In [1]:

```
import numpy as np
import keras
from keras.datasets import mnist
from keras.models import Model
from keras.layers import Dense, Input
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten
from keras import backend as k
```

In [2]:

```
(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.goo
gleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 12s 1us/step
```

In [3]:

```
img_rows, img_cols=28, 28
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)
 inpx = (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)
 inpx = (img_rows, img_cols, 1)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

In [4]:

```
y_train = keras.utils.to_categorical(y_train)
27
y_test = keras.utils.to_categorical(y_test)
```

In [5]:

```
inpx = Input(shape=inpx)
layer1 = Conv2D(32, kernel_size=(3, 3), activation='relu')(inpx)
layer2 = Conv2D(64, (3, 3), activation='relu')(layer1)
layer3 = MaxPooling2D(pool_size=(3, 3))(layer2)
layer4 = Dropout(0.5)(layer3)
layer5 = Flatten()(layer4)
layer6 = Dense(250, activation='sigmoid')(layer5)
layer7 = Dense(10, activation='softmax')(layer6)
```

In [6]:

```
model = Model([inpx], layer7)
model.compile(optimizer=keras.optimizers.Adadelta(),
loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
model.fit(x_train, y_train, epochs=12, batch_size=500)
```

```
Epoch 1/12
120/120 [==============================] - 38s 276ms/step - loss: 2.4615 - accuracy: 0.1056
Epoch 2/12
120/120 [==============================] - 33s 276ms/step - loss: 2.4475 - accuracy: 0.1077
Epoch 3/12
120/120 [==============================] - 32s 267ms/step - loss: 2.4337 - accuracy: 0.1084
Epoch 4/12
120/120 [==============================] - 32s 268ms/step - loss: 2.4199 - accuracy: 0.1088
Epoch 5/12
120/120 [==============================] - 33s 277ms/step - loss: 2.4071 - accuracy: 0.1082
Epoch 6/12
120/120 [==============================] - 32s 269ms/step - loss: 2.3948 - accuracy: 0.1099
Epoch 7/12
120/120 [==============================] - 32s 270ms/step - loss: 2.3829 - accuracy: 0.1115
Epoch 8/12
120/120 [==============================] - 32s 267ms/step - loss: 2.3713 - accuracy: 0.1112
Epoch 9/12
120/120 [==============================] - 32s 268ms/step - loss: 2.3606 - accuracy: 0.1121
Epoch 10/12
120/120 [==============================] - 32s 269ms/step - loss: 2.3506 - accuracy: 0.1125
Epoch 11/12
120/120 [==============================] - 32s 270ms/step - loss: 2.3408 - accuracy: 0.1140
Epoch 12/12
120/120 [==============================] - 33s 274ms/step - loss: 2.3317 - accuracy: 0.1147
```

Out[6]:

```
<keras.callbacks.History at 0x1c80dee3b50>
```

In [7]:

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

```
loss= 2.3270552158355713
accuracy= 0.10740000009536743
```