```
import numpy as np
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras import backend as K
from keras.preprocessing import image
from keras.applications.mobilenet import MobileNet
from keras.applications.vgg16 import preprocess_input, decode_predictions
from keras.models import Model
import timeit

import warnings
warnings.filterwarnings('ignore')


batch_size = 128
num_classes = 10
epochs = 2

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
11490434/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

```
model = Sequential()
model.add(Conv2D(8, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 8)         80        
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 13, 13, 8)        0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 11, 11, 16)        1168      
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 5, 5, 16)         0         
 2D)                                                             
                                                                 
 dropout_4 (Dropout)         (None, 5, 5, 16)          0         
                                                                 
 flatten_2 (Flatten)         (None, 400)               0         
                                                                 
 dense_4 (Dense)             (None, 32)                12832     
                                                                 
 dropout_5 (Dropout)         (None, 32)                0         
                                                                 
 dense_5 (Dense)             (None, 10)                330       
                                                                 
=================================================================
Total params: 14,410
Trainable params: 14,410
Non-trainable params: 0
_________________________________________________________________
```

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

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

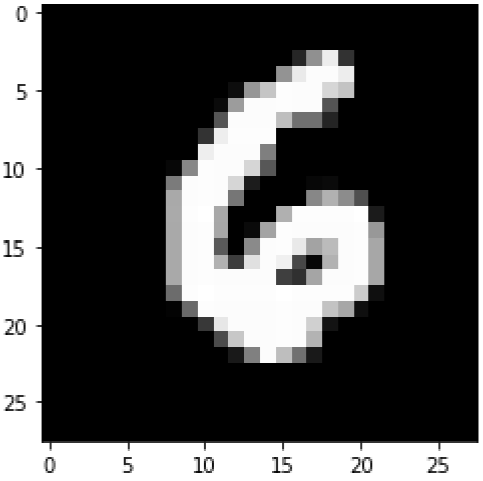
```
Epoch 1/2
469/469 [==============================] - 31s 63ms/step - loss: 2.3205 - accur
Epoch 2/2
469/469 [==============================] - 24s 50ms/step - loss: 2.3172 - accur
<keras.callbacks.History at 0x7f1866f951f0>
```

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

```
Test loss: 2.30600905418396
Test accuracy: 0.05400000140070915
```

```
import pylab as plt

plt.imshow(x_test[130:131].reshape(28,28),cmap='gray')
plt.show()
```



```
import numpy as np
prediction = model.predict(x_test[130:131])
print('Prediction Score:\n',prediction[0])
thresholded = (prediction>0.5)*1
print('\nThresholded Score:\n',thresholded[0])
print('\nPredicted Digit:\n',np.where(thresholded == 1))
```

```
1/1 [==============================] - 0s 270ms/step
Prediction Score:
 [0.04635328 0.1971374  0.09714174 0.0856269  0.09554319 0.12607086
 0.08828516 0.07313626 0.10755599 0.08314925]

Thresholded Score:
 [0 0 0 0 0 0 0 0 0 0]

Predicted Digit:
 (array([], dtype=int64), array([], dtype=int64))
```

```
model = MobileNet(input_shape=None, alpha=0.25, depth_multiplier=1, dropout=1e-3,
                              include_top=True, weights='imagenet', input_tensor=None,

model.summary()
```

```
Model: "mobilenet_0.25_224"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 conv1 (Conv2D)              (None, 112, 112, 8)       216
```

```
      conv1_bn (BatchNormalizatio  (None, 112, 112, 8)      32
      n)

      conv1_relu (ReLU)           (None, 112, 112, 8)       0

      conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 8)      72

      conv_dw_1_bn (BatchNormaliz  (None, 112, 112, 8)      32
      ation)

      conv_dw_1_relu (ReLU)       (None, 112, 112, 8)       0

      conv_pw_1 (Conv2D)          (None, 112, 112, 16)      128

      conv_pw_1_bn (BatchNormaliz  (None, 112, 112, 16)     64
      ation)

      conv_pw_1_relu (ReLU)       (None, 112, 112, 16)      0

      conv_pad_2 (ZeroPadding2D)  (None, 113, 113, 16)      0

      conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 16)       144

      conv_dw_2_bn (BatchNormaliz  (None, 56, 56, 16)       64
      ation)

      conv_dw_2_relu (ReLU)       (None, 56, 56, 16)        0

      conv_pw_2 (Conv2D)          (None, 56, 56, 32)        512

      conv_pw_2_bn (BatchNormaliz  (None, 56, 56, 32)       128
      ation)

      conv_pw_2_relu (ReLU)       (None, 56, 56, 32)        0

      conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 32)       288

      conv_dw_3_bn (BatchNormaliz  (None, 56, 56, 32)       128
      ation)

      conv_dw_3_relu (ReLU)       (None, 56, 56, 32)        0

      conv_pw_3 (Conv2D)          (None, 56, 56, 32)        1024

      conv_pw_3_bn (BatchNormaliz  (None, 56, 56, 32)       128
      ation)

      conv_pw_3_relu (ReLU)       (None, 56, 56, 32)        0

      conv_pad_4 (ZeroPadding2D)  (None, 57, 57, 32)        0
```

```
!wget https://notebooks.azure.com/vipulmishra/projects/labgail/raw/Cat.jpg
```

```
     --2023-03-15 09:53:51--  https://notebooks.azure.com/vipulmishra/projects/labga
     Resolving notebooks.azure.com (notebooks.azure.com)... 13.107.237.38, 13.107.23
```

```
Connecting to notebooks.azure.com (notebooks.azure.com)|13.107.237.38|:443... c
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://visualstudio.microsoft.com/vs/features/notebooks-at-microsoft
--2023-03-15 09:53:51--  https://visualstudio.microsoft.com/vs/features/noteboo
Resolving visualstudio.microsoft.com (visualstudio.microsoft.com)... 23.60.121.
Connecting to visualstudio.microsoft.com (visualstudio.microsoft.com)|23.60.121
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://visualstudio.microsoft.com/vs/features/notebooks-at-microsoft
--2023-03-15 09:53:52--  https://visualstudio.microsoft.com/vs/features/noteboo
Reusing existing connection to visualstudio.microsoft.com:443.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘Cat.jpg’

Cat.jpg                 [  <=>               ] 200.64K   641KB/s    in 0.3s

2023-03-15 09:53:52 (641 KB/s) - ‘Cat.jpg’ saved [205459]
```

```
# Write the image name below

img_path = 'Cat.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
print('Predicted:\n', decode_predictions(preds))
```

```
------------------------------------------------
---------------------------
AttributeError
Traceback (most recent call last)
<ipython-input-24-a2b9e714b104> in <module>
      2
      3 img_path = 'Cat.jpg'
----> 4 img = image.load_img(img_path,
target_size=(224, 224))
      5 x = image.img_to_array(img)
      6 x = np.expand_dims(x, axis=0)

AttributeError: module
'keras.preprocessing.image' has no attribute
```

```
features = model.predict(x)
print('\nFeature Shape:\n',features.shape)
print('\nFeatures:\n',features)
```

```
Feature Shape:
 (1, 1000)

Features:
 [[1.66888913e-06 7.13712478e-04 1.65773949e-04 1.10442423e-04
  1.78126080e-04 4.26019914e-03 3.12949414e-04 1.41129553e-04
  6.23050655e-05 1.60201679e-07 9.00852172e-07 2.61550358e-05
```

```
        8.94293578e-07 1.74001452e-05 5.53141617e-05 3.42096632e-06
        2.29536290e-05 1.24991828e-04 2.50077574e-06 7.43013152e-06
        1.03183777e-06 1.27221181e-04 7.50785694e-06 6.57101191e-05
        2.86815452e-06 1.59369131e-06 5.43632996e-05 5.69003023e-05
        3.04480545e-05 3.72154475e-03 6.52500773e-07 4.07434391e-06
        2.95472978e-06 2.60654792e-06 2.85955139e-06 1.32700757e-06
        5.19292325e-06 2.66793165e-07 3.75387492e-04 9.01975272e-06
        9.18589649e-06 7.87469708e-06 1.11574482e-05 4.63036631e-05
        1.34447000e-05 7.70627594e-06 4.13208290e-05 1.56747512e-04
        1.50154918e-07 1.80126281e-05 6.31855801e-05 3.84678220e-04
        1.07686401e-05 4.20531542e-05 2.18738896e-05 2.58909786e-05
        4.34409340e-05 5.82936593e-07 2.88289320e-05 8.68797088e-06
        9.98507949e-06 1.99287228e-06 6.61912372e-06 4.11974543e-06
        4.94684718e-05 8.18191074e-06 1.08436034e-04 3.19992932e-06
        1.51962577e-05 4.55965733e-07 1.20070615e-06 3.42101544e-06
        4.29523789e-06 2.68541244e-05 2.85612423e-05 4.45490826e-07
        1.50208198e-05 3.16139449e-06 3.84000487e-05 1.69067662e-05
        1.82684416e-05 5.04864547e-05 9.19906324e-06 1.62637753e-05
        3.03273691e-06 1.42945801e-05 5.46305637e-06 2.26383941e-06
        3.21294669e-06 4.28476051e-04 1.72072123e-06 2.40450709e-06
        4.46220469e-07 2.25696596e-07 4.45920392e-04 1.35594610e-06
        4.53920507e-07 1.13232625e-06 5.29129295e-07 6.32942829e-06
        1.30360525e-06 3.97373611e-07 2.23191634e-07 8.75962769e-08
        1.33252597e-05 5.42196119e-07 3.71434840e-06 1.97501085e-03
        3.31861549e-04 1.85564277e-05 4.22540773e-03 5.92080432e-05
        1.60200223e-02 7.55966976e-05 2.15063512e-04 1.12470016e-02
        7.98921428e-06 6.46611588e-05 5.58132626e-07 2.74175335e-07
        2.90674279e-07 5.97470398e-06 1.18446178e-06 2.04822896e-07
        1.31906609e-06 1.10680239e-05 3.89876914e-05 1.04638948e-05
        1.00669979e-06 3.41091974e-04 2.75601742e-05 6.39991049e-06
        8.02993163e-05 1.70527528e-06 3.83506631e-06 3.11332406e-05
        9.49885271e-06 1.39489180e-06 1.50960921e-06 1.24261999e-06
        7.28587793e-06 1.45960039e-05 5.05366233e-06 7.73188549e-06
        3.27978023e-06 1.18008938e-06 7.55732344e-06 2.40890245e-06
        1.71447955e-05 1.63804107e-05 6.22064181e-06 3.54598882e-03
        9.77691496e-04 3.96944908e-03 2.05594275e-04 7.72019630e-05
        2.41241196e-05 2.11249031e-02 5.85951726e-04 3.07262053e-06
        4.19998069e-05 7.37487994e-07 5.71288865e-07 8.32925593e-07
        1.79322242e-07 1.92527665e-07 2.35278932e-07 1.44567920e-07
        1.04773903e-06 2.23539519e-05 1.85781641e-06 1.25755241e-05
        1.66605230e-06 5.38221029e-05 1.81694049e-04 4.26808811e-06
        2.07505764e-06 2.51837491e-07 2.95892028e-06 1.31583283e-05
        5.83013070e-06 1.71111787e-05 3.08867925e-06 8.29892451e-05
        3.09993593e-05 1.06881460e-04 1.60454575e-03 1.41253427e-03
        2.34789317e-04 7.32020999e-05 2.93969992e-04 6.35958088e-07
        5.92165394e-04 1.97934001e-04 7.90607228e-05 1.14216258e-04
        4.74516601e-05 7.03533124e-06 3.87190921e-05 2.83576595e-03
        1.30046988e-04 1.70255356e-04 1.88684135e-05 5.48596382e-02
        2.07964986e-04 1.73351589e-06 6.91947761e-08 3.77151155e-05
        5.82823532e-06 1.49515316e-07 1.07932742e-07 1.12447265e-06
```

```
model_minimal = Model(input=model.input, output=model.get_layer('conv_dw_2_relu').output)

conv_dw_2_relu_features = model_minimal.predict(x)
print('Features of conv_dw_2_relu:',conv_dw_2_relu_features.shape)
```

```
    Features of conv_dw_2_relu: (1, 56, 56, 16)
```

```
import matplotlib as mp
%matplotlib inline
import matplotlib.pyplot as plt
import tensorflow as tf
import tensorflow.contrib.slim as slim
from tensorflow.examples.tutorials.mnist import input_data
import math
```

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
    W0727 18:11:57.166813 140145501456256 deprecation.py:323] From <ipython-input-1
    Instructions for updating:
    Please use alternatives such as official/mnist/dataset.py from tensorflow/model
    W0727 18:11:57.178676 140145501456256 deprecation.py:323] From /usr/local/lib/p
    Instructions for updating:
    Please write your own downloading logic.
    W0727 18:11:57.180489 140145501456256 deprecation.py:323] From /usr/local/lib/p
    Instructions for updating:
    Please use urllib or similar directly.
    W0727 18:11:57.255538 140145501456256 deprecation.py:323] From /usr/local/lib/p
    Instructions for updating:
    Please use tf.data to implement this functionality.
    Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
    Extracting MNIST_data/train-images-idx3-ubyte.gz
    W0727 18:11:57.570780 140145501456256 deprecation.py:323] From /usr/local/lib/p
    Instructions for updating:
    Please use tf.data to implement this functionality.
    W0727 18:11:57.573707 140145501456256 deprecation.py:323] From /usr/local/lib/p
    Instructions for updating:
    Please use tf.one_hot on tensors.
    W0727 18:11:57.668081 140145501456256 deprecation.py:323] From /usr/local/lib/p
    Instructions for updating:
    Please use alternatives such as official/mnist/dataset.py from tensorflow/model
    Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
    Extracting MNIST_data/train-labels-idx1-ubyte.gz
    Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
    Extracting MNIST_data/t10k-images-idx3-ubyte.gz
    Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
    Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

```
tf.reset_default_graph()

x = tf.placeholder(tf.float32, [None, 784],name="x-in")
true_y = tf.placeholder(tf.float32, [None, 10],name="y-in")
keep_prob = tf.placeholder("float")

x_image = tf.reshape(x,[-1,28,28,1])
hidden_1 = slim.conv2d(x_image,5,[5,5])
pool_1 = slim.max_pool2d(hidden_1,[2,2])
hidden_2 = slim.conv2d(pool_1,5,[5,5])
pool_2 = slim.max_pool2d(hidden_2,[2,2])
hidden_3 = slim.conv2d(pool_2,20,[5,5])
hidden_3 = slim.dropout(hidden_3,keep_prob)
```

```
out_y = slim.fully_connected(slim.flatten(hidden_3),10,activation_fn=tf.nn.softmax)

cross_entropy = -tf.reduce_sum(true_y*tf.log(out_y))
correct_prediction = tf.equal(tf.argmax(out_y,1), tf.argmax(true_y,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
```

```
W0727 18:11:58.743405 140145501456256 deprecation.py:323] From /usr/local/lib/p
Instructions for updating:
Use keras.layers.flatten instead.
```

```
batchSize = 50
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)
for i in range(1001):
    batch = mnist.train.next_batch(batchSize)
    sess.run(train_step, feed_dict={x:batch[0],true_y:batch[1], keep_prob:0.5})
    if i % 100 == 0 and i != 0:
        trainAccuracy = sess.run(accuracy, feed_dict={x:batch[0],true_y:batch[1], keep_pro
        print("step %d, training accuracy %g"%(i, trainAccuracy))
```

```
step 100, training accuracy 0.16
step 200, training accuracy 0.64
step 300, training accuracy 0.76
step 400, training accuracy 0.92
step 500, training accuracy 0.9
step 600, training accuracy 0.86
step 700, training accuracy 0.86
step 800, training accuracy 0.88
step 900, training accuracy 0.84
step 1000, training accuracy 0.92
```

```
testAccuracy = sess.run(accuracy, feed_dict={x:mnist.test.images,true_y:mnist.test.labels,
print("test accuracy %g"%(testAccuracy))
```

```
test accuracy 0.912
```

## Get activation values and plotting

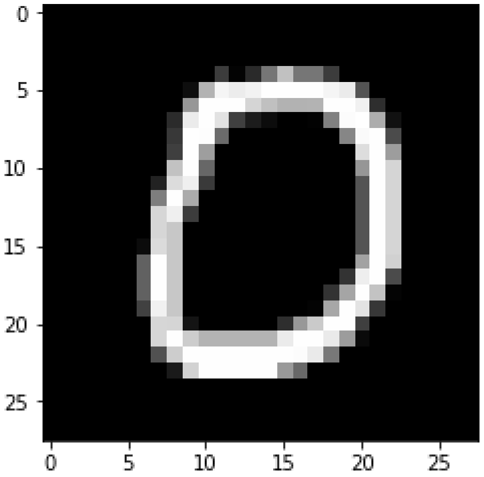
```
def getActivations(layer,stimuli):
    units = sess.run(layer,feed_dict={x:np.reshape(stimuli,[1,784],order='F'),keep_prob:1.
    plotNNFilter(units)

def plotNNFilter(units):
    filters = units.shape[3]
    plt.figure(1, figsize=(20,20))
    n_columns = 6
    n_rows = math.ceil(filters / n_columns) + 1
    for i in range(filters):
        plt.subplot(n_rows, n_columns, i+1)
        plt.title('Filter ' + str(i))
        plt.imshow(units[0,:,:,i], interpolation="nearest", cmap="gray")
```

## Input Image

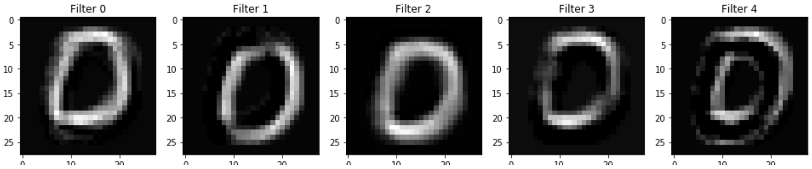
```
imageToUse = mnist.test.images[10]
plt.imshow(np.reshape(imageToUse,[28,28]), interpolation="nearest", cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7f75dae254e0>
```



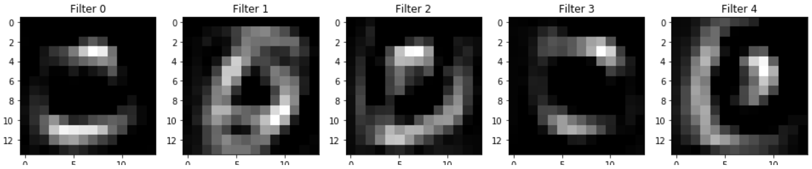
## Activation in Layer 1

```
getActivations(hidden_1,imageToUse)
```



## Activation in Layer 2

```
getActivations(hidden_2,imageToUse)
```



## Activation in Layer 3

```
getActivations(hidden_3,imageToUse)
```

```
---------------------------------------------------
---------------------------
NameError
Traceback (most recent call last)
<ipython-input-15-ad2a3ca9549c> in <module>
----> 1 getActivations(hidden_3,imageToUse)

NameError: name 'getActivations' is not defined
```

SEARCH STACK OVERFLOW

```
K.clear_session()
start = timeit.default_timer()
model = Sequential()
model.add(Conv2D(8, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adade
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_da
end = timeit.default_timer()
print("Time Taken to run the model:",end - start, "seconds")
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 8)         80
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 16)        1168
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                294944
_________________________________________________________________
dense_2 (Dense)              (None, 10)                330
=================================================================
Total params: 296,522
Trainable params: 296,522
Non-trainable params: 0
_________________________________________________________________
```

```
    Train on 60000 samples, validate on 10000 samples
    Epoch 1/2
    60000/60000 [==============================] - 36s 595us/step - loss: 0.2562 -
    Epoch 2/2
    60000/60000 [==============================] - 38s 625us/step - loss: 0.0735 -
    Time Taken to run the model: 73.62484016500002 seconds
```

```
K.clear_session()
start = timeit.default_timer()
model = Sequential()
model.add(Conv2D(8, kernel_size=(9, 9), activation='relu', input_shape=input_shape))
model.add(Conv2D(16, (9, 9), activation='relu'))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adade
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_da
end = timeit.default_timer()
print("Time Taken to run the model:",end - start, "seconds")
```

```
start = timeit.default_timer()
model = Sequential()
model.add(Conv2D(8, kernel_size=(7, 7), strides=2, activation='relu', input_shape=input_sh
model.add(Conv2D(16, (7, 7), strides=2, activation='relu'))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adade
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_da
end = timeit.default_timer()
print("Time Taken to run the model:",end - start, "seconds")
```

```
start = timeit.default_timer()
model = Sequential()
model.add(Conv2D(8, kernel_size=(7, 7), strides=1, padding='same', activation='relu', inpu
model.add(Conv2D(16, (7, 7), strides=1, padding='same', activation='relu'))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adade
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_da
end = timeit.default_timer()
print("Time Taken to run the model:",end - start, "seconds")
```

```
start = timeit.default_timer()
model = Sequential()
model.add(Conv2D(8, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adade
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_da
end = timeit.default_timer()
print("Time Taken to run the model:",end - start, "seconds")


# Write your code here

# Use the same model design from the above cell
```