Lab 11 : Exploration of Convolutional Neural Networks Design

Name : veena tgs
Roll No : 225229145

Import Dataset:

In [2]:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
```

In [3]:

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 9s 1us/step
```

In [4]:

```
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0
```

In [5]:

```
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)
```

Step 1 : Number of filters:

In [6]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense
from tensorflow.keras.optimizers import Adam
import time
```

In [39]:

```
def model(n):
  model = Sequential()
  model.add(Conv2D(filters=n, kernel_size=(3, 3), activation='relu', input_shape=(28,28,
  model.add(Flatten())
  model.add(Dense(10,activation = 'softmax'))
  return model
```

In [8]:

```
model4 = model(4)
```

In [9]:

```
model4.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [10]:

```
num_params = model4.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 27,090
```

In [11]:

```
start_time = time.time()
model4.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.3281 -
accuracy: 0.9072 - val_loss: 0.2199 - val_accuracy: 0.9403
Epoch 2/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1670 -
accuracy: 0.9539 - val_loss: 0.1234 - val_accuracy: 0.9635
Epoch 3/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1107 -
accuracy: 0.9689 - val_loss: 0.0963 - val_accuracy: 0.9703
Epoch 4/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0835 -
accuracy: 0.9764 - val_loss: 0.0783 - val_accuracy: 0.9769
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0686 -
accuracy: 0.9792 - val_loss: 0.0775 - val_accuracy: 0.9764
Training time: 28.86 seconds
```

In [12]:

```
model32 = model(32)
```

In [13]:

```
model32.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [14]:

```
num_params = model32.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 216,650
```

In [15]:

```
start_time = time.time()
model32.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.1746
- accuracy: 0.9488 - val_loss: 0.0807 - val_accuracy: 0.9759
Epoch 2/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0682
- accuracy: 0.9802 - val_loss: 0.0631 - val_accuracy: 0.9784
Epoch 3/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0507
- accuracy: 0.9844 - val_loss: 0.0659 - val_accuracy: 0.9810
Epoch 4/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0377
- accuracy: 0.9887 - val_loss: 0.0636 - val_accuracy: 0.9800
Epoch 5/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0295
- accuracy: 0.9904 - val_loss: 0.0745 - val_accuracy: 0.9773
Training time: 62.41 seconds
```

In [16]:

```
model128 = model(128)
```

In [17]:

```
model128.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [18]:

```
num_params = model128.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 866,570
```

In [19]:

```
start_time = time.time()
model128.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test)
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 59s 31ms/step - loss: 0.1425
- accuracy: 0.9578 - val_loss: 0.0733 - val_accuracy: 0.9767
Epoch 2/5
1875/1875 [==============================] - 61s 32ms/step - loss: 0.0566
- accuracy: 0.9826 - val_loss: 0.0713 - val_accuracy: 0.9770
Epoch 3/5
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0372
- accuracy: 0.9882 - val_loss: 0.0620 - val_accuracy: 0.9811
Epoch 4/5
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0244
- accuracy: 0.9921 - val_loss: 0.0721 - val_accuracy: 0.9791
Epoch 5/5
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0168
- accuracy: 0.9946 - val_loss: 0.0674 - val_accuracy: 0.9817
Training time: 310.12 seconds
```

In [51]:

```
def model(n):
  model = Sequential()
  for i in range(n):
        model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=
  model.add(Flatten())
  model.add(Dense(10,activation = 'softmax'))
  return model
```

2 Number of Layers:

In [52]:

```
modellayer2 = model(2)
```

In [53]:

```
modellayer2.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accura
```

In [54]:

```
num_params = modellayer2.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 193,898
```

In [59]:

```
start_time = time.time()
modellayer2.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_te
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 59s 31ms/step - loss: 0.0816
- accuracy: 0.9761 - val_loss: 0.0628 - val_accuracy: 0.9812
Epoch 2/5
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0449
- accuracy: 0.9861 - val_loss: 0.0483 - val_accuracy: 0.9850
Epoch 3/5
1875/1875 [==============================] - 54s 29ms/step - loss: 0.0298
- accuracy: 0.9902 - val_loss: 0.0422 - val_accuracy: 0.9875
Epoch 4/5
1875/1875 [==============================] - 58s 31ms/step - loss: 0.0206
- accuracy: 0.9934 - val_loss: 0.0460 - val_accuracy: 0.9869
Epoch 5/5
1875/1875 [==============================] - 56s 30ms/step - loss: 0.0140
- accuracy: 0.9954 - val_loss: 0.0470 - val_accuracy: 0.9879
Training time: 291.93 seconds
```

3 Number of Layers :

In [56]:

```
modellayer3 = model(3)
```

In [57]:

```
modellayer3.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accurac
```

In [58]:

```
num_params = modellayer3.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 173,706
```

In [30]:

```
start_time = time.time()
modellayer3.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_te
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.1795
- accuracy: 0.9483 - val_loss: 0.0767 - val_accuracy: 0.9758
Epoch 2/5
1875/1875 [==============================] - 17s 9ms/step - loss: 0.0710
- accuracy: 0.9777 - val_loss: 0.0663 - val_accuracy: 0.9791
Epoch 3/5
1875/1875 [==============================] - 12s 7ms/step - loss: 0.0497
- accuracy: 0.9847 - val_loss: 0.0637 - val_accuracy: 0.9794
Epoch 4/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.0376
- accuracy: 0.9884 - val_loss: 0.0707 - val_accuracy: 0.9806
Epoch 5/5
1875/1875 [==============================] - 12s 7ms/step - loss: 0.0297
- accuracy: 0.9908 - val_loss: 0.0628 - val_accuracy: 0.9815
Training time: 69.41 seconds
```

4 Number of layers :

In [60]:

```
modellayer4 = model(4)
```

In [61]:

```
modellayer4.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accura
```

In [62]:

```
num_params = modellayer4.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 156,074
```

In [34]:

```
start_time = time.time()
modellayer4.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_te
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.1714
- accuracy: 0.9503 - val_loss: 0.0776 - val_accuracy: 0.9765
Epoch 2/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.0663
- accuracy: 0.9801 - val_loss: 0.0667 - val_accuracy: 0.9791
Epoch 3/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0467
- accuracy: 0.9859 - val_loss: 0.0589 - val_accuracy: 0.9819
Epoch 4/5
1875/1875 [==============================] - 14s 8ms/step - loss: 0.0345
- accuracy: 0.9892 - val_loss: 0.0625 - val_accuracy: 0.9803
Epoch 5/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0276
- accuracy: 0.9914 - val_loss: 0.0673 - val_accuracy: 0.9792
Training time: 73.19 seconds
```

Step 3 : Size of Filters :

In [63]:

```
modelsize3 = Sequential([
       Conv2D(16, (3, 3), activation='relu', input_shape=(28, 28, 1)),
       Conv2D(16, (3, 3), activation='relu'),
       Flatten(),
       Dense(10, activation='softmax')])
```

In [64]:

```
modelsize3.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accurac
```

In [65]:

```
num_params = modelsize3.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 94,650
```

In [66]:

```
start_time = time.time()
modelsize3.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_tes
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 37s 20ms/step - loss: 0.1536
- accuracy: 0.9545 - val_loss: 0.0664 - val_accuracy: 0.9790
Epoch 2/5
1875/1875 [==============================] - 29s 15ms/step - loss: 0.0545
- accuracy: 0.9830 - val_loss: 0.0540 - val_accuracy: 0.9820
Epoch 3/5
1875/1875 [==============================] - 28s 15ms/step - loss: 0.0365
- accuracy: 0.9888 - val_loss: 0.0447 - val_accuracy: 0.9857
Epoch 4/5
1875/1875 [==============================] - 28s 15ms/step - loss: 0.0261
- accuracy: 0.9916 - val_loss: 0.0445 - val_accuracy: 0.9864
Epoch 5/5
1875/1875 [==============================] - 25s 14ms/step - loss: 0.0193
- accuracy: 0.9942 - val_loss: 0.0422 - val_accuracy: 0.9874
Training time: 147.41 seconds
```

In [67]:

```
modelsize5 = Sequential([
       Conv2D(16, (5, 5), activation='relu', input_shape=(28, 28, 1)),
       Conv2D(16, (5, 5), activation='relu'),
       Flatten(),
       Dense(10, activation='softmax')])
```

In [68]:

```
modelsize5.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accurac
```

In [69]:

```
num_params = modelsize5.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 70,842
```

In [70]:

```
start_time = time.time()
modelsize5.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_tes
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.1350
- accuracy: 0.9594 - val_loss: 0.0542 - val_accuracy: 0.9830
Epoch 2/5
1875/1875 [==============================] - 39s 21ms/step - loss: 0.0494
- accuracy: 0.9845 - val_loss: 0.0382 - val_accuracy: 0.9864
Epoch 3/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0335
- accuracy: 0.9897 - val_loss: 0.0410 - val_accuracy: 0.9857
Epoch 4/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0253
- accuracy: 0.9922 - val_loss: 0.0340 - val_accuracy: 0.9881
Epoch 5/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0182
- accuracy: 0.9940 - val_loss: 0.0399 - val_accuracy: 0.9880
Training time: 203.27 seconds
```

In [71]:

```
modelsize7 = Sequential([
       Conv2D(16, (7, 7), activation='relu', input_shape=(28, 28, 1)),
       Conv2D(16, (7, 7), activation='relu'),
       Flatten(),
       Dense(10, activation='softmax')])
```

In [72]:

```
modelsize7.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accurac
```

In [73]:

```
num_params = modelsize7.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 54,330
```

In [74]:

```
start_time = time.time()
modelsize7.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_tes
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 45s 24ms/step - loss: 0.1360
- accuracy: 0.9588 - val_loss: 0.0487 - val_accuracy: 0.9842
Epoch 2/5
1875/1875 [==============================] - 45s 24ms/step - loss: 0.0497
- accuracy: 0.9846 - val_loss: 0.0434 - val_accuracy: 0.9863
Epoch 3/5
1875/1875 [==============================] - 49s 26ms/step - loss: 0.0352
- accuracy: 0.9889 - val_loss: 0.0376 - val_accuracy: 0.9882
Epoch 4/5
1875/1875 [==============================] - 51s 27ms/step - loss: 0.0284
- accuracy: 0.9909 - val_loss: 0.0384 - val_accuracy: 0.9881
Epoch 5/5
1875/1875 [==============================] - 48s 26ms/step - loss: 0.0220
- accuracy: 0.9930 - val_loss: 0.0362 - val_accuracy: 0.9894
Training time: 237.42 seconds
```

Step 4 : Activation funciton :

In [78]:

```
def model(n,act):
    model = Sequential()
    for i in range(n):
        model.add(Conv2D(filters=16, kernel_size=(3, 3), activation=act, input_shape=(28
        model.add(Flatten())
        model.add(Dense(10,activation = 'softmax'))
        return model
```

In [79]:

```
modeltanh = model(2, 'tanh')
```

In [80]:

```
modeltanh.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy
```

In [81]:

```
modeltanh.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_tes
```

```
Epoch 1/10
1875/1875 [==============================] - 9s 4ms/step - loss: 0.3488 -
accuracy: 0.8992 - val_loss: 0.2904 - val_accuracy: 0.9183
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2922 -
accuracy: 0.9183 - val_loss: 0.2778 - val_accuracy: 0.9212
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2746 -
accuracy: 0.9227 - val_loss: 0.2692 - val_accuracy: 0.9282
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2506 -
accuracy: 0.9290 - val_loss: 0.2598 - val_accuracy: 0.9258
Epoch 5/10
1875/1875 [==============================] - 8s 5ms/step - loss: 0.2308 -
accuracy: 0.9350 - val_loss: 0.2419 - val_accuracy: 0.9346
Epoch 6/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2131 -
accuracy: 0.9395 - val_loss: 0.2328 - val_accuracy: 0.9370
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1996 -
accuracy: 0.9429 - val_loss: 0.2270 - val_accuracy: 0.9413
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1857 -
accuracy: 0.9461 - val_loss: 0.2161 - val_accuracy: 0.9413
Epoch 9/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1695 -
accuracy: 0.9515 - val_loss: 0.2047 - val_accuracy: 0.9440
Epoch 10/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1517 -
accuracy: 0.9557 - val_loss: 0.1861 - val_accuracy: 0.9502
```

Out[81]:

```
<keras.src.callbacks.History at 0x22ea5a5d290>
```

In [82]:

```
score = modeltanh.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.18610134720802307
Test accuracy: 0.9502000212669373
```

In [83]:

```
modelrelu = model(2, 'relu')
```

In [85]:

```
modelrelu.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy
```

In [86]:

```
modelrelu.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_tes
```

```
Epoch 1/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2099
- accuracy: 0.9403 - val_loss: 0.0933 - val_accuracy: 0.9712
Epoch 2/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0839
- accuracy: 0.9756 - val_loss: 0.0777 - val_accuracy: 0.9761
Epoch 3/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0601 -
accuracy: 0.9819 - val_loss: 0.0653 - val_accuracy: 0.9793
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0464 -
accuracy: 0.9860 - val_loss: 0.0707 - val_accuracy: 0.9775
Epoch 5/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0362 -
accuracy: 0.9891 - val_loss: 0.0740 - val_accuracy: 0.9778
Epoch 6/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0299 -
accuracy: 0.9903 - val_loss: 0.0699 - val_accuracy: 0.9802
Epoch 7/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0240 -
accuracy: 0.9927 - val_loss: 0.0689 - val_accuracy: 0.9809
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0186 -
accuracy: 0.9943 - val_loss: 0.0693 - val_accuracy: 0.9817
Epoch 9/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0149
- accuracy: 0.9954 - val_loss: 0.0750 - val_accuracy: 0.9808
Epoch 10/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0127 -
accuracy: 0.9959 - val_loss: 0.0817 - val_accuracy: 0.9792
```

Out[86]:

```
<keras.src.callbacks.History at 0x22ed275e6d0>
```

In [87]:

```
score = modelrelu.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.08171696215867996
Test accuracy: 0.979200005531311
```

Step 5 : Filter Size combinations :

In [94]:

```
model2 = Sequential()
model2.add(Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_shape=(28,28,
model2.add(Conv2D(filters=16, kernel_size=(5,5), activation='relu', input_shape=(28,28,1
model2.add(Flatten())
model2.add(Dense(10,activation = 'softmax'))
```

In [95]:

```
model2.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [97]:

```
model2.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0652
- accuracy: 0.9806 - val_loss: 0.0473 - val_accuracy: 0.9845
Epoch 2/5
1875/1875 [==============================] - 46s 24ms/step - loss: 0.0389
- accuracy: 0.9882 - val_loss: 0.0422 - val_accuracy: 0.9875
Epoch 3/5
1875/1875 [==============================] - 49s 26ms/step - loss: 0.0277
- accuracy: 0.9911 - val_loss: 0.0406 - val_accuracy: 0.9882
Epoch 4/5
1875/1875 [==============================] - 40s 22ms/step - loss: 0.0202
- accuracy: 0.9938 - val_loss: 0.0360 - val_accuracy: 0.9889
Epoch 5/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0136
- accuracy: 0.9954 - val_loss: 0.0470 - val_accuracy: 0.9885
```

Out[97]:

```
<keras.src.callbacks.History at 0x22ec0273450>
```

Step 7 : Influence of Striding :

In [110]:

```
model3 = Sequential()
model3.add(Conv2D(filters=32, kernel_size=(3, 3),strides=(2,2), activation='relu', input
model3.add(Conv2D(filters=32, kernel_size=(5,5),strides=(2,2), activation='relu', input_
model3.add(Flatten())
model3.add(Dense(10,activation = 'softmax'))
```

In [111]:

```
model3.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [112]:

```
num_params = model3.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 33,962
```

In [113]:

```
start_time = time.time()
model3.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2145
- accuracy: 0.9366 - val_loss: 0.0798 - val_accuracy: 0.9749
Epoch 2/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0734
- accuracy: 0.9777 - val_loss: 0.0525 - val_accuracy: 0.9833
Epoch 3/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0523 -
accuracy: 0.9841 - val_loss: 0.0456 - val_accuracy: 0.9851
Epoch 4/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0398 -
accuracy: 0.9876 - val_loss: 0.0412 - val_accuracy: 0.9867
Epoch 5/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0324 -
accuracy: 0.9902 - val_loss: 0.0437 - val_accuracy: 0.9859
Training time: 47.48 seconds
```

In [115]:

```
score = model3.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.04365567862987518
Test accuracy: 0.9858999848365784
```

In [116]:

```
model4 = Sequential()
model4.add(Conv2D(filters=32, kernel_size=(3, 3),strides=(3,3), activation='relu', input
model4.add(Conv2D(filters=32, kernel_size=(5,5),strides=(3,3), activation='relu', input_
model4.add(Flatten())
model4.add(Dense(10,activation = 'softmax'))
```

In [118]:

```
model4.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [119]:

```
num_params = model4.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 27,242
```

In [121]:

```
start_time = time.time()
model4.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2995 -
accuracy: 0.9131 - val_loss: 0.1173 - val_accuracy: 0.9641
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1178 -
accuracy: 0.9647 - val_loss: 0.0966 - val_accuracy: 0.9704
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0872 -
accuracy: 0.9740 - val_loss: 0.0813 - val_accuracy: 0.9754
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0712 -
accuracy: 0.9778 - val_loss: 0.0738 - val_accuracy: 0.9773
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0605 -
accuracy: 0.9819 - val_loss: 0.0642 - val_accuracy: 0.9787
Training time: 16.49 seconds
```

In [122]:

```
score = model4.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.06423342972993851
Test accuracy: 0.9786999821662903
```

In [123]:

```
model5 = Sequential()
model5.add(Conv2D(filters=32, kernel_size=(5,5),strides=(2,2), activation='relu', input_
model5.add(Conv2D(filters=32, kernel_size=(5,5),strides=(2,2), activation='relu', input_
model5.add(Flatten())
model5.add(Dense(10,activation = 'softmax'))
```

In [124]:

```
model5.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [125]:

```
num_params = model5.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 31,594
```

In [127]:

```
start_time = time.time()
model5.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0243 -
accuracy: 0.9923 - val_loss: 0.0438 - val_accuracy: 0.9872
Epoch 2/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0207 -
accuracy: 0.9933 - val_loss: 0.0420 - val_accuracy: 0.9880
Epoch 3/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0168 -
accuracy: 0.9947 - val_loss: 0.0460 - val_accuracy: 0.9877
Epoch 4/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0137 -
accuracy: 0.9955 - val_loss: 0.0523 - val_accuracy: 0.9856
Epoch 5/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0123 -
accuracy: 0.9960 - val_loss: 0.0465 - val_accuracy: 0.9883
Training time: 37.95 seconds
```

In [128]:

```
model6 = Sequential()
model6.add(Conv2D(filters=32, kernel_size=(7,7),strides=(2,2), activation='relu', input_
model6.add(Conv2D(filters=32, kernel_size=(7,7),strides=(2,2), activation='relu', input_
model6.add(Flatten())
model6.add(Dense(10,activation = 'softmax'))
```

In [130]:

```
model6.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [131]:

```
num_params = model6.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 54,698
```

In [132]:

```
start_time = time.time()
model6.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2021 -
accuracy: 0.9392 - val_loss: 0.0734 - val_accuracy: 0.9774
Epoch 2/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0690 -
accuracy: 0.9794 - val_loss: 0.0480 - val_accuracy: 0.9859
Epoch 3/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0480 -
accuracy: 0.9853 - val_loss: 0.0408 - val_accuracy: 0.9861
Epoch 4/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0370 -
accuracy: 0.9885 - val_loss: 0.0438 - val_accuracy: 0.9873
Epoch 5/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0272 -
accuracy: 0.9913 - val_loss: 0.0495 - val_accuracy: 0.9839
Training time: 37.31 seconds
```

Step 8 : Influence of Padding :

In [133]:

```
model7 = Sequential()
model7.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',padding='same', input
model7.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',padding='same', input
model7.add(Flatten())
model7.add(Dense(10,activation = 'softmax'))
```

In [134]:

```
model7.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [135]:

```
num_params = model7.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 277,354
```

In [136]:

```
start_time = time.time()
model7.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 97s 52ms/step - loss: 0.1101
- accuracy: 0.9667 - val_loss: 0.0465 - val_accuracy: 0.9851
Epoch 2/5
1875/1875 [==============================] - 98s 52ms/step - loss: 0.0403
- accuracy: 0.9874 - val_loss: 0.0379 - val_accuracy: 0.9877
Epoch 3/5
1875/1875 [==============================] - 99s 53ms/step - loss: 0.0267
- accuracy: 0.9916 - val_loss: 0.0348 - val_accuracy: 0.9898
Epoch 4/5
1875/1875 [==============================] - 97s 52ms/step - loss: 0.0188
- accuracy: 0.9941 - val_loss: 0.0402 - val_accuracy: 0.9878
Epoch 5/5
1875/1875 [==============================] - 108s 57ms/step - loss: 0.013
7 - accuracy: 0.9957 - val_loss: 0.0426 - val_accuracy: 0.9872
Training time: 498.68 seconds
```

In [137]:

```
model8 = Sequential()
model8.add(Conv2D(filters=32, kernel_size=(7,7), activation='relu',padding='same', input
model8.add(Conv2D(filters=32, kernel_size=(7,7), activation='relu',padding='same', input
model8.add(Flatten())
model8.add(Dense(10,activation = 'softmax'))
```

In [138]:

```
model8.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [139]:

```
num_params = model8.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 302,698
```

In [140]:

```
start_time = time.time()
model8.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 225s 120ms/step - loss: 0.11
02 - accuracy: 0.9669 - val_loss: 0.0483 - val_accuracy: 0.9837
Epoch 2/5
1875/1875 [==============================] - 222s 118ms/step - loss: 0.04
13 - accuracy: 0.9873 - val_loss: 0.0448 - val_accuracy: 0.9857
Epoch 3/5
1875/1875 [==============================] - 221s 118ms/step - loss: 0.02
81 - accuracy: 0.9916 - val_loss: 0.0370 - val_accuracy: 0.9892
Epoch 4/5
1875/1875 [==============================] - 230s 123ms/step - loss: 0.01
92 - accuracy: 0.9942 - val_loss: 0.0336 - val_accuracy: 0.9903
Epoch 5/5
1875/1875 [==============================] - 228s 122ms/step - loss: 0.01
53 - accuracy: 0.9954 - val_loss: 0.0376 - val_accuracy: 0.9897
Training time: 1126.27 seconds
```

Step 9 : Influence of Pooling :

In [142]:

```
import tensorflow as tf
```

In [143]:

```
model8 = Sequential()
model8.add(Conv2D(filters=32, kernel_size=(7,7), activation='relu',input_shape=(28,28,1)
model8.add(Conv2D(filters=32, kernel_size=(7,7), activation='relu',input_shape=(28,28,1)
tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
model8.add(Flatten())
model8.add(Dense(10,activation = 'softmax'))
```

In [144]:

```
model8.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [145]:

```
num_params = model8.count_params()
print("Number of parameters: {:,}".format(num_params))
```

```
Number of parameters: 133,738
```

In [ ]:

```
start_time = time.time()
model8.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
end_time = time.time()
training_time = end_time - start_time
print("Training time: {:.2f} seconds".format(training_time))
```

```
Epoch 1/5
1875/1875 [==============================] - 100s 53ms/step - loss: 0.112
8 - accuracy: 0.9663 - val_loss: 0.0527 - val_accuracy: 0.9830
Epoch 2/5
1875/1875 [==============================] - 99s 53ms/step - loss: 0.0418
- accuracy: 0.9867 - val_loss: 0.0359 - val_accuracy: 0.9890
Epoch 3/5
1875/1875 [==============================] - 100s 53ms/step - loss: 0.030
4 - accuracy: 0.9906 - val_loss: 0.0371 - val_accuracy: 0.9886
Epoch 4/5
1529/1875 [=======================>......] - ETA: 19s - loss: 0.0227 - ac
curacy: 0.9926
```