Name : Berchmans Kevin S

RollNo : 215229107

PDL Lab11. Exploration of Convolutional Neural Networks Design

In [*]:

```
from __future__ import print_function
import keras
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
from keras.utils import np_utils
import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

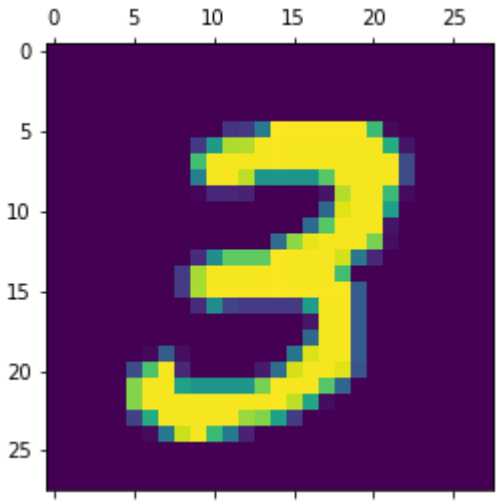
```
(x_train,y_train),(x_test,y_test)=tf.keras.datasets.mnist.load_data()
```

In [7]:

```
plt.matshow(x_train[7])
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x203a8a7e610>
```



In [4]:

```
X_train = x_train.astype('float32')/255
X_test = x_test.astype('float32')/255
```

In [5]:

```
X_train.shape
```

Out[5]:

```
(60000, 28, 28)
```

In [6]:

```
y_train.shape
```

Out[6]:

```
(60000,)
```

1.Number of Filters:

In [3]:

```
def mod(n):
    model = Sequential()
    model.add(Conv2D(filters=n, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1
    model.add(Flatten())
    model.add(Dense(10,activation = 'softmax'))
    return model
```

In [ ]:

```
model2=mod(4)
model2.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model2.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 15s 8ms/step - loss: 27.3046 -
accuracy: 0.0997
Epoch 2/5
1875/1875 [==============================] - 15s 8ms/step - loss: 27.3046 -
accuracy: 0.0978
Epoch 3/5
1875/1875 [==============================] - 15s 8ms/step - loss: 27.3046 -
accuracy: 0.0978
Epoch 4/5
1875/1875 [==============================] - 19s 10ms/step - loss: 27.3046 -
accuracy: 0.0999
Epoch 5/5
1875/1875 [==============================] - 15s 8ms/step - loss: 27.3046 -
accuracy: 0.0974
```

Out[14]:

```
<keras.callbacks.History at 0x7f5eb49885d0>
```

In [ ]:

```
model2=mod(32)
model2.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model2.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 30s 16ms/step - loss: 27.3046 -
accuracy: 0.0980
Epoch 2/5
1875/1875 [==============================] - 30s 16ms/step - loss: 27.3046 -
accuracy: 0.1047
Epoch 3/5
1875/1875 [==============================] - 31s 16ms/step - loss: 27.3046 -
accuracy: 0.1040
Epoch 4/5
1875/1875 [==============================] - 30s 16ms/step - loss: 27.3046 -
accuracy: 0.1025
Epoch 5/5
1875/1875 [==============================] - 28s 15ms/step - loss: 27.3046 -
accuracy: 0.1032
```

Out[15]:

```
<keras.callbacks.History at 0x7f5eb1855f10>
```

In [ ]:

```
model2=mod(128)
model2.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model2.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 69s 37ms/step - loss: 27.3046 -
accuracy: 0.0997
Epoch 2/5
1875/1875 [==============================] - 72s 38ms/step - loss: 27.3046 -
accuracy: 0.1009
Epoch 3/5
1875/1875 [==============================] - 62s 33ms/step - loss: 27.3046 -
accuracy: 0.0999
Epoch 4/5
1875/1875 [==============================] - 76s 41ms/step - loss: 27.3046 -
accuracy: 0.1048
Epoch 5/5
1875/1875 [==============================] - 62s 33ms/step - loss: 27.3046 -
accuracy: 0.1053
```

Out[16]:

```
<keras.callbacks.History at 0x7f5eb4861c50>
```

2. Number of layers:

In [ ]:

```
def mod(n):
    model = Sequential()
    for i in range(n):
        model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(28
        model.add(Flatten())
        model.add(Dense(10,activation = 'softmax'))
        return model
```

In [ ]:

```
model2=mod(2)
model2.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model2.fit(X_train,y_train,epochs=5,batch_size=64)
```

```
Epoch 1/5
938/938 [==============================] - 106s 112ms/step - loss: 27.3045 -
accuracy: 0.1092
Epoch 2/5
938/938 [==============================] - 91s 97ms/step - loss: 27.3045 - a
ccuracy: 0.1009
Epoch 3/5
938/938 [==============================] - 92s 98ms/step - loss: 27.3045 - a
ccuracy: 0.0976
Epoch 4/5
938/938 [==============================] - 90s 96ms/step - loss: 27.3045 - a
ccuracy: 0.0975
Epoch 5/5
938/938 [==============================] - 98s 104ms/step - loss: 27.3045 -
accuracy: 0.0971
```

Out[21]:

```
<keras.callbacks.History at 0x7f5eb43e8550>
```

In [ ]:

```
model2=mod(3)
model2.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model2.fit(X_train,y_train,epochs=5,batch_size=64)
```

```
Epoch 1/5
938/938 [==============================] - 247s 167ms/step - loss: 27.3045 -
accuracy: 0.1036
Epoch 2/5
938/938 [==============================] - 150s 160ms/step - loss: 27.3045 -
accuracy: 0.1034
Epoch 3/5
938/938 [==============================] - 149s 159ms/step - loss: 27.3045 -
accuracy: 0.1021
Epoch 4/5
938/938 [==============================] - 148s 158ms/step - loss: 27.3045 -
accuracy: 0.1019
Epoch 5/5
938/938 [==============================] - 148s 158ms/step - loss: 27.3045 -
accuracy: 0.1016
```

Out[22]:

```
<keras.callbacks.History at 0x7f5eb42b0bd0>
```

In [ ]:

```
model2=mod(4)
model2.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model2.fit(X_train,y_train,epochs=5,batch_size=64)
```

```
Epoch 1/5
938/938 [==============================] - 197s 209ms/step - loss: 27.3045 -
accuracy: 0.1076
Epoch 2/5
938/938 [==============================] - 202s 215ms/step - loss: 27.3045 -
accuracy: 0.1017
Epoch 3/5
938/938 [==============================] - 209s 223ms/step - loss: 27.3045 -
accuracy: 0.0968
Epoch 4/5
938/938 [==============================] - 197s 210ms/step - loss: 27.3045 -
accuracy: 0.0964
Epoch 5/5
938/938 [==============================] - 196s 209ms/step - loss: 27.3045 -
accuracy: 0.0972
```

Out[23]:

```
<keras.callbacks.History at 0x7f5eb462c290>
```

3.Size of filters:

In [ ]:

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

model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 25s 13ms/step - loss: 27.3046 -
accuracy: 0.0972
Epoch 2/5
1875/1875 [==============================] - 24s 13ms/step - loss: 27.3046 -
accuracy: 0.0940
Epoch 3/5
1875/1875 [==============================] - 22s 12ms/step - loss: 27.3046 -
accuracy: 0.0968
Epoch 4/5
1875/1875 [==============================] - 22s 12ms/step - loss: 27.3046 -
accuracy: 0.0989
Epoch 5/5
1875/1875 [==============================] - 22s 12ms/step - loss: 27.3046 -
accuracy: 0.0986
```

Out[26]:

```
<keras.callbacks.History at 0x7f5eb46f3650>
```

In [ ]:

```
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=(7,7), activation='relu', input_shape=(28,28,1)))
model.add(Flatten())
model.add(Dense(10,activation = 'softmax'))

model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 25s 13ms/step - loss: 27.3046 -
accuracy: 0.1009
Epoch 2/5
1875/1875 [==============================] - 24s 13ms/step - loss: 27.3046 -
accuracy: 0.1000
Epoch 3/5
1875/1875 [==============================] - 24s 13ms/step - loss: 27.3046 -
accuracy: 0.0965
Epoch 4/5
1875/1875 [==============================] - 24s 13ms/step - loss: 27.3046 -
accuracy: 0.0975
Epoch 5/5
1875/1875 [==============================] - 24s 13ms/step - loss: 27.3046 -
accuracy: 0.0993
```

Out[27]:

```
<keras.callbacks.History at 0x7f5eb8dd5d50>
```

4. Activation Function:

In [9]:

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

In [11]:

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

model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 57s 30ms/step - loss: 27.3046 -
accuracy: 0.0969
Epoch 2/5
1875/1875 [==============================] - 69s 37ms/step - loss: 27.3046 -
accuracy: 0.0965
Epoch 3/5
1875/1875 [==============================] - 59s 32ms/step - loss: 27.3046 -
accuracy: 0.0954
Epoch 4/5
1875/1875 [==============================] - 57s 30ms/step - loss: 27.3046 -
accuracy: 0.0974
Epoch 5/5
1875/1875 [==============================] - 58s 31ms/step - loss: 27.3046 -
accuracy: 0.0969
```

Out[11]:

```
<keras.callbacks.History at 0x7f751e26ebd0>
```

In [12]:

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

```
Test loss: 27.251352310180664
Test accuracy: 0.08269999921321869
```

In [13]:

```
model1=mod(2,'relu')

model1.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model1.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 55s 29ms/step - loss: 27.3045 -
accuracy: 0.1027
Epoch 2/5
1875/1875 [==============================] - 55s 29ms/step - loss: 27.3046 -
accuracy: 0.1021
Epoch 3/5
1875/1875 [==============================] - 56s 30ms/step - loss: 27.3046 -
accuracy: 0.1011
Epoch 4/5
1875/1875 [==============================] - 57s 30ms/step - loss: 27.3046 -
accuracy: 0.1027
Epoch 5/5
1875/1875 [==============================] - 64s 34ms/step - loss: 27.3046 -
accuracy: 0.1015
```

Out[13]:

```
<keras.callbacks.History at 0x7f751c13d8d0>
```

In [14]:

```
score = model1.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 27.260662078857422
Test accuracy: 0.10819999873638153
```

5. Filter size combination:

In [16]:

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

model2.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model2.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 107s 57ms/step - loss: 27.3046
- accuracy: 0.0980
Epoch 2/5
1875/1875 [==============================] - 108s 58ms/step - loss: 27.3046
- accuracy: 0.0992
Epoch 3/5
1875/1875 [==============================] - 114s 61ms/step - loss: 27.3046
- accuracy: 0.1013
Epoch 4/5
1875/1875 [==============================] - 105s 56ms/step - loss: 27.3046
- accuracy: 0.1000
Epoch 5/5
1875/1875 [==============================] - 117s 63ms/step - loss: 27.3046
- accuracy: 0.0993
```

Out[16]:

```
<keras.callbacks.History at 0x7f7518430b90>
```

In [17]:

```
score = model2.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 27.255985260009766
Test accuracy: 0.08250000327825546
```

In [8]:

```
model = Sequential([
    Conv2D(filters=16, kernel_size=(5, 5), activation='tanh', input_shape=(28,28,1)),
    Conv2D(filters=16, kernel_size=(7, 7), activation='tanh'),
    Flatten(),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=5)

model.evaluate(X_test, y_test)
```

```
Epoch 1/5
1875/1875 [==============================] - 131s 69ms/step - loss: 27.3046
- accuracy: 0.0986
Epoch 2/5
1875/1875 [==============================] - 127s 68ms/step - loss: 27.3046
- accuracy: 0.0970
Epoch 3/5
1875/1875 [==============================] - 132s 71ms/step - loss: 27.3045
- accuracy: 0.0994
Epoch 4/5
1875/1875 [==============================] - 124s 66ms/step - loss: 27.3046
- accuracy: 0.0988
Epoch 5/5
1875/1875 [==============================] - 126s 67ms/step - loss: 27.3046
- accuracy: 0.1000
313/313 [==============================] - 7s 20ms/step - loss: 27.2503 - ac
curacy: 0.0760
```

Out[8]:

```
[27.25031280517578, 0.07599999755620956]
```

6. Layer Filter Combinations

In [9]:

```
model = Sequential([
    Conv2D(filters=16, kernel_size=(3, 3), activation='tanh', input_shape=(28,28,1)),
    Conv2D(filters=32, kernel_size=(3, 3), activation='tanh'),
    Flatten(),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=3)

model.evaluate(X_test, y_test)
```

```
Epoch 1/3
1875/1875 [==============================] - 81s 42ms/step - loss: 27.3046 -
accuracy: 0.1028
Epoch 2/3
1875/1875 [==============================] - 80s 43ms/step - loss: 27.3046 -
accuracy: 0.1000
Epoch 3/3
1875/1875 [==============================] - 79s 42ms/step - loss: 27.3046 -
accuracy: 0.1024
313/313 [==============================] - 5s 16ms/step - loss: 27.2503 - ac
curacy: 0.0826
```

Out[9]:

```
[27.25031280517578, 0.08259999752044678]
```

In [10]:

```
model = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), activation='tanh', input_shape=(28,28,1)),
    Conv2D(filters=16, kernel_size=(3, 3), activation='tanh'),
    Flatten(),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=3)

model.evaluate(X_test, y_test)
```

```
Epoch 1/3
1875/1875 [==============================] - 106s 56ms/step - loss: 27.3046
- accuracy: 0.1098
Epoch 2/3
1875/1875 [==============================] - 105s 56ms/step - loss: 27.3046
- accuracy: 0.1127
Epoch 3/3
1875/1875 [==============================] - 106s 56ms/step - loss: 27.3046
- accuracy: 0.1073
313/313 [==============================] - 6s 17ms/step - loss: 27.2503 - ac
curacy: 0.0940
```

Out[10]:

```
[27.25031280517578, 0.09399999678134918]
```

In [ ]:

```
```

7. Influence of Striding:

In [23]:

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

model3.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model3.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 26s 13ms/step - loss: 27.3046 -
accuracy: 0.0943
Epoch 2/5
1875/1875 [==============================] - 27s 15ms/step - loss: 27.3046 -
accuracy: 0.0966
Epoch 3/5
1875/1875 [==============================] - 27s 15ms/step - loss: 27.3046 -
accuracy: 0.0984
Epoch 4/5
1875/1875 [==============================] - 27s 14ms/step - loss: 27.3046 -
accuracy: 0.0999
Epoch 5/5
1875/1875 [==============================] - 29s 16ms/step - loss: 27.3046 -
accuracy: 0.1013
```

Out[23]:

```
<keras.callbacks.History at 0x7f75179e0e10>
```

In [24]:

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

```
Test loss: 27.2541561126709
Test accuracy: 0.10559999942779541
```

In [25]:

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

model4.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model4.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 20s 10ms/step - loss: 27.3046 -
accuracy: 0.0973
Epoch 2/5
1875/1875 [==============================] - 16s 9ms/step - loss: 27.3046 -
accuracy: 0.1002
Epoch 3/5
1875/1875 [==============================] - 20s 11ms/step - loss: 27.3046 -
accuracy: 0.1006
Epoch 4/5
1875/1875 [==============================] - 22s 12ms/step - loss: 27.3046 -
accuracy: 0.0984
Epoch 5/5
1875/1875 [==============================] - 15s 8ms/step - loss: 27.3046 -
accuracy: 0.0993
```

Out[25]:

```
<keras.callbacks.History at 0x7f75140cdb90>
```

In [26]:

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

```
Test loss: 27.255359649658203
Test accuracy: 0.10689999908208847
```

In [27]:

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

model5.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model5.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 19s 10ms/step - loss: 27.3046 -
accuracy: 0.1021
Epoch 2/5
1875/1875 [==============================] - 19s 10ms/step - loss: 27.3046 -
accuracy: 0.1006
Epoch 3/5
1875/1875 [==============================] - 18s 10ms/step - loss: 27.3046 -
accuracy: 0.0995
Epoch 4/5
1875/1875 [==============================] - 18s 10ms/step - loss: 27.3046 -
accuracy: 0.1004
Epoch 5/5
1875/1875 [==============================] - 18s 10ms/step - loss: 27.3046 -
accuracy: 0.1015
```

Out[27]:

```
<keras.callbacks.History at 0x7f751c17bc50>
```

In [28]:

```
score = model5.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 27.2515811920166
Test accuracy: 0.09880000352859497
```

In [29]:

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

model6.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model6.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 23s 12ms/step - loss: 27.3046 -
accuracy: 0.0993
Epoch 2/5
1875/1875 [==============================] - 22s 12ms/step - loss: 27.3046 -
accuracy: 0.0990
Epoch 3/5
1875/1875 [==============================] - 22s 12ms/step - loss: 27.3046 -
accuracy: 0.1002
Epoch 4/5
1875/1875 [==============================] - 22s 11ms/step - loss: 27.3046 -
accuracy: 0.0984
Epoch 5/5
1875/1875 [==============================] - 22s 11ms/step - loss: 27.3046 -
accuracy: 0.0998
```

Out[29]:

```
<keras.callbacks.History at 0x7f75211e17d0>
```

In [30]:

```
score = model6.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 27.250402450561523
Test accuracy: 0.11569999903440475
```

8. Influence of Padding

In [37]:

```
model9 = Sequential()
model9.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',padding='same', input_sh
model9.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',padding='same', input_sh
model9.add(Flatten())
model9.add(Dense(10,activation = 'softmax'))

model9.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model9.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 304s 162ms/step - loss: 27.3046
- accuracy: 0.1017
Epoch 2/5
1875/1875 [==============================] - 304s 162ms/step - loss: 27.3046
- accuracy: 0.1006
Epoch 3/5
1875/1875 [==============================] - 302s 161ms/step - loss: 27.3046
- accuracy: 0.1005
Epoch 4/5
1875/1875 [==============================] - 306s 163ms/step - loss: 27.3046
- accuracy: 0.0988
Epoch 5/5
1875/1875 [==============================] - 303s 162ms/step - loss: 27.3046
- accuracy: 0.0989
```

Out[37]:

```
<keras.callbacks.History at 0x7f751fe3a1d0>
```

In [38]:

```
score = model9.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 27.252286911010742
Test accuracy: 0.11720000207424164
```

9. Influence of Pooling:

In [32]:

```
model7 = Sequential()
model7.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=(28,28,1)))
model7.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=(28,28,1)))
model7.add(MaxPooling2D(pool_size=(2,2)))
model7.add(Flatten())
model7.add(Dense(10,activation = 'softmax'))

model7.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model7.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 168s 89ms/step - loss: 27.3046
- accuracy: 0.0969
Epoch 2/5
1875/1875 [==============================] - 166s 89ms/step - loss: 27.3046
- accuracy: 0.1009
Epoch 3/5
1875/1875 [==============================] - 168s 90ms/step - loss: 27.3046
- accuracy: 0.1014
Epoch 4/5
1875/1875 [==============================] - 167s 89ms/step - loss: 27.3046
- accuracy: 0.1011
Epoch 5/5
1875/1875 [==============================] - 166s 89ms/step - loss: 27.3046
- accuracy: 0.1011
```

Out[32]:

```
<keras.callbacks.History at 0x7f751f32fcd0>
```

In [33]:

```
score = model7.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 27.2547664642334
Test accuracy: 0.07010000199079514
```

In [34]:

```
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)))
model8.add(MaxPooling2D(pool_size=(3,3)))
model8.add(Flatten())
model8.add(Dense(10,activation = 'softmax'))

model8.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model8.fit(X_train,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 218s 116ms/step - loss: 27.3045
- accuracy: 0.0899
Epoch 2/5
1875/1875 [==============================] - 216s 115ms/step - loss: 27.3046
- accuracy: 0.1009
Epoch 3/5
1875/1875 [==============================] - 214s 114ms/step - loss: 27.3046
- accuracy: 0.0999
Epoch 4/5
1875/1875 [==============================] - 214s 114ms/step - loss: 27.3046
- accuracy: 0.1007
Epoch 5/5
1875/1875 [==============================] - 214s 114ms/step - loss: 27.3046
- accuracy: 0.0974
```

Out[34]:

```
<keras.callbacks.History at 0x7f75208601d0>
```

In [35]:

```
score = model8.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 27.25075340270996
Test accuracy: 0.09130000323057175
```

In [39]: