Abstract – In this paper, we have proposed a system to grade CAPTCHA strength, according to various parameters determined by breaking the CAPTCHA. There are basically four major steps involved in the system. First step is preprocessing which includes – binarization and noise removal. Second step is segmenting the image into individual characters. Third step is to obtain skeletons of characters i.e., Thinning. The final step is character recognition. For character recognition we have used artificial neural networks.

I. INTRODUCTION
CAPTCHA, which stands for “Completely Automated Public Turing test to tell Computers and Humans Apart”, is a type of challenge-response test used in computing as an attempt to ensure that a person generates the response. The process usually involves one computer (a server) asking a user to complete a simple test, which the computer is able to generate and grade. Because other computers are assumed to be unable to solve the CAPTCHA, any user entering a correct solution is presumed to be human. Thus, it is sometimes described as a reverse Turing test, because it is administered by a machine and targeted to a human, in contrast to the standard Turing test that is typically administered by a human and targeted to a machine. An artificial neural network (ANN), usually called neural network (NN), is a mathematical model or computational model that is inspired by the structure and/or functional aspects of biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modelling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data. Using the neural networks the characters in the CAPTCHA image can be recognized using the various algorithms and the strength of the CAPTCHA can be measured.

II. PREPROCESSING
In this step, CAPTCHA image is reduced to two colour image and cleaned.

1. Binarization
Binarization is the process of converting a colour (or grayscale) image into a binary colour image i.e., black and white. A binary image simplifies further processing of image.

Generally, binarization is done with a single threshold value, say x, such that all the colours above x are truncated to black and rest to white colour or vice versa. Not only it is essential to determine ‘right’ value for x, a single value binarization proves insufficient for the required results. In the following text, we have proposed our algorithm, which not only determines value for x on its own but also dynamically decides how many threshold values, such as x, are required. This algorithm is able to handle multi-colour background as well as multi-coloured characters.

Assumption:

a. Number of pixels of (same or similar, explained later) colour in background always dominates number of pixels in characters to be recognized.

b. There is usually colour difference of at least 40 levels between foreground colours and background colours.
Steps involved in Binarization-

i. A colour image (24-bit RGB) is converted into grayscale image (8-bit) using following formula: 
   \[ \text{Grey colour} = (0.299 \times R + 0.587 \times G + 0.114 \times B) \]
   This essentially discards all colour information but keeps luminosity of pixel preserved.

   ![Fig.1](image)

   (a) Colour Image (b) Grey scale image only colours discarded (c) Colour 255 replaced with white (d) Colour 233 replaced 233 with white

ii. Whole image is scanned for the next dominating colour.
iii. That colour becomes the seed colour and all the neighbouring colours whose pixel count is greater than or equal to the eight per cent (experimental) of that of seed colour are grouped together as similar colours. They are stored in an array, similarColor.
iv. All the colours stored in similarColor array are replaced with white colour.
v. Cycle repeats (go to step 2) for next the dominating colour if it lies close (experimental, 40) to the previous seed colour.
vi. Once more a dominating colour is searched, this happens to be the colour of characters. This seed colour along with neighbouring colours is replaced with black.

2. Noise Removal

Lines are the major part of the noise present in most of the CAPTCHA images. If not removed, they can create an enormous amount of ambiguity for the recognition. These lines can be straight, curved or even broken. Detection and removal of such lines has been very challenging since the pattern of line changes for every image. The proposed algorithm aims at removing the lines assumes that the thickness of the line is far more less than that of a character, which is true for most of the cases. The algorithm scans through the image vertically for each column and as soon as the first black pixel is found the thickness is calculated. If the thickness appears to be less than the threshold then the unit is considered as a part of line, which is set to the white colour. In this way when the whole image is scanned completely all the lines with the thickness less than the threshold and coloured white i.e. removed.

Algorithm

For each column
   For each row
      If black pixel found
         Find the thickness
         If thickness < threshold
            Replace all these black pixels to white
         End if
      End if
   End for

3. Dot Removal

Some CAPTCHA images contain noise in the form of small dots spread all over the image which create ambiguity at the time of recognition. Thus before the image is ready for segmentation these dots must be removed. The proposed algorithm tries to detect the dots by checking each pixel individually. All the eight neighbours of the pixel are checked and if the black pixel is surrounded by more than 4 white pixels it is considered as a dot. Thus this pixel colour is changed to white.
Algorithm
For every pixel in image
    If number of white neighboring pixels > 4
        Change pixel color to white
    End if
End for

4. Discontinuity Removal
One of the commonly observed imperfections in CAPTCHA image is the discontinuities in the strokes of character. Many of the times, the CAPTCHA image comes with discontinuous character and a many a times this discontinuities are formed after some pre-processing. So it is important for us remove this discontinuities to regain the basic frame of the character. This is done using the discontinuity removal methods.

Algorithm
A character image can contain both horizontal as well as vertical discontinuities, and both should be removed. To remove the vertical discontinuities in the character following steps are followed:
   a. Traverse each row of the image from top to bottom along its length.
   b. When you encounter a white pixel put it in stack
   c. If you encounter a black pixel
      i. Count the number of pixels in the stack
      ii. If the number is less then threshold(say 4), pop those pixels and repaint them into white
      iii. Clear the stack
   d. Repeat

   We have considered that a continuous stretch of 4 white pixels can only be part of a discontinuity.

To remove horizontal discontinuities we will follow the same technique, but this time traversing each column of image along its width.

Removing discontinuities aid a lot in the character recognition process, as we are able to make the character structure continuous, and thinning giving us the exact skeleton of the character, the job of the character recognizer becomes easy.

III. PROJECTION BASED SEGMENTATION
After the various noises removal algorithms clean the image, it is sent to the segmentation phase so that the characters from the image can be individually recognized. The method that has been used for segmentation divides the image into different units based on the projections, thus the name projection based segmentation. The algorithm is based on the fact that the count of the black pixels between the characters is very low and in some cases it may be zero. To calculate the projections the image is scanned vertically and for each black pixel the count is incremented. After the whole image is done with the scan the projections are checked upon so that the image could be segmented.

Algorithm:
For each column
    For each row
        If black pixel found
            Increment the count
        End if
    End for
End for

Fig.3 (a) CAPTCHA image with dot image (b) Result after dot removal

(a)
(b)

Fig.4 (a)Input image with discontinuities(b) After applying vertical discontinuity removal method (c) After applying vertical discontinuity removal method

(a)
(b)
(c)
The above algorithm gives the count of the black pixels for each column, which is nothing but the projection, based on which the image is segmented.

![Image](a) CAPTCHA Image (b) CAPTCHA Image with segmentation

**IV. THINNING**

Characters obtained after segmentation come in various shapes and sizes. To lessen the burden of Neural Network and for more accurate result skeletonization of characters is carried out, we call this step as thinning of characters. We have implemented fast thinning algorithm proposed by T.Y. Zhang and C.Y. Suen. This procedure is carried out in two steps, in the first step, southeast boundary and, in the second step, northeast boundary of the characters is erased. While removing the boundary pixels care is taken that the basic skeleton of the character is unharmed and the strokes of character don’t get disconnected.

**Algorithm**

To determine which pixels should be removed during the thinning processes following assumptions are made:

a. The binarized character image is defined in a matrix IT where value of pixel TI(i,j) is either 0 or 1.(1 when colour of pixel is black and 0 when it is white).

b. Transformation of any pixel in IT depends on values of small set of neighboring pixels. Neighbours of point (i,j) are (i-1,j), (i-1,j+1), (i,j+1), (i+1,j+1), (i+1,j), (i+1,j-1), (i,j-1) and (i-1,j-1) as shown in figure below:

<table>
<thead>
<tr>
<th>P9 (i-1,j-1)</th>
<th>P2 (i-1,j)</th>
<th>P3 (i-1,j+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P8 (i-j-1)</td>
<td>P1 (i,j)</td>
<td>P4 (i,j+1)</td>
</tr>
<tr>
<td>P7 (i+1,j-1)</td>
<td>P6 (i+1,j)</td>
<td>P5 (i+1,j+1)</td>
</tr>
</tbody>
</table>

c. During computation, the new value at a point during nth iteration depends on its own value as well as values of those of its eight neighbours after (n-1)th iteration

We have used a 3x3 window for computation whereas some algorithms use a 4x4 window.

The basic idea to perform thinning is to remove all the pixels except those, which form the skeleton of the character. One should take care that the skeletal structure of character does not get disconnected. This is achieved by applying the conditions given by Hilditch algorithm.

A point P1 is deleted from the image if it satisfies following conditions:

a. 2 <= B (P1) <= 6  
b. A (P1) = 1  
c. P2 x P4 x P6 = 0  
d. P4 x P6 x P8 = 0

Point P1 is deleted if and only if all the four conditions are satisfied.

Where A (P1) is number of 01 patterns observed in ordered pair P2, P3, P4,…,P8, P9.

In above figure A (P1) = 2.  
B (P1) is number of non-zero neighbours of P1  
B (P1) = P2+P3+P4+P5+P6+P7+P8+P9

In above figure B (P1) = 3  
Conditions ‘c’ and ‘d’ make sure that only southwest boundary of the image is removed.

To remove the northeast boundary conditions ‘a’ and ‘b’ are the same but ‘c’ and ‘d’ are slightly modified, now the new conditions are:

a. 2 <= B (P1) <= 6  
b. A (P1) = 1  
c. P2 x P4 x P6 = 0  
d. P4 x P6 x P8 = 0

Condition ‘a’ guarantees that endpoints of skeleton lines are preserved whereas condition ‘b’...
V. RECOGNITION

Character recognition is a trivial task for humans, but to make a computer program recognize characters is a difficult task. Recognizing patterns and different characters is one of those things that humans do well and computers don’t. The reason for this maybe, the high level of abstraction; there are thousands styles of type in common use and a character recognition program must recognize most of these to be of any use. There exist several different techniques for recognizing characters. A common technique uses back-propagation in a neural network which is been implemented in this project.

Artificial Neural Network

The ANN used in our project consists of 3 layers:-

1. Input Layer
   The input to the neural network is given via the ‘Input Layer’. In this project the input data to the input layer is segmented characters.

2. Hidden Layer
   This layer is called ‘Hidden Layer’ because it is invisible to the user and is only required to manipulate the input to generate a specified output.

3. Output Layer
   This layer serves as the output to the user. In our project, the output is nothing but the recognized character.

   The Neurons present in different layers are connected to each other via a scalar quantity also called as ‘Weights’.

   Back Propagation is used to reduce the error in recognizing a character.

This step involves comparing the pattern of the input character against that of the stored character and finding out matching probability.

We compute I and M matrices from segmented characters and store in a text file. For various possible variants of same character, we add M matrix to the one which is stored in file of that character. Thus we have several patterns of the same character stored in the same file. The matrix stored in file is W matrix.

The recognition of patterns is now done on the basis of certain statistics. First we calculate I matrix which has value 0 for white and 1 for black pixels in image, then we calculate M matrix by replacing 0 with -1. These matrices are further used to find out matching probability. The character with maximum matching probability is selected as output.

Neural Network Architecture

I Matrix

This matrix represents the original segmented character image containing zeros for white background and ones for black text.

M matrix

This matrix is derived from I matrix just by replacing zeros by -1.

Pseudo code-

\[
\text{If } I(i,j) = 1 \\
\quad \text{Then } M(i,j) = 1 \\
\text{Else: } \\
\quad \text{If } I(i,j) = 0 \text{ Then } M(i,j) = -1
\]

Weight matrix

For a character gets initialized by the first character pattern that is taught to the system. It then gets updated every time a new character pattern is taught to system.
Pseudo code

for all i=1 to x
{
  for all j=1 to x
  {
    W_k(i,j) = W_k(i,j) + M(i,j)
  }
}

Candidate Score

This statistic is a product of corresponding elements of the weight matrix \( W_k \) of the k\textsuperscript{th} learnt pattern and an input pattern \( I \) as its candidate. It is formulated by taking bit by bit product of \( I \) and \( W \) matrices and adding the results.

\[
\Psi(k) = \sum_{i=1}^{x} \sum_{j=1}^{y} W_k(i,j) \ast I(i,j)
\]

Ideal-Weight Model Score

This statistic simply gives the sum total of all the positive elements of the weight matrix of a learnt pattern. It may be formulated as follows-

Pseudo code

for i = 1 to x
{
  for j = 1 to y
  {
    if \( W(i,j) \) > 0 then
    {
      \( \mu(k) = \mu(k) + W(i,j) \)
    }
  }
}

Recognition Quotient

This statistic gives a measure of how well the recognition system identifies an input pattern as a matching candidate for one of its many learnt patterns. It is simply given by:

\[
Q(k) = \frac{\Psi(k)}{\mu(k)}
\]

The greater the value of Q(K), the more confidence does the system bestow on the input pattern as being similar to a pattern already known to it.

VI. SPECIAL CAPTCHAs

Besides normal type of CAPTCHAs we have also considered some special types of CAPTCHA.

1. Hollow Characters

Hollow CAPTCHAs are the ones with hollow characters. Such characters must be made solid so that the projection based segmentation works correctly. To do this boundary fill algorithm is used. Firstly the background is filled with the black colour, which results in an image with only the characters in the white pixels. Then the image is inverted which gives the black characters with white background, an ideal input for the segmentation. The basic assumption of this algorithm is that the noise such as lines and dots don’t interfere when the boundary fill algorithm runs for the background. To overcome the boundary fill is applied rigorously to the whole image so that the whole background is covered irrespective of the noise.

Fig.8 (a) CAPTCHA image with hollow characters (b) After Boundary fill (c) After Inverting Colours

Fig.9 (a) Chequered Image (b) Processed Image

1. Chequered

In chequered CAPTCHA, whole image is composed of alternating blocks of colour (here black and white).
To obtain normal image i.e., white background with black foreground, dimensions of top left block are measured. Then colours of blocks are alternatively inverted depending on the background colour of first block.

VII. APPLICATIONS AND LIMITATIONS

An effective CAPTCHA breaker is one which can recognize all characters in CAPTCHA image correctly. CAPTCHA is the only security mechanism used to differentiate between bots and humans. If a CAPTCHA breaker, it is vulnerable to other bots with malicious intentions. Thus using our system we are able to grade CAPTCHA and expose its susceptibility.

Also depending on the results, we can infer which type of noises is more difficult to handle and which ones are easy. We can ask websites to use this to improve the strength of the CAPTCHA.

As we have used neural networks for character segmentation, our system can continuously learn new patterns of character thus increasing the probability of accurate recognition. Although we have considered most of the noise present in CAPTCHA and have efficiently removed them. Still we are unable to handle few of them. Also connected characters create a problem during segmentation. Hence our CAPTCHA breaker cannot break images with connected characters.

VIII. CONCLUSION

We have proposed a system to grade CAPTCHA based on their strength. Factors like noise present in the image, time taken to break CAPTCHA & number of characters recognized is taken into consideration. Different types of CAPTCHAs are handled viz. CAPTCHAs with dots and lines, Chequered CAPTCHAs and Hollow CAPTCHA. ANN is used for Character Recognition in which we have used typical concepts like machine learning and training of characters. It helps our system to adapt itself in learning different patterns and orientation of various characters. Our system will help websites to know the strength of the CAPTCHA used by their servers to protect attacks against bots.

IX. REFERENCES

[1] Ahmad El Ahmad, Jeff Yan, Wai-Yin Ng “CAPTCHA design: colour, usability and security,” IEEE Internet Computing, 1089-7801, Digital Object Identifier 10.1109/MIC.2011.102
[5] Shashank Araokar “Visual Character Recognition using Artificial Neural Networks”