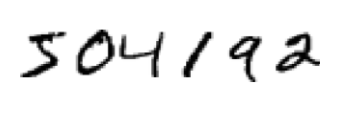
**Khellas bouchra**

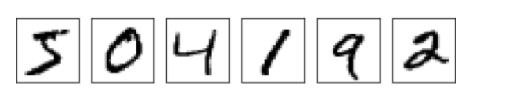
**GPRG0Z**

[**Using neural nets to recognize handwritten digits**](http://neuralnetworksanddeeplearning.com/chap1.html)

We can split the problem of recognizing handwritten digits into two sub-problems. First, we'd like a way of breaking an image containing many digits into a sequence of separate images, each containing a single digit. For example, we'd like to break the image

****

into six separate images,



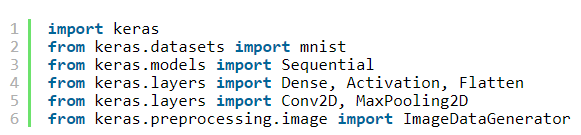
We humans solve this segmentation problem with ease, but it's challenging for a computer program to correctly break up the image. Once the image has been segmented, the program then needs to classify each individual digit. So, for instance, we'd like our program to recognize that the first digit above,



is a 5 we'll focus on writing a program to solve the second problem, that is, classifying individual digits. We do this because it turns out that the segmentation problem is not so difficult to solve, once you have a good way of classifying individual digits. There are many approaches to solving the segmentation problem.

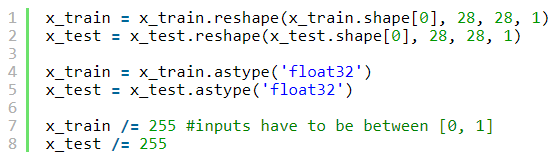
**\*Explanation of the project :**

Firstly, we will import required keras libraries:



Secondly, we would load mnist dataset. This dataset has already seperated as train and test sets. The both train and data set includes features and labels. 

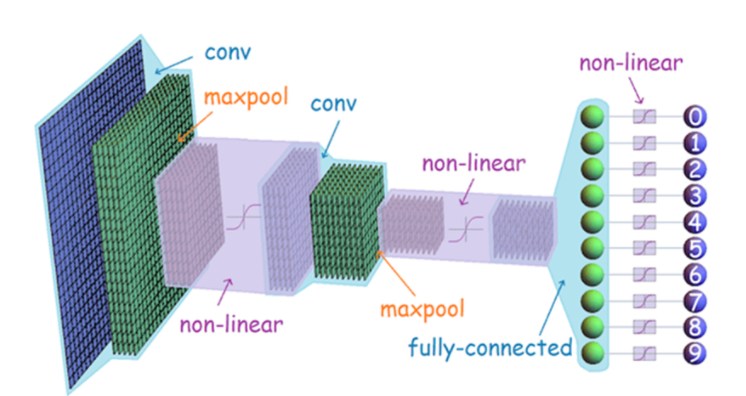
Thirdly, keras enforces us to work on 3D matrixes for input features. So, we would transform train set and test set features to 3D matrix. Input features are two dimensional matrix size of 28×28. These matrixes remain same, we just add a dummy dimension and matrix would be transformed to 28x28x1. Moreover, input features have to be between 0 and 1. That’s why, features would be divided to 255 to normalize [0, 1].



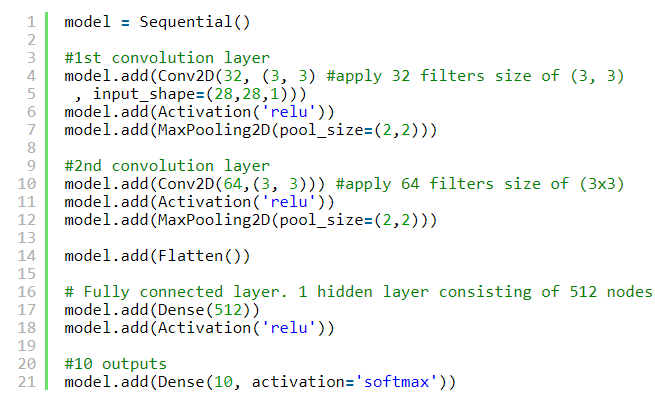
Dataset labels are in scale fo 0 to 9. Keras enforces us to work on binary class labels. The following block would transform labels to binary format. (e.g. label 2 would be represented as 0010000000)



That is not a must but we would stay loyal following structure. Convolution and pooling operations would be applied twice. After then, learned features would be transferred to a fully connected neural networks consisting of a hidden layer. You might change the structure of the network and monitor the effect on accuracy.

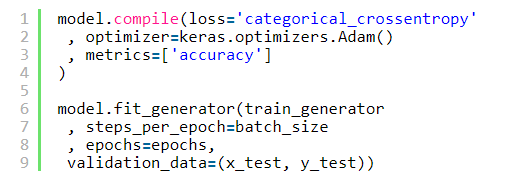


Now, we would construct the CNN structure.

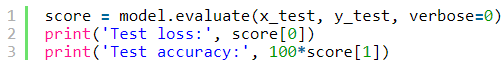


You may be gotten attention that output layer of fully connected neural networks is connected to CNN output layer with a non linear function. That function should be [softmax](https://sefiks.com/2017/11/08/softmax-as-a-neural-networks-activation-function/" \t "_blank). In this way, output values are normalized between [0, 1]. Also, sum of the outputs are always equal to 1. Finally, the maximum index would fire the result.

Now, it is time to train the network.



Once network is trained, we can question the success metrics.



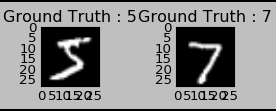
Classical fully connected neural networks retrieved 98.01% accuracy whereas convolutional neural networks did exceed the 99% accuracy limit. That is an incredible result.

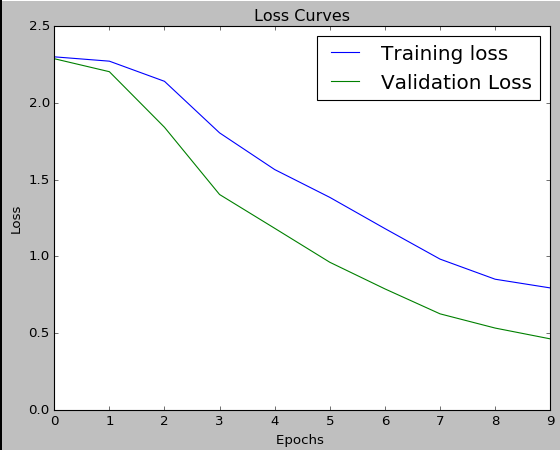


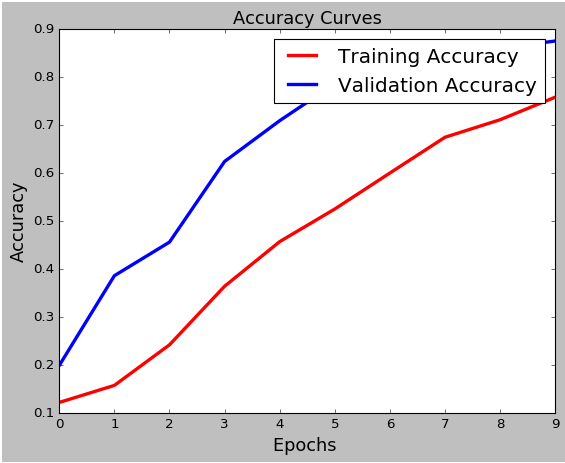
Finally, I have created the model with the following configuration



Result:







Train on 60000 samples, validate on 10000 samples

