Import Packages

```
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
from keras.models import load_model
import timeit

import warnings
warnings.filterwarnings('ignore')
```

Image Data Processing

```
batch_size = 128
num_classes = 10
epochs = 10

# 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')
print(y_train)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
[5 0 4 ... 5 6 8]
```

```
# 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)



print(y_train)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]]
```

Build CNN Architecture

```
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"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 8)         80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 16)        1168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
dropout (Dropout)            (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 32)                12832
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                330
=================================================================
Total params: 14,410
```

```
Trainable params: 14,410
Non-trainable params: 0
_________________________________________________________________
```

Model Training

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

```
model_checkpoint_callback = keras.callbacks.ModelCheckpoint("best_Model.h5",save_best_only=True)
```

```
history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test),
          callbacks=[model_checkpoint_callback])
```

```
Epoch 1/10
469/469 [==============================] - 19s 40ms/step - loss: 2.3213 - accuracy: 0.0986 - val_loss: 2.2949 - val_acc
Epoch 2/10
469/469 [==============================] - 18s 39ms/step - loss: 2.3141 - accuracy: 0.1009 - val_loss: 2.2891 - val_acc
Epoch 3/10
469/469 [==============================] - 19s 40ms/step - loss: 2.3074 - accuracy: 0.1041 - val_loss: 2.2836 - val_acc
Epoch 4/10
469/469 [==============================] - 19s 40ms/step - loss: 2.3012 - accuracy: 0.1041 - val_loss: 2.2784 - val_acc
Epoch 5/10
469/469 [==============================] - 19s 40ms/step - loss: 2.2972 - accuracy: 0.1092 - val_loss: 2.2734 - val_acc
Epoch 6/10
469/469 [==============================] - 19s 40ms/step - loss: 2.2927 - accuracy: 0.1116 - val_loss: 2.2686 - val_acc
Epoch 7/10
469/469 [==============================] - 19s 40ms/step - loss: 2.2883 - accuracy: 0.1125 - val_loss: 2.2640 - val_acc
```

```
Epoch 8/10
469/469 [==============================] - 19s 40ms/step - loss: 2.2842 - accuracy: 0.1177 - val_loss: 2.2595 - val_acc
Epoch 9/10
469/469 [==============================] - 19s 40ms/step - loss: 2.2802 - accuracy: 0.1183 - val_loss: 2.2551 - val_acc
Epoch 10/10
469/469 [==============================] - 19s 40ms/step - loss: 2.2731 - accuracy: 0.1264 - val_loss: 2.2507 - val_acc
```

Model Testing

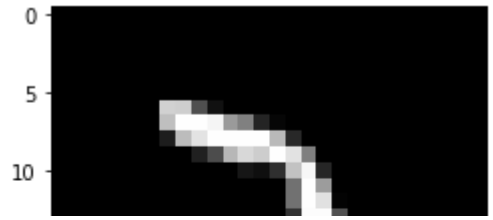
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 2.250657320022583
Test accuracy: 0.15479999780654907
```

Prediction Using Trained CNN Model

```
import pylab as plt

plt.imshow(x_test[122].reshape(28,28),cmap='gray')
plt.show()
```




```
import numpy as np
prediction = model.predict(x_test[119:120])
print('Prediction Score:\n',prediction[0])
thresholded = (prediction>0.5)*1
print('\nThresholded Score:\n',thresholded[0])
print('\nPredicted Digit:\n',np.argmax(thresholded))
```

```
Prediction Score:
 [0.08731683 0.08747971 0.10921669 0.10653644 0.09074822 0.11547276
 0.09577825 0.09448154 0.119935   0.09303454]

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

Predicted Digit:
 0
```

Model Deployment

```
model = load_model('best_Model.h5')
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 8)         80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 8)         0
_________________________________________________________________
```

```
conv2d_1 (Conv2D)            (None, 11, 11, 16)        1168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
dropout (Dropout)            (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 32)                12832
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                330
=================================================================
Total params: 14,410
Trainable params: 14,410
Non-trainable params: 0
_________________________________________________________________
```

```
import numpy as np
prediction = model.predict(x_test[119:120])
print('Prediction Score:\n',prediction[0])
thresholded = (prediction>0.5)*1
print('\nThresholded Score:\n',thresholded[0])
print('\nPredicted Digit:\n',np.argmax(thresholded))
```

```
Prediction Score:
 [0.08731683 0.08747971 0.10921669 0.10653644 0.09074822 0.11547276
 0.09577825 0.09448154 0.119935   0.09303454]

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

Predicted Digit:
 0
```