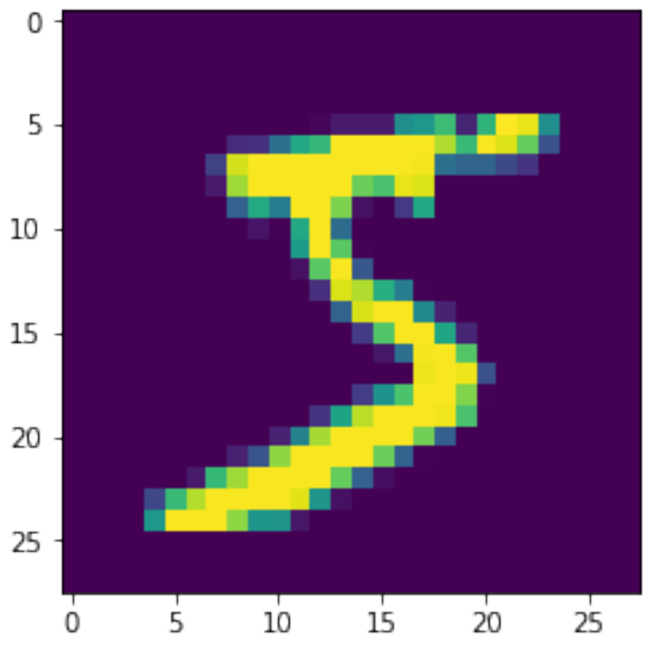
9. Code Listing (no screenshots)

9.1 Python Codes

```
                                                                In [122]:
#The trained model in pyhton - MSc Advanced Software Enginnering

#It provides a link between pyhton 2 and 3
from __future__ import print_function
#importing library to plot
from matplotlib import pyplot as plt

#Importing keras and sequential model to add the layers
import keras
#Importing MNIST dataset
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import np_utils
from keras import backend as K

#Importing coremltools to convert the model in the form of core ml. Also, Keras version 2.2.2 does not suppor
t core ml which should be kept in mind. Keras version 2.0.6
#Keras updated version downgraded
import coremltools
print('keras version ', keras.__version__)

keras version  2.0.6
                                                                In [123]:
#Confusion matrix is imported to test the model with errors
from sklearn.metrics import confusion_matrix
                                                                In [124]:
#The dataser is divided in train (60000) and validation (10000)
(x_train, y_train), (x_val, y_val) = mnist.load_data()
                                                                In [125]:
# Inspect x data
print('x_train shape: ', x_train.shape)
# Displays (60000, 28, 28)
print(x_train.shape[0], 'training samples')
# Displays 60000 train samples
print('x_val shape: ', x_val.shape)
# Displays (10000, 28, 28)
print(x_val.shape[0], 'validation samples')
# Displays 10000 validation samples
```

```
print('First x element\n', x_train[0])
# Displays an array of 28 arrays, each containing 28 gray-scale values between 0 and 255
# Plot first x sample
plt.imshow(x_train[0])
plt.show()

# Inspect the data
print('y_train shape: ', y_train.shape)
# Displays (60000,)
print('First 10 y_train elements:', y_train[:10])
# Displays [5 0 4 1 9 2 1 3 1 4]
```

```
x_train shape:  (60000, 28, 28)
60000 training samples
x_val shape:  (10000, 28, 28)
10000 validation samples
First x element
 [[   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    3   18   18   18  126  136
   175   26  166  255  247  127    0    0    0    0]
 [   0    0    0    0    0    0    0    0   30   36   94  154  170  253  253  253  253  253
   225  172  253  242  195   64    0    0    0    0]
 [   0    0    0    0    0    0    0   49  238  253  253  253  253  253  253  253  253  251
    93   82   82   56   39    0    0    0    0    0]
 [   0    0    0    0    0    0    0   18  219  253  253  253  253  253  198  182  247  241
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0   80  156  107  253  253  205   11    0   43  154
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0   14    1  154  253   90    0    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0  139  253  190    2    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0   11  190  253   70    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0   35  241  225  160  108    1
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0   81  240  253  253  119
    25    0    0    0    0    0    0    0    0    0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```



```
y_train shape:  (60000,)
First 10 y_train elements: [5 0 4 1 9 2 1 3 1 4]
```

In [126]:

```
print('y_train shape: ', y_train.shape)
print('x_val shape: ', x_val.shape)
print('x_train shape: ', x_train.shape)
y_train shape:  (60000,)
x_val shape:  (10000, 28, 28)
x_train shape:  (60000, 28, 28)
```

In [127]:

```
#MNIST data is shaped rows and coloums
#There 10 classes as we have ten numbers from 0 to 9
img_rows, img_cols = x_train.shape[1], x_train.shape[2]
num_classes = 10
```

In [128]:

```
#Sets input_shape for channels_first or channels_last
#We have one channel because of black and white images
if K.image_data_format() == 'channels_first':
  x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
  x_val = x_val.reshape(x_val.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_val = x_val.reshape(x_val.shape[0], img_rows, img_cols, 1)
  input_shape = (img_rows, img_cols, 1)
```

In [129]:

```
print('x_train shape:', x_train.shape)
# x_train shape: (60000, 28, 28, 1)
print('x_val shape:', x_val.shape)
# x_val shape: (10000, 28, 28, 1)
print('input_shape:', input_shape)
# input_shape: (28, 28, 1) as it is always data shape is used.
#TensorFlow image data format is channels last
x_train shape: (60000, 28, 28, 1)
x_val shape: (10000, 28, 28, 1)
input_shape: (28, 28, 1)
```

In [130]:

```
#The model needs the data values in a specific format
#Our MNIST image data values are uint8 in the range of [0,255]
#Keras needs values float32 in the range of [0,1]
x_train = x_train.astype('float32')
x_val = x_val.astype('float32')
x_train /= 255
x_val /= 255
```

In [131]:

```
print('First x sample, normalized\n', x_train[0])
# An array of 28 arrays, each containing 28 arrays, each with one value between 0 and 1

First x sample, normalized
 [[[0.        ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]]

[[0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.          ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]]

[[0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]]

[[0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.         ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]]

[[0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
```

```
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.01176471]
  [0.07058824]
  [0.07058824]
  [0.07058824]
  [0.49411765]
  [0.53333336]
  [0.6862745 ]
  [0.10196079]
  [0.6509804 ]
  [1.        ]
  [0.96862745]
  [0.49803922]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.11764706]
  [0.14117648]
  [0.36862746]
  [0.6039216 ]
  [0.6666667 ]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.88235295]
  [0.6745098 ]
  [0.99215686]
  [0.9490196 ]
  [0.7647059 ]
  [0.2509804 ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.19215687]
  [0.93333334]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.9843137 ]
  [0.3647059 ]
  [0.32156864]
  [0.32156864]
  [0.21960784]
  [0.15294118]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
```

```
  [0.        ]
  [0.        ]
  [0.        ]
  [0.07058824]
  [0.85882354]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.7764706 ]
  [0.7137255 ]
  [0.96862745]
  [0.94509804]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.3137255 ]
  [0.6117647 ]
  [0.41960785]
  [0.99215686]
  [0.99215686]
  [0.8039216 ]
  [0.04313726]
  [0.        ]
  [0.16862746]
  [0.6039216 ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.05490196]
  [0.00392157]
  [0.6039216 ]
  [0.99215686]
  [0.3529412 ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.54509807]
  [0.99215686]
  [0.74509805]
  [0.00784314]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.04313726]
  [0.74509805]
  [0.99215686]
  [0.27450982]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]
```

```
[[0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.13725491]
 [0.94509804]
 [0.88235295]
 [0.627451  ]
 [0.42352942]
 [0.00392157]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]]

[[0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.31764707]
 [0.9411765 ]
 [0.99215686]
 [0.99215686]
 [0.46666667]
```

```
  [0.09803922]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.1764706 ]
  [0.7294118 ]
  [0.99215686]
  [0.99215686]
  [0.5882353 ]
  [0.10588235]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.0627451 ]
  [0.3647059 ]
  [0.9882353 ]
  [0.99215686]
  [0.73333335]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.9764706 ]
  [0.99215686]
  [0.9764706 ]
  [0.2509804 ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.18039216]
  [0.50980395]
  [0.7176471 ]
  [0.99215686]
  [0.99215686]
  [0.8117647 ]
  [0.00784314]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.15294118]
  [0.5803922 ]
  [0.8980392 ]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.98039216]
  [0.7137255 ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.09411765]
  [0.44705883]
  [0.8666667 ]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.7882353 ]
  [0.30588236]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.09019608]
  [0.25882354]
  [0.8352941 ]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.7764706 ]
  [0.31764707]
  [0.00784314]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.07058824]
  [0.67058825]
  [0.85882354]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.7647059 ]
  [0.3137255 ]
  [0.03529412]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.21568628]
  [0.6745098 ]
  [0.8862745 ]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.95686275]
  [0.52156866]
  [0.04313726]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.53333336]
  [0.99215686]
  [0.99215686]
  [0.99215686]
  [0.83137256]
  [0.5294118 ]
  [0.5176471 ]
  [0.0627451 ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]
```

```
[[0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]]

[[0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
```

```
  [0.         ]
  [0.         ]
  [0.         ]
  [0.         ]
  [0.         ]
  [0.         ]
  [0.         ]
  [0.         ]
  [0.         ]
  [0.         ]
  [0.         ]]]
```

In [132]:

```
print('y_train shape: ', y_train.shape)
# (60000,) as there are 60000 images
print('First 10 y_train elements:', y_train[:10])
# [5 0 4 1 9 2 1 3 1 4]
# Convert 1-dimensional class arrays to 10-dimensional class matrices
y_train = np_utils.to_categorical(y_train, num_classes)
y_val = np_utils.to_categorical(y_val, num_classes)
print('New y_train shape: ', y_train.shape)
# (60000, 10)
```

```
y_train shape:  (60000,)
First 10 y_train elements: [5 0 4 1 9 2 1 3 1 4]
New y_train shape:  (60000, 10)
```

In [133]:

```
print('New y_train shape: ', y_train.shape)
# (60000, 10)
print('First 10 y_train elements, reshaped:\n', y_train[:10])
# An array of 10 arrays, each with 10 elements,
# all zeros except at index 5, 0, 4, 1, 9, 2, 1, 3, 1, 4.
```

```
New y_train shape:  (60000, 10)
First 10 y_train elements, reshaped:
 [[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]
```

In [134]:

```
#First layer has information about the input
#32, 64, 128 are the number of filters
#(5,5), (3,3), (1,1) are the kernel size
```

```
#It is specifying the width and height
#Relu and Softmax are activation functions
#They decided to fire/activate if r=the weight sum is greater than threshold
#Maxpooling 2x2 filter
#Flatten makes the convolutional layers 1 dimensional
model_m = Sequential()
model_m.add(Conv2D(32, (5, 5), input_shape=input_shape, activation='relu'))
model_m.add(MaxPooling2D(pool_size=(2, 2)))
model_m.add(Dropout(0.5))
model_m.add(Conv2D(64, (3, 3), activation='relu'))
model_m.add(MaxPooling2D(pool_size=(2, 2)))
model_m.add(Dropout(0.2))
model_m.add(Conv2D(128, (1, 1), activation='relu'))
model_m.add(MaxPooling2D(pool_size=(2, 2)))
model_m.add(Dropout(0.2))
model_m.add(Flatten())
model_m.add(Dense(128, activation='relu'))
model_m.add(Dense(num_classes, activation='softmax'))
# Inspect model's layers, output shapes, number of trainable parameters
print(model_m.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_10 (Conv2D)           (None, 24, 24, 32)        832       
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 12, 12, 32)        0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 12, 12, 32)        0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 10, 10, 64)        18496     
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 5, 5, 64)          0         
_________________________________________________________________
dropout_11 (Dropout)         (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 5, 5, 128)         8320      
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 2, 2, 128)         0         
_________________________________________________________________
dropout_12 (Dropout)         (None, 2, 2, 128)         0         
_________________________________________________________________
flatten_4 (Flatten)          (None, 512)               0         
_________________________________________________________________
dense_7 (Dense)              (None, 128)               65664     
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290      
```

```
=================================================================
Total params: 94,602
Trainable params: 94,602
Non-trainable params: 0
_________________________________________________________________
None
```

In [135]:

```
#This is optional for the fit function, if it fails to improve for two consecutive epochs
callbacks_list = [
   keras.callbacks.ModelCheckpoint(
      filepath='best_model.{epoch:02d}-{val_loss:.2f}.h5',
      monitor='val_loss', save_best_only=True),
   keras.callbacks.EarlyStopping(monitor='acc', patience=1)
]
```

In [136]:

```
model_m.compile(loss='categorical_crossentropy',
            optimizer='adam', metrics=['accuracy'])

# Hyper-parameters
batch_size = 200
epochs = 20

# Enable validation to use ModelCheckpoint and EarlyStopping callbacks.
model_m.fit(
   x_train, y_train, batch_size=batch_size, epochs=epochs,
   callbacks=callbacks_list, validation_data=(x_val, y_val), verbose=1)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
60000/60000 [==============================] - 72s - loss: 0.5977 - acc: 0.
8037 - val_loss: 0.1381 - val_acc: 0.9573
Epoch 2/20
60000/60000 [==============================] - 76s - loss: 0.1900 - acc: 0.
9408 - val_loss: 0.0890 - val_acc: 0.9730
Epoch 3/20
60000/60000 [==============================] - 82s - loss: 0.1390 - acc: 0.
9567 - val_loss: 0.0643 - val_acc: 0.9810
Epoch 4/20
60000/60000 [==============================] - 76s - loss: 0.1122 - acc: 0.
9649 - val_loss: 0.0532 - val_acc: 0.9825
Epoch 5/20
60000/60000 [==============================] - 71s - loss: 0.0984 - acc: 0.
9684 - val_loss: 0.0504 - val_acc: 0.9846
Epoch 6/20
60000/60000 [==============================] - 71s - loss: 0.0838 - acc: 0.
9737 - val_loss: 0.0445 - val_acc: 0.9871
```

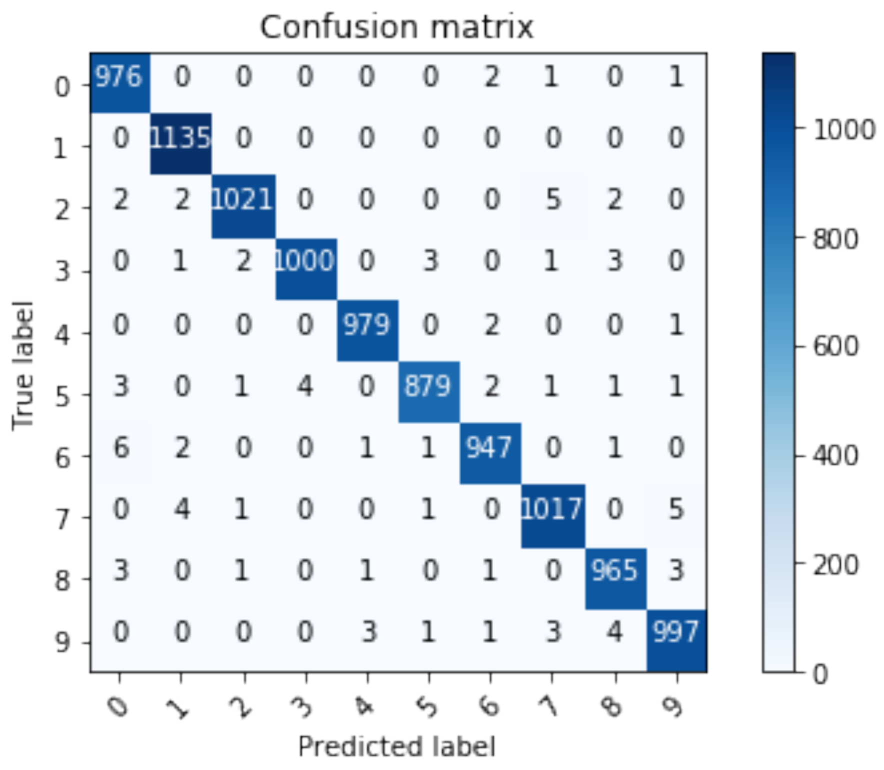
```
Epoch 7/20
60000/60000 [==============================] - 74s - loss: 0.0765 - acc: 0.
9758 - val_loss: 0.0401 - val_acc: 0.9875
Epoch 8/20
60000/60000 [==============================] - 70s - loss: 0.0702 - acc: 0.
9779 - val_loss: 0.0378 - val_acc: 0.9890
Epoch 9/20
60000/60000 [==============================] - 70s - loss: 0.0664 - acc: 0.
9790 - val_loss: 0.0334 - val_acc: 0.9897
Epoch 10/20
60000/60000 [==============================] - 76s - loss: 0.0611 - acc: 0.
9802 - val_loss: 0.0333 - val_acc: 0.9894
Epoch 11/20
60000/60000 [==============================] - 75s - loss: 0.0591 - acc: 0.
9812 - val_loss: 0.0304 - val_acc: 0.9902
Epoch 12/20
60000/60000 [==============================] - 69s - loss: 0.0562 - acc: 0.
9823 - val_loss: 0.0293 - val_acc: 0.9915
Epoch 13/20
60000/60000 [==============================] - 69s - loss: 0.0533 - acc: 0.
9833 - val_loss: 0.0271 - val_acc: 0.9922
Epoch 14/20
60000/60000 [==============================] - 68s - loss: 0.0502 - acc: 0.
9843 - val_loss: 0.0315 - val_acc: 0.9906
Epoch 15/20
60000/60000 [==============================] - 68s - loss: 0.0496 - acc: 0.
9839 - val_loss: 0.0275 - val_acc: 0.9919
Epoch 16/20
60000/60000 [==============================] - 69s - loss: 0.0464 - acc: 0.
9853 - val_loss: 0.0251 - val_acc: 0.9917
Epoch 17/20
60000/60000 [==============================] - 68s - loss: 0.0425 - acc: 0.
9859 - val_loss: 0.0257 - val_acc: 0.9925
Epoch 18/20
60000/60000 [==============================] - 69s - loss: 0.0431 - acc: 0.
9858 - val_loss: 0.0267 - val_acc: 0.9919
Epoch 19/20
60000/60000 [==============================] - 69s - loss: 0.0425 - acc: 0.
9862 - val_loss: 0.0238 - val_acc: 0.9928
Epoch 20/20
60000/60000 [==============================] - 68s - loss: 0.0386 - acc: 0.
9880 - val_loss: 0.0257 - val_acc: 0.9916
```

Out[136]:

```
<keras.callbacks.History at 0x1c32a1c9b0>
```

In [137]:

```python
import numpy as np
import argparse
import pickle
from sklearn.metrics import confusion_matrix
from keras.models import load_model
                                                            In [138]:
y_pred = model_m.predict(x_val)
                                                            In [139]:
labels = [np.argmax(pred) for pred in y_pred]
                                                            In [140]:
labels_actual = [np.argmax(pred) for pred in y_val]
                                                            In [141]:
confusion_matrix(labels_actual, labels)
                                                            Out[141]:
array([[ 976,    0,    0,    0,    0,    0,    2,    1,    0,    1],
       [   0, 1135,    0,    0,    0,    0,    0,    0,    0,    0],
       [   2,    2, 1021,    0,    0,    0,    0,    5,    2,    0],
       [   0,    1,    2, 1000,    0,    3,    0,    1,    3,    0],
       [   0,    0,    0,    0,  979,    0,    2,    0,    0,    1],
       [   3,    0,    1,    4,    0,  879,    2,    1,    1,    1],
       [   6,    2,    0,    0,    1,    1,  947,    0,    1,    0],
       [   0,    4,    1,    0,    0,    1,    0, 1017,    0,    5],
       [   3,    0,    1,    0,    1,    0,    1,    0,  965,    3],
       [   0,    0,    0,    0,    3,    1,    1,    3,    4,  997]])
                                                            In [142]:
x=confusion_matrix(labels_actual, labels)
                                                            In [143]:
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


plot_confusion_matrix(x, range(10))
```



In [144]:

```
#Saving the model for Core ML

output_labels = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
# For the first argument, use the filename of the newest .h5 file in the notebook folder.
coreml_mnist = coremltools.converters.keras.convert(
    'best_model.00-0.19.h5', input_names=['image'], output_names=['output'],
    class_labels=output_labels, image_input_names='image')
```

```
0 : conv2d_7_input, <keras.engine.topology.InputLayer object at 0x1c41ecc97
8>
1 : conv2d_7, <keras.layers.convolutional.Conv2D object at 0x1c41eccc18>
2 : conv2d_7__activation__, <keras.layers.core.Activation object at 0x1c338
6fa90>
3 : max_pooling2d_7, <keras.layers.pooling.MaxPooling2D object at 0x1c41ecc
cc0>
4 : conv2d_8, <keras.layers.convolutional.Conv2D object at 0x1c41accc88>
5 : conv2d_8__activation__, <keras.layers.core.Activation object at 0x1c338
6fc50>
```

```
6 : max_pooling2d_8, <keras.layers.pooling.MaxPooling2D object at 0x1c41ecc
a90>
7 : conv2d_9, <keras.layers.convolutional.Conv2D object at 0x1c32bd0b00>
8 : conv2d_9__activation__, <keras.layers.core.Activation object at 0x1c338
6f7b8>
9 : max_pooling2d_9, <keras.layers.pooling.MaxPooling2D object at 0x1c330b2
400>
10 : flatten_3, <keras.layers.core.Flatten object at 0x1c32c2a630>
11 : dense_5, <keras.layers.core.Dense object at 0x1c32bf7278>
12 : dense_5__activation__, <keras.layers.core.Activation object at 0x1c338
6fc18>
13 : dense_6, <keras.layers.core.Dense object at 0x1c32e3f198>
14 : dense_6__activation__, <keras.layers.core.Activation object at 0x1c338
6f6a0>
```

In [145]:

```
print(coreml_mnist)
```

```
input {
  name: "image"
  type {
    imageType {
      width: 28
      height: 28
      colorSpace: GRAYSCALE
    }
  }
}
output {
  name: "output"
  type {
    dictionaryType {
      stringKeyType {
      }
    }
  }
}
output {
  name: "classLabel"
  type {
    stringType {
    }
  }
}
predictedFeatureName: "classLabel"
predictedProbabilitiesName: "output"
```

In [146]:

```
coreml_mnist.author = 'Mansur Can'
coreml_mnist.license = 'Mansur'
coreml_mnist.short_description = 'This is a trained model with MNIST data set for MSc project'
coreml_mnist.input_description['image'] = 'Digit image'
coreml_mnist.output_description['output'] = 'Probability of each digit'
coreml_mnist.output_description['classLabel'] = 'Labels of digits'
```

In [147]:

```
coreml_mnist.save('MansurCanMNISTClassifier.mlmodel')
```

In [ ]:

9.2 Swift Codes

9.2.1 View Controller Codes

```
//
//  ViewController.swift
//  MScProject
//
//  Created by Mansur Can on 22/08/2018.
//  Copyright © 2018 Mansur Can. All rights reserved.
//

import UIKit
import Vision
import CoreML

class ViewController: UIViewController {

    @IBOutlet weak var drawView: DrawView!
    @IBOutlet weak var digitLabel: UILabel!

    var requests = [VNRequest]()

    override func viewDidLoad() {
        super.viewDidLoad()
        setupVision()
        // Do any additional setup after loading the view, typically
from a nib.
    }

    func setupVision() {
        guard let visionModel = try? VNCoreMLModel(for:
MansurCanMNISTClassifier().model) else {fatalError("ML Model can not
be loaded")}

        let classificationRequest = VNCoreMLRequest(model:
visionModel, completionHandler: self.handleClassification)

         self.requests = [classificationRequest]
    }
    func handleClassification(request:VNRequest, error:Error?) {
        guard let observations = request.results else {print("There
is no result"); return}

        let classifications = observations
        .compactMap({$0 as? VNClassificationObservation})
        .filter({$0.confidence > 0.7})
        .map({$0.identifier})

        DispatchQueue.main.async {
            self.digitLabel.text = classifications.first
        }
    }

    @IBAction func clearDraw(_ sender: Any) {

        drawView.clearDraw()
    }

    @IBAction func predictDigit(_ sender: Any) {

        let image = UIImage(view: drawView)
        let scaledImage = scaleImage(image: image, toSize:
CGSize(width: 28, height: 28))

        let imageRequestHandler = VNImageRequestHandler(cgImage:
scaledImage.cgImage!, options: [:])

        do{
            try imageRequestHandler.perform(self.requests)
        }catch{
            print(error)
        }
    }

    func scaleImage (image:UIImage, toSize size:CGSize) -> UIImage {
        UIGraphicsBeginImageContextWithOptions(size, false, 1.0)
        image.draw(in: CGRect (x: 0, y: 0, width: size.width,
height: size.height))
        let newImage = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
        return newImage!
    }
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```