Project Report

Machine Learning Project 3

Ahmed Hassan
Ahmad Ayaz
Muhammad Faiq

Supervisor: Dr. Malik Jahan Khan

Abstract

This document is a report of the Project 3 for the course CS - 464 (Machine Learning). This report summarizes the problems statements, implementation and its techniques, challenges and the results of the overall project.

Contents

Acronyms **2**

1 Introduction **3**
1.1 Problem Statement 3
1.2 Objectives 3

2 Artificial Neural Networks **4**
2.1 Introduction 4
2.2 Building Block 4

3 Pre-Processing **5**
3.1 ROI 5
3.2 Adaptive Thresholding 5

4 Implementation **7**
4.1 Dataset Preparation 7
4.1.1 Using ROI 7
4.1.2 Using Adaptive Thresholding 8
4.2 Predictor Models 8
4.3 Testing Procedure 8
4.4 Results 9
4.5 Challenges 9

Acronyms

ANN Artificial Neural Networks

CNN Convolutional Neural Networks

HOG Histogram of Oriented Gradients

ML Machine Learning

MNIST Modified National Institute of Standards and Technology database

ROI Rectangle of Interest

Chapter 1

Introduction

In today's modern world everything revolve around computer and smart devices. Everywhere there is automation which is making human life easy and care free. But computers needs instruction and automation devices are sensor based. They don't know what to do if a similar situation comes up.

This is where machine learning comes in. Humans can't monitor the process 24/7 because it is hectic and what's the advantage of having so much technology if it can't make their own decisions. Machine learning is basically about training machine (computer) to perform tasks like humans in their absence.

1.1 Problem Statement

This project is based on the Modified National Institute of Standards and Technology database (MNIST) available as the part of TensorFlow / Keras datasets. We had to extend the MNIST standard implementation to recognize hand written digits to develop the following features.

- To gather our own data of hand-written digits (at least 200 unique samples).
- to test the MNIST implementation on your data.
- To make our program able to convert an image containing an integer of up to 5 digits into equivalent number. For example, 45321, 88881, 23019 etc

1.2 Objectives

The objective of this project was to get familiar with working on Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) implementations and other pre-processing techniques in solving and developing solution for a real life problem.

Chapter 2

Artificial Neural Networks

2.1 Introduction

The term ANN is relatively newer to the computer science world. ANN can be defined as the computational tools that are being extensively used in solving many real life day to day problems. The ANN are best recognized because of their remarkable information processing capabilities. We will just have a bird's eye view of the workings of ANN

2.2 Building Block

The ANN were made from the inspiration of a biological neuron such as the ones found in human brain. The artificial neurons, also called perceptrons, are the basic building block of an ANN. A hypothetical structure to help in comprehension is given in the figure 2.1

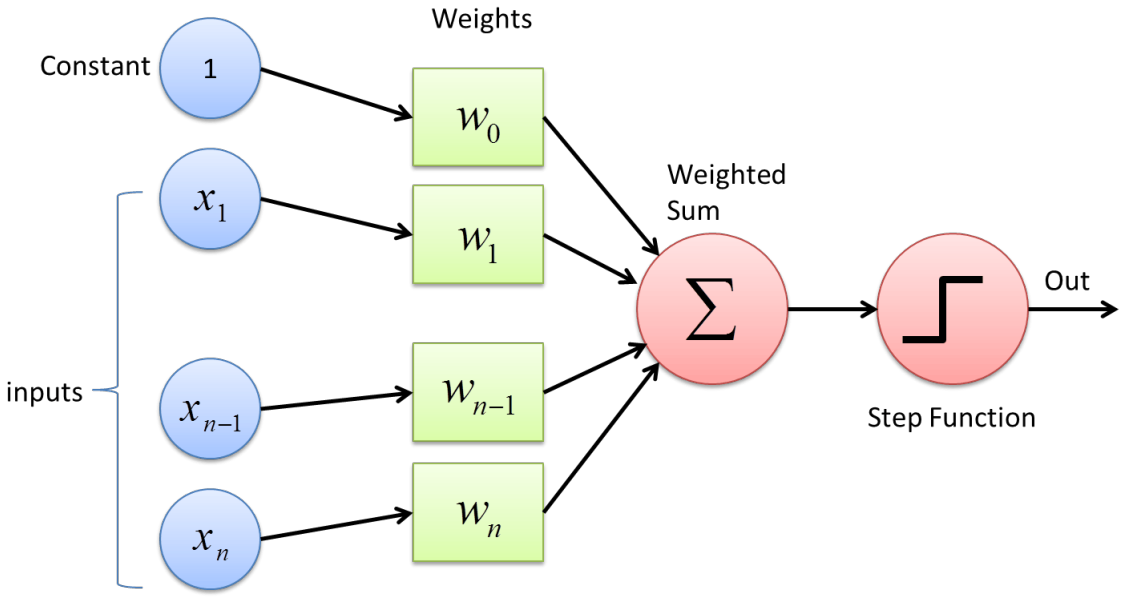

Figure 2.1: A Perceptron[1]

The figure 2.1 explains the working of a perceptron. The inputs are fed to the perceptron in parallel along with a "bias" value on which the perceptron performs weighted sum and passes that to a step function that maps the output between a desired limit to produce an output. A combination or arrangement of many such perceptrons forms a ANN. The details of the working of the perceptrons and ANN are beyond the scope of this report however it be read about in detail here[2].

Chapter 3

Pre-Processing

This chapter gives a brief explanation of the pre-processing techniques used in our project like Rectangle of Interest (ROI) and Adaptive Thresholding.

3.1 ROI

Region of Interest, as the name states it is the desired we want to work on. Sometimes we have large amount of data and we want to work on specific region or detail of the given data. For this purpose in image processing we use ROI, which take specified region of the picture for processing, out of the whole picture. This all happens on the run time, we don't have to separate detail out before processing.

3.2 Adaptive Thresholding

In thresholding RGB or Gray Scale images are converted into Black and White images. Thresholding has three types: Global, local and adaptive. All of the mentioned has same outcome but differ in the methodology of working.

In Adaptive thresholding the image is converted according to the thresholds of neighboring pixels. Threshold in adaptive thresholding vary over the image. It depends upon the distribution of histogram. When it is neither uniform nor clustered separately, then we apply adaptive thresholding with multiple thresholds, depending upon the values of pixels for an example consider the figure 3.1.

Figure 3.1: An example image

The histogram made would not be uniform and it would spread over the whole intensity. It is difficult to use global thresholding because it will not convert the image accurately. The intensity histogram is shown in figure 3.2 on the next page.

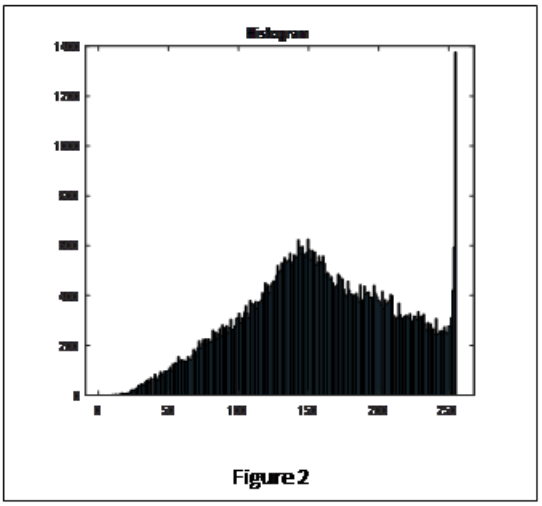

Figure 3.2: Intensity histogram for figure 3.1

For the conversion of image in to Black and White image in this kind of scenario, we use multiple techniques. One of them is Adaptive Thresholding. In this technique thresholds are selected according to the neighbors of input pixels. In simple words, threshold depends upon the neighboring pixels and different thresholds will be selected for different regions of the same image. The output is a Black and White image, converted using adaptive thresholding, which can be seen in Figure 3.3.

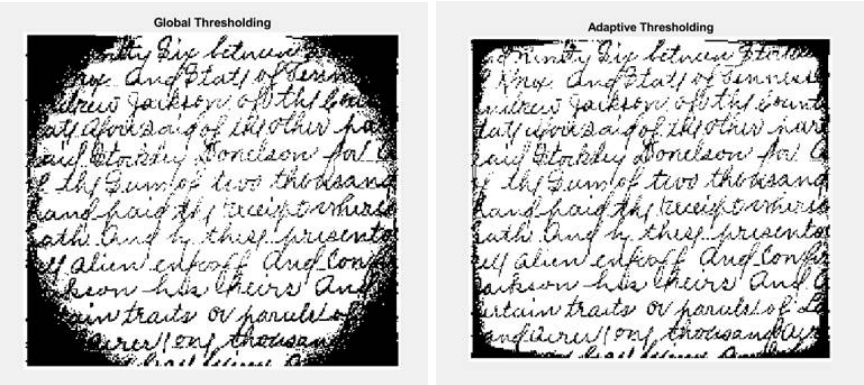

Figure 3.3: comparison of Adaptive vs Global Thresholding

Chapter 4

Implementation

4.1 Dataset Preparation

We started by taking photographs of hand written digits (up to five digits) with smartphones app FV-5 Lite [3]. This app allows the adjustment of all the photographic parameters manually, such as exposure compensation, ISO, light metering mode, focus mode, white balance and program mode. An example is shown in the figure 3.1.

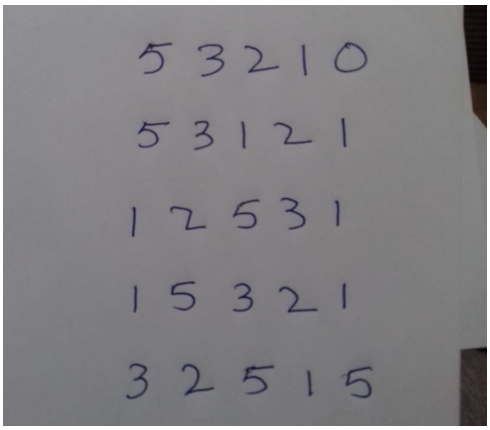

Figure 4.1: Example of initial stage of Dataset fabrication

4.1.1 Using ROI

We then proceeded over to crop the individual numbers to have one number in an image. For this purpose we manually selected the numbers by using the ROI from OpenCV. each number was manually selected from an image and was separated as shown in the figure 4.2. The figure shows the use of ROI to select the desired area from the image and these images were converted into black and white images.

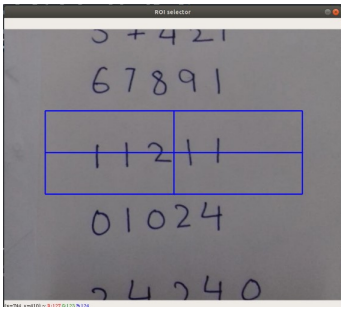

Figure 4.2: Using ROI to get desired part [4]

4.1.2 Using Adaptive Thresholding

Even though we had taken images using FV-5 Lite app, still some processing was required. For this purpose we used Adaptive Thresholding. This thresholding algorithm explained earlier in section 3.2, converted the image into Black and White image. The thresholding was don manually by adjusting a slider as shown in the following figure 4.3.

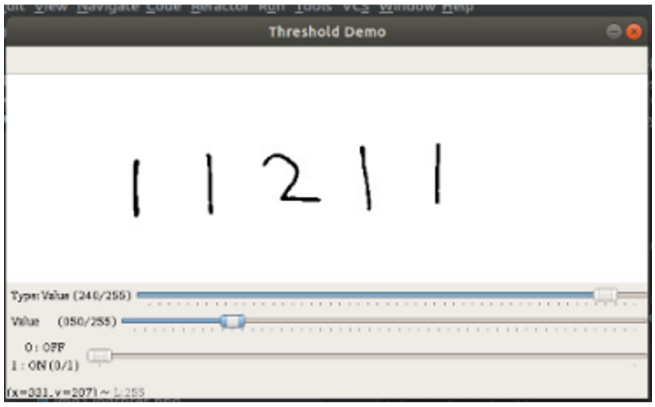

Figure 4.3: Adaptive thresholding [5]

After this the labels for these images were generated manually and stored in a numpy array to be used for the purpose of verification later.

4.2 Predictor Models

We used 2 different predictor models for a detailed analysis, namely SVM and CNN [6] both trained with the entire MNIST dataset with 80 - 20 train to test ratio.

4.3 Testing Procedure

The data prepared as explained in the earlier sections is then tested. We started by fetching the images in our data set and converted them in grey-scale images and applied Gaussian filter after which we applied the thresholding on these images and found out any contours in the image. These contours are sorted according to the position on the image [7]. Rectangles are made about these contours. To this point the testing procedure is same for both the predictor models.

The following flowchart summarises the whole working of our implementation.

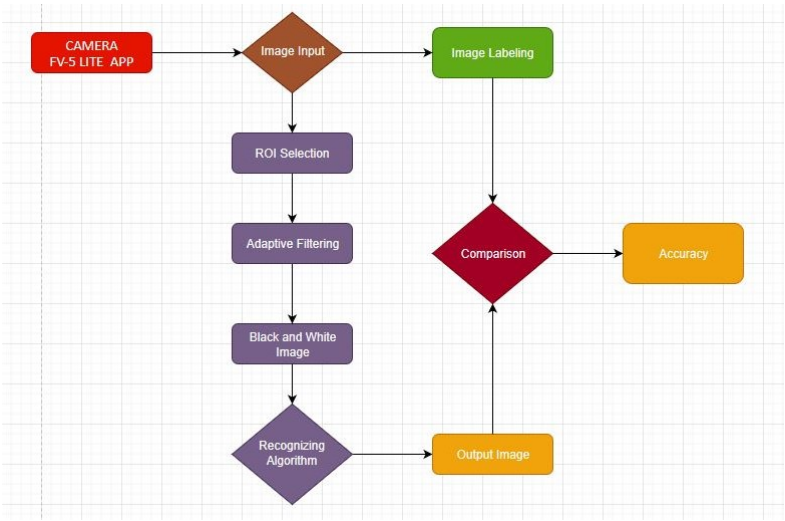

Figure 4.4: Flowchart

For using the SVM, we apply Histogram of Oriented Gradients (HOG) feature extractor on each of these rectangles with the contours and each one is passed to the trained SVM model that returns the predicted number. Which is then compared with the label against the image fed to the model to get the output if the prediction was made correctly.

For using the CNN model, we simply feed the rectangles with contours to the trained model which in turn returns the predicted number. This output is then compared with the label against the image fed to the model to get the output if the prediction was made correctly.

4.4 Results

We ran the tests for each model a hundred times on the same 204 images in our dataset. The accuracy achieved with CNN was **96.56%** and the time it took to execute was **78.78** seconds. While on the other hand, SVM did not achieve a decent level of accuracy standing at **54.47%** in a time of **2.41** seconds.

4.5 Challenges

The main challenge we had to face was the removal of background while preparing the dataset as each of the pictures may have different resolutions, sharpness and other such properties. So to overcome it we applied the thresholding manually. Another challenge was in the recognition of the number 7. If the number was slashed the predictor models recognized it as 3 and if the 7 was slanted a little to the left, it was recognized as number 1. Another such ambiguity arises if the digit 8 is written with larger circles it usually gets predicted as two zeros. Also if the digit 3 is written in slanting hand, it gets predicted as digit 8.

Bibliography

[1] Hypothetical structure of a perceptron [Diagram] 2017
towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

[2] JAIN, A. K., MOHIUDDIN, K. M. AND MAO, J.
Artificial neural networks: a tutorial - IEEE Journals & Magazine. IEEE, 10.1109/2.485891.

[3] CAMERA FV-5 LITE app for setting same focus,sharpness,resolution etc of each image taken from mobile camera
https://bit.ly/2sQm2h1

[4] How to select a bounding box (ROI) in OpenCV
https://www.learnopencv.com/how-to-select-a-bounding-box-roi-in-opencv-cpp-python/

[5] Trackbar as the Adaptive Threshold Values
https://www.learnopencv.com/how-to-select-a-bounding-box-roi-in-opencv-cpp-python/

[6] MNIST CNN Model
https://keras.io/examples/mnist_cnn/

[7] Python opencv sorting contours
https://stackoverflow.com/questions/39403183/python-opencv-sorting-contours