

# **Developing an handwritten Digit Recognizer app for Android**

## **Project Proposal**

**Kabya Basu**

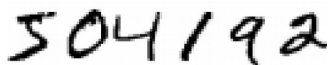
### **Domain Background:**

Image Classification has been a problem in computer vision for a while now. Many people tried many different approaches to solve this, possibly the most recent approach being deep learning. Deep learning has allowed us to 'teach' computers what different things/objects look like and let it identify them all by itself.

In this case we will be developing an android app to draw and detect handwritten digit. Although this might be a very basic app however I consider this project as my stepping stone to develop more useful app in the area of computer vision.

### **Problem Statement**

The human visual system is one of the wonders of the world. Consider the following sequence of handwritten digits:

A photograph of the handwritten digits '504192' in black ink on a white background. The digits are written in a casual, slightly slanted style.

Most people effortlessly recognize those digits as 504192. That ease is deceptive. In each hemisphere of our brain, humans have a primary visual cortex, also known as V1, containing 140 million neurons, with tens of billions of connections between them. And yet human vision involves not just V1, but an entire series of visual cortices - V2, V3, V4, and V5 - doing progressively more complex image processing. We carry in our heads a supercomputer, tuned by evolution over hundreds of millions of years, and superbly adapted to understand the visual world. Recognizing handwritten digits isn't easy. Rather, we humans are stupendously, astoundingly good at making sense of what our eyes show us. But nearly all that work is done unconsciously. And so we don't usually appreciate how tough a problem our visual systems solve.

The difficulty of visual pattern recognition becomes apparent if you attempt to write a computer program to recognize digits like those above. What seems easy when we do it ourselves suddenly becomes extremely difficult. Simple intuitions about how we recognize shapes - "a 9 has a loop at the top, and a vertical stroke in the bottom right" - turn out to be not so simple to express algorithmically. When you try to make such rules precise, you quickly get lost in a morass of exceptions and caveats and special cases. It seems hopeless.

### **Datasets and Inputs**

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.[1][2] The database is also widely used for training and testing in the field of machine learning.[3][4] It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments.[5] Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.[5]

The MNIST database contains 60,000 training images and 10,000 testing images.[6] Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset.[7] There have been a number of scientific papers on attempts to achieve the lowest error rate; one paper, using a hierarchical system of convolutional neural networks, manages to get an error rate

on the MNIST database of 0.23 percent.[8] The original creators of the database keep a list of some of the methods tested on it.[5] In their original paper, they use a support vector machine to get an error rate of 0.8 percent.[9]

The handwritten digits were centered in a 28x28 pixel image by calculating the center of mass of the pixels.

## **Solution Statement**

As I mentioned briefly before, I plan to use deep learning toward the final solution. The reason for this is that a deep learning model may be more effective at determining the important features in a given image than a human being using manual gradient and color thresholds in typical computer vision techniques, as well as other manual programming needed within that process. Specifically, I plan to use a convolutional neural network (CNN), which are very effective at finding patterns within images by using filters to find specific pixel groupings that are important. My aim is for the end result to be both more effective at detecting the digits as well as faster at doing so than common computer vision techniques.

## **Benchmark Model**

I plan to compare the results of the CNN versus the output of the computer vision-based model I created with softmax regression based model. I will compare the accuracy/mean-squared error to see which is more effective.

Also as I will be having two models one is from tensorflow and the other is from keras, I will have both the results printed when we press the detect button of the app. This will be interesting to see which one predicts better in live cases.

## **Evaluation Metrics**

I will define an accuracy function for all the models and use it for evaluating the model.

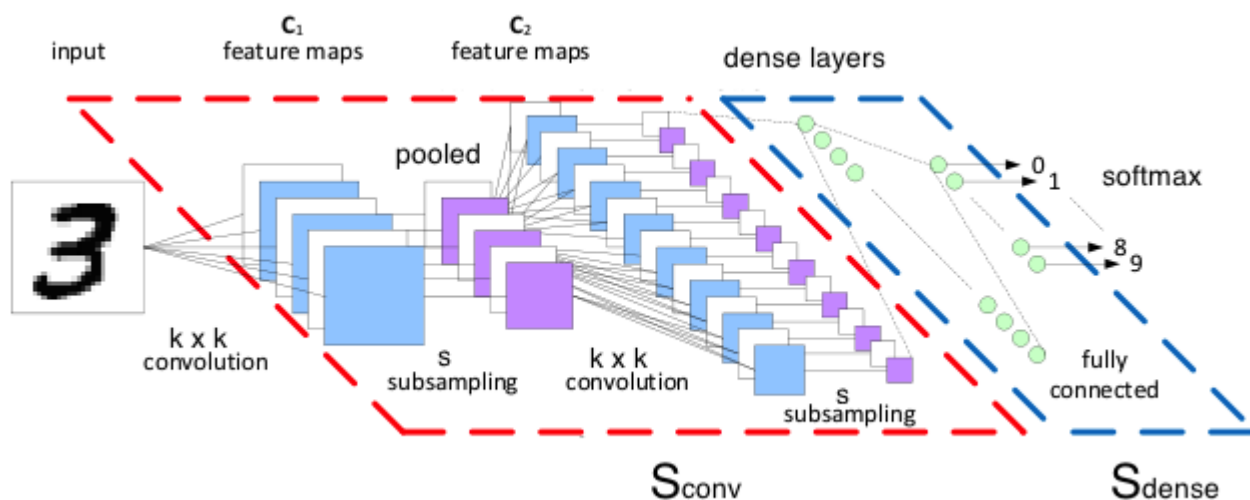
## **Project Design**

The first step of the project will be loading the data and training the model with tensorflow and Keras.

Then we need to save the model which can be later used in app.

Now to create the simple app we used Android studio. The app has two simple things (a) where you can draw a digit on a canvas, (b) press the detect button to predict the image.

To train the model we have used convolutional neural network, an architecture of convolutional neural network is given below.




Workflow map :



## References

1. ["Support vector machines speed pattern recognition - Vision Systems Design"](#). Vision Systems Design. Retrieved 17 August 2013.
2. Gangaputra, Sachin. ["Handwritten digit database"](#). Retrieved 17 August 2013.
3. Qiao, Yu (2007). ["THE MNIST DATABASE of handwritten digits"](#). Retrieved 18 August 2013.
4. Platt, John C. (1999). ["Using analytic QP and sparseness to speed training of support vector machines"](#) (PDF). Advances in Neural Information Processing Systems: 557-563. Retrieved 18 August 2013.
5. [Jump up to: abcdef](#) LeCun, Yann; Corinna Cortes; Christopher J.C. Burges. ["MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges"](#). Retrieved 17 August 2013.
6. Kussul, Ernst; Tatiana Baidyk (2004). "Improved method of handwritten digit recognition tested on MNIST database". Image and Vision Computing. 22(12): 971-981. [doi:10.1016/j.imavis.2004.03.008](#).
7. Zhang, Bin; Sargur N. Srihari (2004). ["Fast k -Nearest Neighbor Classification Using Cluster-Based Trees"](#) (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 26(4): 525-528. [PMID15382657](#). [doi:10.1109/TPAMI.2004.1265868](#). Retrieved 18 August 2013.
8. [Jump up to: abc](#) Ciresan, Dan; Ueli Meier; Jürgen Schmidhuber (2012). ["Multi-column deep neural networks for image classification"](#) (PDF). 2012 IEEE Conference on Computer Vision and

Pattern Recognition: 3642–3649. [ISBN978-1-4673-1228-8](#). [arXiv:1202.2745](#)   
[.doi:10.1109/CVPR.2012.6248110](#).

9. [Jump up to:abcd](#)LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "[Gradient-Based Learning Applied to Document Recognition](#)"(PDF). Proceedings of the IEEE 86.86(11): 2278–2324. [doi:10.1109/5.726791](#). Retrieved 18 August 2013.