Document and Code by: **Madhusree** **Rana**

ALGORITHMS USED TO SOLVE THE PROBLEM

IDEA AND MATHEMATICS BEHIND THE PROBLEM:

The human brain has no problem recognizing a sloppily written "3" of low resolution, but it’s not the same in the case of computers. When we see a digit or alphabet written in hand by different people or typed in different fonts such as “Calibri”, “Times New Roman”, or “Monotype Corsiva”, our brain resolves them as representing the same idea, but in the case of a computer, it has to “learn” to recognize different digits and alphabets from huge datasets, and this is where “***convolutional neural networks (CNN)”*** comes into play.

For Optical Character Recognition (OCR), before applying the CNN, the image is at first, pre-processed. The color image is first converted into ***grayscale***. This is helpful especially because it is easier to work with an image with one channel than with that having three channels. After this, the image is ***filtered*** to make it noise-free. Finally, ***thresholding*** is implemented to convert it to a binary image.

CNN plays the most vital role in OCR. It works on multiple layers of neurons which hold a number specifically between 0 and (+)1. For example, in the case of identifying digits, for a 28 x 28 binary image consisting of a single digit “9”, the network starts with a bunch of neurons corresponding to each of the 28 x 28 pixels of the input image, which is 784 neurons in total, each holding a floating point number between 0 and 1, representing the grayscale value of the corresponding pixel, ranging from 0 for black pixels up to 1 for white pixels. This number inside the neuron is called its “***Activation***”, which is closer to 1 when the pixel is brighter, and vice versa. These 784 neurons make up the first layer of the network.

Jumping to the last layer, it has 10 neurons, each representing one of the digits between 0 and 9. The activation in these neurons is again some floating point number between 0 and 1, representing how much the system thinks that the given image corresponds to a particular digit. There are also a couple of layers in between, called the “***hidden layers***”, each one of which help in mapping the current information to the correct neurons of the next layer. The hidden layers behave more intelligently by doing critical analysis of the pattern by breaking it down into sub-components such as loops and lines which serve as small edges. Finally, the brightest neuron in the last layer, determines which digit the image corresponds to.

Now, this neural network actually works like a weighted graph, where each neuron, behaving as a node, holds the activation number, while each edge joining the neurons is assigned a weight which is nothing but a number. In each layer, a weighted sum is computed between these weights and the activation numbers. The weighted sum, which serves as the activation for the next layer, is then pumped into some function on the real number line into the range between 0 and 1, which is done by either the “***Sigmoid***” function, or the “***Rectified Linear Unit (RELU)***” function.

For computation, the activations are all put together in a column matrix, and the weights are arranged into another two-dimensional matrix where each row corresponds to the connection between one layer and a particular neuron in the next layer, and finally the matrices are ***multiplied*** to get the activations for the next layer. So basically, the entire network is a function, which takes a total of 784 numbers as input and spits out one number as the output.

IDEA, LANGUAGE AND MODULES USED FOR THE CODE:

1. The programming language used for the coding is ***Python***.
2. The modules used for this purpose are:
3. (for the video):

* Open Source Computer Vision (***OpenCV***)
* ***Pytesseract***, which uses Google’s Tesseract OCR Engine
* ***Os***, for managing the files and folder for storing the files

1. (for the image):

In addition to the above modules, the following :-

* ***PyAutogui*** for taking screenshots
* ***Tkinter*** for creating a button to take screenshots

EXPECTED RESULT

(For the video):

The program is expected to enable the playing of a stored video, from which, the user would be able to take screenshots as per their wish, as many times as wished.

(For the image):

The program is expected to enable the user to take screenshots as many times as wished using a “Take ss” button which should preferably be shifted to a corner of the screen while taking the screenshot.

For both the above cases:

The screenshots taken would be stored inside an empty folder, and whenever a screenshot would be taken, the text embedded in that image would be extracted immediately and all the text would be appended and stored inside a single text file in the same folder where the screenshots would be stored. All text from a particular image can be distinguished as there would be a “page number” which would correspond to the serial order of the screenshot taken.