, epochs=25,
41.                           validation_data=(test_X, test_labels),sh
   uffle = True,batch_size = 32)
42.      test_loss, test_acc = model1.evaluate(test_X,  test_labels,
   verbose=1)
43.
```

1. Final accuracy : 97.64 %
2. Accuracy of the 5 first epochs :
    1. Epoch 1/25 :loss: 2.0716 - accuracy: 0.4557 - val_loss: 1.4673 - val_accuracy: 0.7564
    2. Epoch 2/25 :loss: 0.7506 - accuracy: 0.8267 - val_loss: 0.4291 - val_accuracy: 0.8834
    3. Epoch 3/25: loss: 0.3755 - accuracy: 0.8927 - val_loss: 0.3154 - val_accuracy: 0.9091
    4. Epoch 4/25: loss: 0.3016 - accuracy: 0.9117 - val_loss: 0.2641 - val_accuracy: 0.9248
    5. Epoch 5/25 :loss: 0.2623 - accuracy: 0.9231 - val_loss: 0.2322 - val_accuracy: 0.9323

3. Learning rate = 0.001
4. Average training time : 10 ~ 11 seconds
5. Number of parameters : 121,930
6. Average test time = 4s/step
7. Used optimizer : SGD , LR = 0.001 , momentum = 0.0
8. Layers :
    1. Cnn layer followed by relu activation layer and a 2D maxpooling layer with kernel 2*2 and stride 2*2
    2. Another CNN layer followed by relu activation and a 2D maxpooling layer with kernel 2*2 and stride 2*2
    3. Flattening layer
    4. Fully connected layer with output 64 and relu activation
    5. Fully connected layer with output 10 and softmax activation

---

1.3. Trying 40 epochs

```
model1 =Sequential()

model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
    model1.add(Dense(64, activation='relu'))
    model1.add(Dense(10, activation='softmax'))

    sgd = SGD(learning_rate=0.001, momentum=0.0)

    model1.compile(optimizer= sgd,
                   loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                   metrics=['accuracy'])
    earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                            mode ="min", patience = 5,
                                            restore_best_weights = True)

    time_callback = TimeHistory()
    history = model1.fit(train_X, train_labels, epochs=40,
                         validation_data=(test_X, test_labels),shuffle = True,batch_size = 32 , callbacks =[time_callback,earlystopping])
    times = time_callback.times
```

```
26.   test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbo
   se=1)
```

1. final accuracy : 98.27%
2. accuracy of the first 5 epochs:
    1. Epoch 1/40 :loss: 1.9742 - accuracy: 0.4747 - val_loss: 1.2076 - val_accuracy: 0.7127
    2. Epoch 2/40 :loss: 0.7109 - accuracy: 0.8074 - val_loss: 0.4496 - val_accuracy: 0.8745
    3. Epoch 3/40: loss: 0.4003 - accuracy: 0.8854 - val_loss: 0.3318 - val_accuracy: 0.9051
    4. Epoch 4/40:loss: 0.3158 - accuracy: 0.9082 - val_loss: 0.2719 - val_accuracy: 0.9229
    5. Epoch 5/40: loss: 0.2705 - accuracy: 0.9207 - val_loss: 0.2422 - val_accuracy: 0.9303

3. Average train time : [13.28525161743164, 10.632070064544678, 10.308592081069946, 11.849564552307129, 12.000101089477539, 10.950177907943726, 10.19032907485962, 9.944242477416992, 10.437545776367188, 10.010447025299072, 10.574686527252197, 10.394061803817749, 10.083669185638428, 9.943841934204102, 10.371716260910034, 10.339104890823364, 10.3578941822052, 10.079188823699951, 10.77367901802063, 10.375660419464111, 10.312971591949463, 10.345721960067749, 10.087323665618896, 10.092099905014038, 10.476072311401367, 10.338266134262085, 10.10399079322815, 10.353189706802368, 10.408555746078491, 10.375573873519897, 10.026330709457397, 9.979304075241089, 10.386029243469238, 10.372284412384033, 10.53375768661499, 10.353919267654419, 10.202246904373169, 10.498692274093628, 10.572432041168213, 10.520873308181763]
4. Average test time : 10~ 11 seconds
5. Number of parameters : 121,930
6. Learning rate = 0.001
7. Used optimizer : SGD , LR = 0.001 , momentum = 0.0
8. Layers :
    1. Cnn layer followed by relu activation layer and a 2D maxpooling layer with kernel 2*2 and stride 2*2
    2. Another CNN layer followed by relu activation and a 2D maxpooling layer with kernel 2*2 and stride 2*2
    3. Flattening layer
    4. Fully connected layer with output 64 and relu activation
    5. Fully connected layer with output 10 and softmax activation

FINAL OBSERVATION:

Here we compared the number of epochs to choose the optimal value. We compared 10, 25, and 40. And comparing accuracies of them we find that 40 epochs gives an accuracy that is equal to 98.27% so we will use and test the other parameters with 40 epochs , but I will set an early stopping factor that will stop the process when the model starts overfitting.

We can also observe that the accuracy of the model increases by increasing the number of epochs

---

2. TESTING LEARNING RATES:

1. learning rate = 0.01

```
model1 =Sequential()
```

```
# testing learning rate = 0.01
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.01, momentum=0.0)

model1.compile(optimizer= sgd,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
               metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 32 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Final accuracy of the model :98.93 %
2. Accuracy if the first 5 epochs :
    1. Epoch 1/40 : loss: 0.5620 - accuracy: 0.8344 - val_loss: 0.1737 - val_accuracy: 0.9492
    2. Epoch 2/40 : loss: 0.1492 - accuracy: 0.9554 - val_loss: 0.1009 - val_accuracy: 0.9699
    3. Epoch 3/40 :loss: 0.1030 - accuracy: 0.9688 - val_loss: 0.0723 - val_accuracy: 0.9765
    4. Epoch 4/40 : loss: 0.0827 - accuracy: 0.9742 - val_loss: 0.0676 - val_accuracy: 0.9791
    5. Epoch 5/40 : loss: 0.0701 - accuracy: 0.9786 - val_loss: 0.0528 - val_accuracy: 0.9831
3. Number of parameters :121.930
4. Learning rate = 0.01
5. Average train time = [10.66630744934082, 10.488824844360352, 10.095211267471313, 10.666344404220581, 10.631511688232422, 10.509984016418457, 10.522321939468384, 10.4005765914917, 10.063866376876831, 10.131177425384521, 10.495679378509521, 10.164026021957397, 10.49898624420166, 10.532265424728394, 10.149816751480103, 10.598557233810425, 10.082297086715698, 10.162006616592407, 10.152749061584473, 10.657643795013428, 10.105109691619873, 10.536373138427734, 10.449999809265137, 10.120105981826782, 10.465376615524292, 10.19348406791687]
6. Average test time = 10~ 11
7. Used optimizer : SGD , LR = 0.01 , momentum = 0.0
8. Layers :
    1. Cnn layer followed by relu activation layer and a 2D maxpooling layer with kernel 2*2 and stride 2*2

2. Another CNN layer followed by relu activation and a 2D maxpooling layer with kernel 2*2 and stride 2*2
3. Flattening layer
4. Fully connected layer with output 64 and relu activation
5. Fully connected layer with output 10 and softmax activation.

2.2. Learning rate = 0.1

```
model1 =Sequential()
# testing learning rate = 0.1
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.1, momentum=0.0)

model1.compile(optimizer= sgd,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
               metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 32 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Final accuracy of the model : 99.07
2. Accuracy of the first 5 epochs :
    1. Epoch 1/40 :loss: 0.1779 - accuracy: 0.9438 - val_loss: 0.0575 - val_accuracy: 0.9803
    2. Epoch 2/40:loss: 0.0553 - accuracy: 0.9830 - val_loss: 0.0391 - val_accuracy: 0.9871
    3. Epoch 3/40:loss: 0.0384 - accuracy: 0.9879 - val_loss: 0.0394 - val_accuracy: 0.9864
    4. Epoch 4/40: loss: 0.0296 - accuracy: 0.9911 - val_loss: 0.0359 - val_accuracy: 0.9883
    5. Epoch 5/40:loss: 0.0237 - accuracy: 0.9924 - val_loss: 0.0302 - val_accuracy: 0.9907
3. Average train time : [10.947302103042603, 10.109866380691528, 11.027601480484009, 10.799046277999878, 10.092986822128296, 10.398900985717773, 10.477643251419067, 10.040547370910645, 10.456888198852539, 10.107140302658081, 10.492445945739746]
4. Average test time : 10 ~ 11 seconds

5. Number of parameters : 121,930
6. Learning rate: 0.1
7. Used optimizer : SGD , LR = 0.01 , momentum = 0.0
8. Layers:

    1. Cnn layer followed by relu activation layer and a 2D maxpooling layer with kernel 2*2 and stride 2*2

    2. Another CNN layer followed by relu activation and a 2D maxpooling layer with kernel 2*2 and stride 2*2

    3. Flattening layer

    4. Fully connected layer with output 64 and relu activation

    5. Fully connected layer with output 10 and softmax activation.

2.3. Learning rate = 0.003:

```
model1 =Sequential()
# testing learning rate = 0.003
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.003, momentum=0.0)

model1.compile(optimizer= sgd,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
               metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 32 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Final accuracy of the model : 98.81 %
2. Accuracy of the first 5 epochs :
    1. Epoch 1/40 :loss: 1.1887 - accuracy: 0.6259 - val_loss: 0.3600 - val_accuracy: 0.8965

2. Epoch 2/40 :loss: 0.3083 - accuracy: 0.9075 - val_loss: 0.2367 - val_accuracy: 0.9315
3. Epoch 3/40 :loss: 0.2218 - accuracy: 0.9345 - val_loss: 0.1715 - val_accuracy: 0.9505
4. Epoch 4/40 :loss: 0.1720 - accuracy: 0.9491 - val_loss: 0.1393 - val_accuracy: 0.9578
5. Epoch 5/40 : loss: 0.1413 - accuracy: 0.9584 - val_loss: 0.1137 - val_accuracy: 0.9679

3. Number of parameters : 121,930
4. Learning rate : 0.003
5. Average train time : [10.648336410522461, 10.470914840698242, 10.317973613739014, 10.250239372253418, 10.331140041351318, 10.368586778640747, 9.954809427261353, 10.307901382446289, 10.501755475997925, 9.986394882202148, 10.352505922317505, 10.800374746322632, 10.25293779373169, 10.060513973236084, 10.007049322128296, 10.453199863433838, 10.130411148071289, 10.090394258499146, 9.967403888702393, 10.494374513626099, 9.993850231170654, 10.417911529541016, 12.686509132385254, 10.030638933181763, 9.98386836051941, 10.323788404464722, 9.921845197677612, 10.339212656021118, 9.969069719314575, 9.973387718200684, 10.396514892578125, 10.425524234771729, 10.17065167427063, 10.054506063461304, 10.036363363265991, 10.488706111907959, 10.535224199295044, 10.08469271659851, 10.498995542526245, 10.511680364608765]
6. Average test time: 10 ~11 seconds
7. Used optimizer : SGD , LR = 0.003 , momentum = 0.0
8. Layers:
    - Cnn layer followed by relu activation layer and a 2D maxpooling layer with kernel 2*2 and stride 2*2
    - Another CNN layer followed by relu activation and a 2D maxpooling layer with kernel 2*2 and stride 2*2
    - Flattening layer
    - Fully connected layer with output 64 and relu activation
    - Fully connected layer with output 10 and softmax activation.

2.4. Learning rate = 0.005

```
model1 =Sequential()
# testing learning rate = 0.005
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.005, momentum=0.0)
```

```
model1.compile(optimizer= sgd,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 32 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Final accuracy of the model: 98. %90
2. Accuracy of the first 5 epochs :
   1. Epoch 1/40:loss: 0.7580 - accuracy: 0.7807 - val_loss: 0.3076 - val_accuracy: 0.9096
   2. Epoch 2/40 :loss: 0.2151 - accuracy: 0.9357 - val_loss: 0.1513 - val_accuracy: 0.9544
   3. Epoch 3/40:loss: 0.1489 - accuracy: 0.9556 - val_loss: 0.1075 - val_accuracy: 0.9667
   4. Epoch 4/40 :loss: 0.1177 - accuracy: 0.9650 - val_loss: 0.0930 - val_accuracy: 0.9723
   5. Epoch 5/40:loss: 0.0997 - accuracy: 0.9708 - val_loss: 0.0787 - val_accuracy: 0.9759
3. Learning rate = 0.005
4. Used optimizer : SGD , LR = 0.003 , momentum = 0.0
5. Number of parameters : 121.930
6. Average train time : [6.4069600105285645, 5.524645805358887, 6.771639585494995, 5.5406951904296875, 5.589333772659302, 5.54652214050293, 5.541147708892822, 5.558661460876465, 5.595424175262451, 5.544501066207886, 5.569594860076904, 5.555826902389526, 5.561650991439819, 5.577897071838379, 5.6011059284210205, 5.523564100265503, 5.623640537261963, 5.718517303466797, 5.6122376918792725, 5.6008710861206055, 5.5956079959869385, 5.583516597747803, 5.628449440002441, 5.507747650146484, 5.574894428253174, 5.632693290710449, 5.52852988243103, 5.612900495529175, 5.74003005027771, 5.595170736312866, 5.553348779678345, 5.582526206970215, 5.595680236816406, 5.547945022583008, 5.68924880027771, 5.60068154335022, 5.749489784240723, 5.590587139129639, 5.681508302688599, 5.658182621002197]
7. Average test time : 5 ~ 6 seconds
8. Layers :
   - Cnn layer followed by relu activation layer and a 2D maxpooling layer with kernel 2*2 and stride 2*2
   - Another CNN layer followed by relu activation and a 2D maxpooling layer with kernel 2*2 and stride 2*2
   - Flattening layer
   - Fully connected layer with output 64 and relu activation
   - Fully connected layer with output 10 and softmax activation.

Final observation:

we tested here 4 different learning rates : 0.01 , 0.1 , 0.003, 0.005 , we can observe that the learning rate 0.1 gives the highest accuracy = 99.07, so we will use 0.1 as a learning rate when using SGD as an optimizer.

---

3. TESTING DIFFERENT Architectures:

3.1. 1 CNN AND 1 FC

```
#1CNN AND 1 FC
model1 =Sequential()
# testing learning rate = 0.1
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())

model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.1, momentum=0.0)

model1.compile(optimizer= sgd,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
               metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 32 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Final accuracy of the model :98.45
2. Accuracy of the first 5 epochs :
    1. Epoch 1/40:loss: 0.2722 - accuracy: 0.9195 - val_loss: 0.1197 - val_accuracy: 0.9660
    2. Epoch 2/40:loss: 0.0975 - accuracy: 0.9725 - val_loss: 0.0758 - val_accuracy: 0.9757
    3. Epoch 3/40:loss: 0.0733 - accuracy: 0.9784 - val_loss: 0.0609 - val_accuracy: 0.9810
    4. Epoch 4/40:loss: 0.0611 - accuracy: 0.9822 - val_loss: 0.0622 - val_accuracy: 0.9800
    5. Epoch 5/40:loss: 0.0536 - accuracy: 0.9838 - val_loss: 0.0590 - val_accuracy: 0.9806
3. Number of parameters : 54,410
4. Average train time : [8.484246253967285, 7.416532516479492, 7.487035036087036, 7.415877103805542, 7.5231146812438965, 7.9372618198394775, 7.915063381195068, 7.440031051635742, 7.917072534561157, 7.481266021728516, 7.968287706375122, 7.526801347732544, 8.035681009292603]
5. Average test time : 7~ 8 seconds

6. Learning rate = 0.1
7. Used optimizer : SGD , LR = 0.1 , momentum = 0.0
8. Layers:
    - A CNN layer with a relu activation function and a maxpooling2D with kernel size 2*2 and strides 2*2
    - Flattening layer
    - A fully connected layer with 10 outputs and softmax activation

---

3.2. 2CNN AND 3FC

```
model1 =Sequential()
# testing learning rate = 0.1
# 2 CNN and 3 FC
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(32, activation='relu'))
model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.1, momentum=0.0)

model1.compile(optimizer= sgd,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
               metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 32 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Layers :
    1. 2 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
    2. Flattening layer
    3. 2 Fully connected layers with relu activation functions
    4. Another fully connected layer with softmax activation function
2. Final accuracy of the model : 99.18
3. Accuracy of the first 5 epochs :

1. Epoch 1/40:loss: 0.2088 - accuracy: 0.9340 - val_loss: 0.0618 - val_accuracy: 0.9802
2. Epoch 2/40:loss: 0.0580 - accuracy: 0.9822 - val_loss: 0.0420 - val_accuracy: 0.9871
3. Epoch 3/40:loss: 0.0390 - accuracy: 0.9882 - val_loss: 0.0343 - val_accuracy: 0.9895
4. Epoch 4/40:loss: 0.0295 - accuracy: 0.9906 - val_loss: 0.0355 - val_accuracy: 0.9886
5. Epoch 5/40 loss: 0.0243 - accuracy: 0.9922 - val_loss: 0.0297 - val_accuracy: 0.9898

4. Average train time: [6.328695058822632, 5.764863014221191, 5.6650331020355225, 5.74968147277832, 5.780593156814575, 5.768248081207275, 5.766294717788696, 5.742184400558472, 5.763494253158569, 5.757632493972778, 5.755081653594971, 5.807462215423584, 5.757145643234253, 5.727740287780762]
5. Average test time : 5~7 seconds
6. Learning rate = 0.1
7. Used optimizer : SGD , LR = 0.1 , momentum = 0.0
8. Number of parameters : 123,690

3.3. 3CNN AND 3FC:

```
model1 =Sequential()
# testing learning rate = 0.1 , trying 3CNN and 3FC
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(32, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(32, activation='relu'))
model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.1, momentum=0.0)

model1.compile(optimizer= sgd,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 32 , callbacks =[time_callback,earlystopping])
```

```
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Layers :
    1. 3 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
    2. Flattening layer
    3. 2 Fully connected layers with relu activation functions
    4. Another fully connected layer with softmax activation
2. Accuracy of the final epoch : 98.59%
3. Accuracy of the first 5 epochs:
    1. Epoch 1/40:loss: 0.3402 - accuracy: 0.8899 - val_loss: 0.0895 - val_accuracy: 0.9743
    2. Epoch 2/40:loss: 0.0864 - accuracy: 0.9741 - val_loss: 0.0806 - val_accuracy: 0.9752
    3. Epoch 3/40:loss: 0.0609 - accuracy: 0.9813 - val_loss: 0.0530 - val_accuracy: 0.9854
    4. Epoch 4/40:loss: 0.0492 - accuracy: 0.9849 - val_loss: 0.0613 - val_accuracy: 0.9808
    5. Epoch 5/40:loss: 0.0412 - accuracy: 0.9872 - val_loss: 0.0562 - val_accuracy: 0.9831
4. Number of parameters:1,802
5. Learning rate =0.1
6. Used optimizer : SGD , LR = 0.1 , momentum = 0.0
7. Average train time : [6.6867616176605225, 6.2012739181518555, 6.1085498332977295, 6.056229114532471, 6.767364740371704, 6.054199695587158, 6.107012987136841, 6.086791038513184, 6.102833032608032, 6.042593002319336, 6.016862630844116]
8. Average test time : 6 seconds

3.4. 3CNN AND 2FC

```
model1 =Sequential()
# testing learning rate = 0.1 , trying 3CNN and 2FC
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(32, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(32, activation='relu'))
model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.1, momentum=0.0)

model1.compile(optimizer= sgd,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
               metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
```

```
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 32 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Layers :
    - 3 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
    - Flattening layer
    - Fully connected layer with relu activation
    - Fully connected layer with softmax
2. Accuracy of the final epoch :98.58
3. Number of parameters : 38,666
4. Accuracy of the first 5 epochs :
    1. Epoch 1/40:loss: 0.3089 - accuracy: 0.9020 - val_loss: 0.1018 - val_accuracy: 0.9687
    2. Epoch 2/40:loss: 0.0844 - accuracy: 0.9740 - val_loss: 0.0607 - val_accuracy: 0.9811
    3. Epoch 3/40:loss: 0.0614 - accuracy: 0.9814 - val_loss: 0.0586 - val_accuracy: 0.9834
    4. Epoch 4/40:loss: 0.0483 - accuracy: 0.9850 - val_loss: 0.0603 - val_accuracy: 0.9841
    5. Epoch 5/40:loss: 0.0414 - accuracy: 0.9872 - val_loss: 0.0607 - val_accuracy: 0.9832
5. Average train time : [9.056811332702637, 6.00836968421936, 5.844502925872803, 6.525131464004517, 5.908543109893799, 5.901589393615723, 5.83287239074707, 5.832604169845581, 5.852531909942627, 5.828242301940918, 6.560439586639404, 6.220177412033081]
6. Average test time : 5~ 6 seconds
7. Learning rate =0.1
8. Used optimizer : SGD , LR = 0.1 , momentum = 0.0

FINAL OBSERVATION:

WE can notice that the model that includes 2 convolution layers and 3 fully connected layers has given higher accuracy than the other 3 tested models as it equals 99.18 so we will keep working with that model.

In addition, it is noticeable that models that gave higher accuracies have included bigger number of fully connected layers

---

4- TESTING DIFFERENT BATCH SIZE

4.1. Batch size = 64

```
5. model1 =Sequential()
6. # testing learning rate = 0.1
7. # 2 CNN and 3 FC
8. # batch size = 64
```

```
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
    model1.add(MaxPooling2D((2, 2),strides=(2,2)))
    model1.add(Conv2D(64, (3, 3), activation='relu'))
    model1.add(MaxPooling2D((2, 2),strides=(2,2)))

    model1.add(Flatten())
    model1.add(Dense(64, activation='relu'))
    model1.add(Dense(32, activation='relu'))
    model1.add(Dense(10, activation='softmax'))

    sgd = SGD(learning_rate=0.1, momentum=0.0)

    model1.compile(optimizer= sgd,
                   loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                   metrics=['accuracy'])
    earlystopping = callbacks.EarlyStopping(monitor ="val_loss",

                                            mode ="min", patience = 5,
                                            restore_best_weights = True)

    time_callback = TimeHistory()
    history = model1.fit(train_X, train_labels, epochs=40,
                         validation_data=(test_X, test_labels),shuffle = True,batch_size = 64 , callbacks =[time_callback,earlystopping])
    times = time_callback.times
    test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. accuracy of the first 5 layers :
   - Epoch 1/40:loss: 0.2903 - accuracy: 0.9075 - val_loss: 0.0708 - val_accuracy: 0.9785
   - Epoch 2/40:loss: 0.0686 - accuracy: 0.9784 - val_loss: 0.0447 - val_accuracy: 0.9858
   - Epoch 3/40:loss: 0.0479 - accuracy: 0.9852 - val_loss: 0.0413 - val_accuracy: 0.9865
   - Epoch 4/40:loss: 0.0379 - accuracy: 0.9879 - val_loss: 0.0328 - val_accuracy: 0.9885
   - Epoch 5/40:loss: 0.0299 - accuracy: 0.9905 - val_loss: 0.0310 - val_accuracy: 0.9886
2. final accuracy of the model : 99.19 %
3. layers :
   - 2 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
   - Flattening layer
   - 2 Fully connected layers with relu activation functions
   - Another fully connected layer with softmax activation function

4. Average train time : [4.016432762145996, 3.4067440032958984, 3.7209880352020264, 3.364039897918701, 3.3387527465820312, 3.359880208969116, 3.801670789718628,

3.6276445388793945, 3.6455094814300537, 3.5915675163269043, 3.55423903465271, 3.357398748397827, 3.322221040725708, 3.3238909244537354, 3.3913235664367676, 3.455738067626953, 3.7189836502075195, 3.399812698364258]

5. Average test time : 3 ~ 4 seconds
6. Number of parameters: 123,690
7. Learning rate = 0.1
8. Used optimizer : SGD , LR = 0.1 , momentum = 0.0

4.2. Batch size = 96

```
model1 =Sequential()
# testing learning rate = 0.1
# 2 CNN and 3 FC
# batch size = 96
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
    model1.add(MaxPooling2D((2, 2),strides=(2,2)))
    model1.add(Conv2D(64, (3, 3), activation='relu'))
    model1.add(MaxPooling2D((2, 2),strides=(2,2)))

    model1.add(Flatten())
    model1.add(Dense(64, activation='relu'))
    model1.add(Dense(32, activation='relu'))
    model1.add(Dense(10, activation='softmax'))

    sgd = SGD(learning_rate=0.1, momentum=0.0)

    model1.compile(optimizer= sgd,
                   loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                   metrics=['accuracy'])
    earlystopping = callbacks.EarlyStopping(monitor ="val_loss",

                                            mode ="min", patience = 5,
                                            restore_best_weights = True)

    time_callback = TimeHistory()
    history = model1.fit(train_X, train_labels, epochs=40,
                         validation_data=(test_X, test_labels),shuffle = True,batch_size = 96 , callbacks =[time_callback,earlystopping])
    times = time_callback.times
    test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. accuracy of the first 5 layers :

- Epoch 1/4: loss: 0.3860 - accuracy: 0.8763 - val_loss: 0.1010 - val_accuracy: 0.9678

- Epoch 2/40:loss: 0.0862 - accuracy: 0.9729 - val_loss: 0.0884 - val_accuracy: 0.9701
- Epoch 3/40:loss: 0.0609 - accuracy: 0.9807 - val_loss: 0.0547 - val_accuracy: 0.9820
- Epoch 4/40:loss: 0.0464 - accuracy: 0.9859 - val_loss: 0.0449 - val_accuracy: 0.9850
- Epoch 5/40:loss: 0.0374 - accuracy: 0.9881 - val_loss: 0.0329 - val_accuracy: 0.9894

2. final accuracy of the model : 99.03 %
3. layers :
    - 2 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
    - Flattening layer
    - 2 Fully connected layers with relu activation functions
    - Another fully connected layer with softmax activation function

4. Average train time : [2.959050178527832, 2.2978005409240723, 2.601702928543091, 2.311624765396118, 2.168905019760132, 2.2645761966705322, 2.32275128364563, 2.289583683013916, 2.210627317428589, 2.1708009243011475, 2.188220739364624, 2.693422794342041, 2.355470895767212]

5. Average test time : 2 seconds
6. Number of parameters: 123,690
7. Learning rate = 0.1
8. Used optimizer : SGD , LR = 0.1 , momentum = 0.0

Full observation:

We can notice that batch size 64 gives higher accuracy than 32 and 96 this accuracy is equal to 99.19% , so we will keep testing with batch size =64

5- Testing different activation functions:

5.1. Sigmoid activation function:

```
model1 =Sequential()
# testing learning rate = 0.1
# 2 CNN and 3 FC
# batch size = 64 , activation function will be sigmoid
  model1.add(Conv2D(32, (3, 3), activation='sigmoid', input_shape=(28, 28, 1)))
  model1.add(MaxPooling2D((2, 2),strides=(2,2)))
  model1.add(Conv2D(64, (3, 3), activation='sigmoid'))
  model1.add(MaxPooling2D((2, 2),strides=(2,2)))

  model1.add(Flatten())
  model1.add(Dense(64, activation='sigmoid'))
  model1.add(Dense(32, activation='sigmoid'))
  model1.add(Dense(10, activation='softmax'))

  sgd = SGD(learning_rate=0.1, momentum=0.0)
```

```
21.
22.   model1.compile(optimizer= sgd,
23.                  loss=tf.keras.losses.SparseCategoricalCrossent
   ropy(),
24.                  metrics=['accuracy'])
25.   earlystopping = callbacks.EarlyStopping(monitor ="val_loss",

26.                                          mode ="min", patienc
   e = 5,
27.                                          restore_best_weights
    = True)
28.
29.   time_callback = TimeHistory()
30.   history = model1.fit(train_X, train_labels, epochs=40,
31.                        validation_data=(test_X, test_labels),sh
   uffle = True,batch_size = 64 , callbacks =[time_callback,earlys
   topping])
32.   times = time_callback.times
33.   test_loss, test_acc = model1.evaluate(test_X,  test_labels,
   verbose=1)
```

1. accuracy of the final epoch :98.48
2. accuracy of the first 5 epochs :
    1. Epoch 1/40:loss: 2.3064 - accuracy: 0.1068 - val_loss: 2.3096 - val_accuracy: 0.1028
    2. Epoch 2/40:loss: 2.3038 - accuracy: 0.1078 - val_loss: 2.3020 - val_accuracy: 0.1135
    3. Epoch 3/40:loss: 2.3020 - accuracy: 0.1124 - val_loss: 2.3026 - val_accuracy: 0.1135
    4. Epoch 4/40:loss: 2.2940 - accuracy: 0.1297 - val_loss: 2.2776 - val_accuracy: 0.2497
    5. Epoch 5/40:loss: 1.7290 - accuracy: 0.4425 - val_loss: 0.9315 - val_accuracy: 0.7372
3. Number of parameters : 123,690
4. Learning rate = 0.1
5. Used optimizer : SGD , LR = 0.1 , momentum = 0.0
6. layers :
    - 2 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
    - Flattening layer
    - 2 Fully connected layers with relu activation functions
    - Another fully connected layer with softmax activation function
7. Average train time: [4.367655992507935, 3.720479726791382, 3.338361978530884, 3.6639647483825684, 3.739867687225342, 3.9349794387817383, 3.557138204574585, 3.3655545711517334, 3.6426117420196533, 3.6802406311035156, 3.6767983436584473, 3.5736868381500244, 3.6751344203948975, 3.640183687210083, 3.734370708465576, 3.7161905765533447, 4.379518747329712, 3.3885579109191895, 3.7565677165985107, 3.66469407081604, 3.3001341819763184, 3.515160322189331, 3.6699423789978027, 4.0349321365356445, 4.885350704193115, 5.046709060668945, 3.7519640922546387, 4.367229223251343, 3.5672640800476074, 3.349890232086181 6, 3.328296661376953, 3.40008544921875, 3.343167304992676, 3.4531607627868652, 4.099081993103027, 4.829409837722778, 3.874272584915161, 4.423378944396973, 3.916652202606201, 3.6958553791046143]
8. Average test time :3 ~ 4 seconds

5.2. Tanh activation function

```
model1 =Sequential()
# testing learning rate = 0.1
# 2 CNN and 3 FC
# batch size = 64 , activation function will be tanh
  model1.add(Conv2D(32, (3, 3), activation='tanh', input_shape=(28, 28, 1)))
  model1.add(MaxPooling2D((2, 2),strides=(2,2)))
  model1.add(Conv2D(64, (3, 3), activation='tanh'))
  model1.add(MaxPooling2D((2, 2),strides=(2,2)))

  model1.add(Flatten())
  model1.add(Dense(64, activation='tanh'))
  model1.add(Dense(32, activation='tanh'))
  model1.add(Dense(10, activation='softmax'))

  sgd = SGD(learning_rate=0.1, momentum=0.0)

  model1.compile(optimizer= sgd,
                 loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                 metrics=['accuracy'])
  earlystopping = callbacks.EarlyStopping(monitor ="val_loss",

                                          mode ="min", patience = 5,
                                          restore_best_weights = True)

  time_callback = TimeHistory()
  history = model1.fit(train_X, train_labels, epochs=40,
                       validation_data=(test_X, test_labels),shuffle = True,batch_size = 64 , callbacks =[time_callback,earlystopping])
  times = time_callback.times
  test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. accuracy of the first five epochs:
    1. Epoch 1/40:loss: 0.2344 - accuracy: 0.9365 - val_loss: 0.0736 - val_accuracy: 0.9785
    2. Epoch 2/40:loss: 0.0673 - accuracy: 0.9805 - val_loss: 0.0517 - val_accuracy: 0.9852
    3. Epoch 3/40:loss: 0.0467 - accuracy: 0.9862 - val_loss: 0.0413 - val_accuracy: 0.9868
    4. Epoch 4/40:loss: 0.0348 - accuracy: 0.9902 - val_loss: 0.0374 - val_accuracy: 0.9877
    5. Epoch 5/40:loss: 0.0278 - accuracy: 0.9920 - val_loss: 0.0365 - val_accuracy: 0.9891
2. Accuracy of the final epoch : 99.06
3. Number of parameters: 123,690
4. Average train time : [4.3000712394714355, 3.316728115081787, 3.315232038497925, 3.4011199474334717, 3.6583423614501953, 3.651688814163208, 3.6783761978149414,

3.6783487796783447, 3.642878293991089, 3.6715195178985596, 3.7345173358917236, 4.851062297821045, 3.7517940998077393, 4.050658464431763, 3.4767770767211914, 3.6623072624206543, 3.714305877685547, 3.3614184856414795]

5. Average test time : 3 ~ 4 seconds
6. Learning rate = 0.1
7. Used optimizer : SGD , LR = 0.1 , momentum = 0.0
8. layers :
    - 2 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
    - Flattening layer
    - 2 Fully connected layers with relu activation functions
    - Another fully connected layer with softmax activation function

---

5.3 testing on softplus:

```
model1 =Sequential()
# testing learning rate = 0.1
# 2 CNN and 3 FC
# batch size = 64 , activation function will be softplus
  model1.add(Conv2D(32, (3, 3), activation='softplus', input_shape=(28, 28, 1)))
  model1.add(MaxPooling2D((2, 2),strides=(2,2)))
  model1.add(Conv2D(64, (3, 3), activation='softplus'))
  model1.add(MaxPooling2D((2, 2),strides=(2,2)))

  model1.add(Flatten())
  model1.add(Dense(64, activation='softplus'))
  model1.add(Dense(32, activation='softplus'))
  model1.add(Dense(10, activation='softmax'))

  sgd = SGD(learning_rate=0.1, momentum=0.0)

  model1.compile(optimizer= sgd,
                 loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                 metrics=['accuracy'])
  earlystopping = callbacks.EarlyStopping(monitor ="val_loss",

                                          mode ="min", patience = 5,
                                          restore_best_weights = True)

  time_callback = TimeHistory()
  history = model1.fit(train_X, train_labels, epochs=40,
                       validation_data=(test_X, test_labels),shuffle = True,batch_size = 64 , callbacks =[time_callback,earlystopping])
```

```
32.   times = time_callback.times
33.   test_loss, test_acc = model1.evaluate(test_X,  test_labels,
   verbose=1)
```

1. accuracy of the first 5 epochs :
    - Epoch 1/40:loss: 2.0021 - accuracy: 0.2448 - val_loss: 0.5331 - val_accuracy: 0.8260
    - Epoch 2/40:loss: 0.1802 - accuracy: 0.9446 - val_loss: 0.1132 - val_accuracy: 0.9635
    - Epoch 3/40:loss: 0.1028 - accuracy: 0.9694 - val_loss: 0.0788 - val_accuracy: 0.9753
    - Epoch 4/40:loss: 0.0729 - accuracy: 0.9773 - val_loss: 0.0977 - val_accuracy: 0.9674
    - Epoch 5/40:loss: 0.0582 - accuracy: 0.9818 - val_loss: 0.0748 - val_accuracy: 0.9755
2. Accuracy of the final epoch : 98.68
3. Number of parameters : 123,690
4. Learning rate = 0.1
5. Average train time: [4.391051292419434, 3.716292142868042, 3.4012131690979004, 3.3829143047332764, 3.4249937534332275, 3.7440598011016846, 3.762845993041992, 3.670835494995117, 3.705263137817383, 3.3708441257476807, 3.508181095123291, 3.6963117122650146, 3.4666929244995117, 3.392502546310425, 3.381305456161499, 3.38926362991333]
6. Average test time:3 ~4 seconds
7. Layers:
    - 2 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
    - Flattening layer
    - 2 Fully connected layers with relu activation functions
    - Another fully connected layer with softmax activation function
8. Used optimizer : SGD , LR = 0.1 , momentum = 0.0

FINAL OBSERVATION:

All of the activation functions gave a kind of close accuracies but the heighest accuracy was given by relu activation function with an accuracy = 99.19 % so we will keep testing with it

---

6. TESTING WITH DIFFERENT OPTIMIZERS:
    6.1. ADAM OPTIMIZER:

```
7. model1 =Sequential()
8. # testing learning rate = 0.1
9. # 2 CNN and 3 FC
10. # batch size = 64 , # activation function = relu , adam optimizer
```

```
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(32, activation='relu'))
model1.add(Dense(10, activation='softmax'))


model1.compile(optimizer= 'adam',
               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
               metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = True,batch_size = 64 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. accuracy of the first 5 epochs :
    1. Epoch 1/40:loss: 0.2007 - accuracy: 0.9396 - val_loss: 0.0568 - val_accuracy: 0.9834
    2. Epoch 2/40:loss: 0.0585 - accuracy: 0.9821 - val_loss: 0.0425 - val_accuracy: 0.9859

    3.Epoch 3/40:loss: 0.0410 - accuracy: 0.9870 - val_loss: 0.0349 - val_accuracy: 0.9887

    4. Epoch 4/40:loss: 0.0307 - accuracy: 0.9901 - val_loss: 0.0353 - val_accuracy: 0.9869
    5. Epoch 5/40:loss: 0.0250 - accuracy: 0.9920 - val_loss: 0.0359 - val_accuracy: 0.9891

2. Accuracy of the final epoch: 99.03

3. Layers:

- 2 CNN layers with a relu activation function and followed by maxpooling 2D with kernel size 2*2 and strides 2*2
- Flattening layer
- 2 Fully connected layers with relu activation functions
- Another fully connected layer with softmax activation function

4- average train time : [5.872049570083618, 3.538182497024536, 3.7079007625579834, 3.6989691257476807, 3.4229421615600586, 3.741835832595825, 3.7338030338287354, 3.412816286087036, 3.443467378616333, 3.4417223930358887, 3.822333574295044, 3.717013120651245]

5. Number of parameters : 123,690
6. Used optimizer : adam

Adedelta optimizer:

```
model1 =Sequential()
# testing learning rate = 0.1
# 2 CNN and 3 FC
# batch size = 64 , # activation function = relu ,  Adadelta optimizer
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
)))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))
model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(MaxPooling2D((2, 2),strides=(2,2)))

model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(32, activation='relu'))
model1.add(Dense(10, activation='softmax'))

sgd = SGD(learning_rate=0.1, momentum=0.0)

model1.compile(optimizer= 'adadelta',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

time_callback = TimeHistory()
history = model1.fit(train_X, train_labels, epochs=40,
                     validation_data=(test_X, test_labels),shuffle = Tru
e,batch_size = 64 , callbacks =[time_callback,earlystopping])
times = time_callback.times
test_loss, test_acc = model1.evaluate(test_X,  test_labels, verbose=1)
```

1. Accuracy : 90.65
2. Epoch 1/40:loss: 2.2913 - accuracy: 0.1049 - val_loss: 2.2710 - val_accuracy: 0.1155
3. Epoch 2/40:loss: 2.2543 - accuracy: 0.1389 - val_loss: 2.2356 - val_accuracy: 0.1570
4. Epoch 3/40:loss: 2.2156 - accuracy: 0.1815 - val_loss: 2.1923 - val_accuracy: 0.2108
5. Epoch 4/40:loss: 2.1684 - accuracy: 0.2344 - val_loss: 2.1422 - val_accuracy: 0.2428
6. Epoch 5/40:loss: 2.1142 - accuracy: 0.2675 - val_loss: 2.0842 - val_accuracy: 0.2810
7. Optimizer : adadelta

FINAL OBSERVATION:

Adam optimizer gives higher accuracy than adadelta and sgd , accuracy is = 99.03 , so will keep using the model with adam optimizer .

ADD A DROPOUT LAYER:

We try using a dropout layer at 2 different places and 2 different rates, we will have 4 probabilities , 2 of them will contain the rates 50% and 75% after the convolution layers once and the another after the 2 fully connected layers and before the final one , it went that way :

Position of dropout layer	Drop rate	Accuracy
After the 2 conv2D layers	50%	99.30
Before the last FC layer	50%	99.02
After the 2 conv2D layers	75%	99.42
Before the last FC layer	75%	98.73

Final observation:

After testing all possibilities we can see that the highest accuracy we can get is 99.42 with drop rate = 75% after the fully connected layer.