**PROBLEM DESCRIPTION**

* Before optical character recognition can be used, the source material, that is the “GE’EZ NUMBERS” must be scanned using an optical scanner to read in the page as a bitmap (a pattern of dots).

Software to recognize the images is also required. The character recognition software then processes these scans to differentiate the “GE’EZ NUMBERS” and determine what are represented in the light and dark areas. The OCR engines add the multiple algorithms of neural network technology to analyze the stroke edge, the line of discontinuity between the characters, and the background. Allowing for irregularities of printed ink on paper, each algorithm averages the light and dark along the side of a stroke, matches it to known characters and makes a best guess as to which character it is. The OCR software then averages or polls the results from all the algorithms to obtain a single reading. OCR software can recognize a wide variety of fonts, but handwriting and script fonts that mimic handwriting are still problematic, therefore additional help of neural network power is required.

Neural networks can be used, if we have a suitable dataset for training and learning purposes. Datasets are one of the most important things when constructing new neural network. Without proper dataset, training will be useless. So first we have to scan the numbers. After that, we will define processing algorithm, which will extract important attributes from the numbers and map them into a database. Extracted attributes will have numerical values and will be usually stored in arrays. With these values, neural network can be trained and we can get a good end results. The problem of well defined datasets lies also in carefully chosen algorithm attributes. Attributes are important and can have a crucial impact on end results. The most important attributes for handwriting algorithms are:

1. Negative image of the number, where the input is defined as 0 or 1. 0 is black, 1 is white, values in between shows the intensity of the relevant pixel.

2. The horizontal position, counting pixels from the left edge of the image, of the center of the smallest rectangular box that can be drawn with all "on" pixels inside the box.

3. The vertical position, counting pixels from the bottom, of the above box.

4. The width, in pixels, of the box.

5. The height, in pixels, of the box.

6. The total number of "on" pixels in the number image.

7. The mean horizontal position of all "on" pixels relative to the center of the box and divided by the width of the box. This feature has a negative value if the number is "left heavy".

8. The mean vertical position of all "on" pixels relative to the center of the box and divided by the height of the box.

9. The mean squared value of the horizontal pixel distances as measured in 6 above. This attribute will have a higher value for images whose pixels are more widely separated in the horizontal direction.

10. The mean squared value of the vertical pixel distances as measured in 7 above.

11. The mean product of the horizontal and vertical distances for each "on" pixel as measured in 6 and 7 above. This attribute has a positive value for diagonal lines that run from bottom left to top right and negative value for diagonal lines from top left to bottom right.

12. The mean value of the squared horizontal distance tunes the vertical distance for each "on" pixel. This measures the correlation of the horizontal variance with the vertical position.

13. The mean value of the squared vertical distance times the horizontal distance for each "on" pixel. This measures the correlation of the vertical variance with the horizontal position.

14. The mean number of edges (an "on" pixel immediately to the right of either an "off pixel or the image boundary) encountered when making systematic scans from left to right at all vertical positions within the box.

15. The sum of the vertical positions of edges encountered as measured in 13 above. This feature will give a higher value if there are more edges at the top of the box.

16. The mean number of edges (an "on" pixel immediately above either an "off pixel or the image boundary) encountered when making systematic scans of the image from bottom to top over all horizontal positions within the box.

17. The sum of horizontal positions of edges encountered as measured in 15 above.

Methodology

* The way to develop this system it may have a lot of schemes and ways.But the approprate methode that we choise is as follows:-
* Fist we preprocessed the input data to become approprate formate
* After that we choise the approprate achitecture and learning algorith which is convient for this problem
* Then the preprocessed data should we divided in to two partes. That means two third (2/3) the input data for traing and one third (1/3) of the data for testing

Detail Solution approach for the problem is as follow

* To solve the defined handwritten character recognition problem of classification we used MATLAB computation software with Neural Network Toolbox and Image Processing Toolbox add-on. And
* The computation code is divided into the next categories:

**Automatic Image Preprocessing**

* The image is first being converted to grayscale image follow by the threshing technique, which make the image become binary image. The binary image is then sent through connectivity test in order to check for the maximum connected component, which is, the box of the form. After locating the box, the individual characters are then cropped into different sub images that are the raw data for the following feature extraction routine.
* The size of the sub-images are may not fixed since they are expose to noises which will affect the cropping process to be vary from one to another. This will causing the input of the network become not standard and hence, prohibit the data from feeding through the network. To solve this problem, the sub-images have been resize by certain value and then by finding the average value in each by certain size blocks, the image can be down to N by M matrices, with fuzzy value, and become L inputs for the network. However, before resize the sub-images, another process must be gone through to eliminate the white space in the boxes. That means
* The we have to Read Image through matlab space and Convert to grayscale image
* Here ,Now the greate work is Convert the image in to binary image
* Then after we perform the rest of procedure we start to train the network and testing

# CONCLUSION

* Handwritten character recognition is a complex problem, which is not easily solvable. There might be a lotof problem lies in pre-processing of data.
* Described application of character recognition can be divided into three main parts. Image preprocessing to get the training data, training the neural network and at the end testing with final recognition results.
* Image preprocessing to get the training data for the neural network is based on input training.
* Training and testing the neural network was only a matter of two MATLAB commands. We decided to use Multilayer Perceptron with two hidden layers .