1. INTRODUCTION

This chapter gives an introduction to the project entitled ‘deCAPTCHA – A CAPTCHA Solver and Analysis Tool’. Brief information about the abstract, motivation of the project, problem statement and scope of the project has been provided in this chapter. This is followed by the organization of the entire project report.

1.1 Abstract

A CAPTCHA is "Completely Automated Public Turing test to tell Computers and Humans Apart"[1]. A CAPTCHA is a program that generates and grades tests that are human solvable, but intends to be beyond the capabilities of current computer programs [2]. This technology is now almost a standard security mechanism for defending against undesirable or malicious Internet bot programs, such as those spreading junk emails and those grabbing thousands of free email accounts instantly. It has found widespread application on numerous commercial web sites including Google, Yahoo, and Microsoft’s MSN. In the project, text-based CAPTCHA images are separated to obtain characters. After the initial processing, features such as area of each character, moments, Euler number, location of character pixels in the image, etc. are extracted. Neural networks are trained in MATLAB using these features based on Error Back Propagation Training Algorithm to classify and recognize the characters. The salient points of the project are the use of a unique combination of features extraction techniques and an effective segmentation algorithm.

1.2 Introduction and Motivation

CAPTCHA was developed fundamentally to establish whether a website user is human or a machine. Because of the various ways in which websites are misused by hackers, CAPTCHA asks registrants/visitors to re-enter a randomly generated code which is usually provided as a distorted image thus blocking out any machines that are attempting to exploit the facility.

The main motivation for this project is trying to achieve automatic behavior of some CAPTCHA systems. In addition, some CAPTCHA design researchers also cited many papers related to CAPTCHA cracking, and proposed various guidelines on CAPTCHA design.

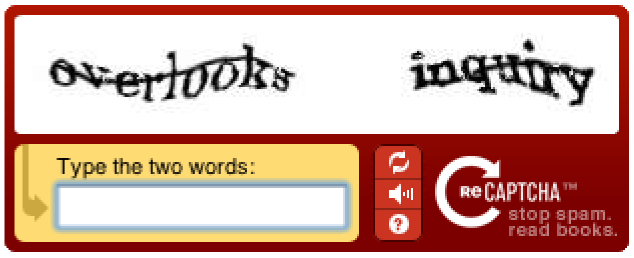


Figure 1.1 Example of text based CAPTCHA

CAPTCHAs are often used in attempts to prevent automated software from performing actions which degrade the quality of service of a given system [3], as mentioned in the first slogan of Google reCAPTCHA service: "Stop Spam". For instance, the e-mail spam can be done in a completely automated way, from e-mail account registration to spam e-mail sending. Now, many famous e-mail service providers have adopted CAPTCHA systems to prevent automatic account registration, thus prevent e-mail spam effectively. It is worth noting that many online file sharing services have also adopted CAPTCHA system, together with countdown timers, to prevent massive automatic file downloading. Each CAPTCHA implementation derives its strength by increasing the system's complexity to perform image preprocessing, segmentation, and classification.

Breaking CAPTCHA would imply uncovering security issues with the current visual CAPTCHA, which would stress for the need to develop more complicated CAPTCHA and thus enhance the strength of one of the parameters on which web security is based. The idea was chosen in order to show the concerns in today's world where the security methods developed to preserve confidentiality in online systems, of which image based CAPTCHA represent of the latest developments, are not only insecure but are prone to attacks by hackers with high success rates. Secondly, to analyze the strength of

CAPTCHA deployments on the Internet by developing the generic image preprocessing engine that can be configured as per the CAPTCHA type being analyzed.

CAPTCHA have several applications for practical security, including (but not limited to):

1. Preventing Comment Spam in Blogs: Most bloggers are familiar with programs that submit bogus comments, usually for the purpose of raising search engine ranks of some website (e.g., "buy penny stocks here"). This is called comment spam. By using a CAPTCHA, only humans can enter comments on a blog. There is no need to make users sign up before they enter a comment, and no legitimate comments are ever lost [1].

2. Protecting Website Registration: Several companies (Yahoo!, Microsoft, etc.) offer free email services. Up until a few years ago, most of these services suffered from a specific type of attack: "bots" that would sign up for thousands of email accounts every minute. The solution to this problem was to use CAPTCHAs to ensure that only humans obtain free accounts. In general, free services should be protected with a CAPTCHA in order to prevent abuse by automated scripts [1].

3. Protecting Email Addresses from Scrapers: Spammers crawl the Web in search of email addresses posted in clear text. CAPTCHAs provide an effective mechanism to hide your email address from Web scrapers. The idea is to require users to solve a CAPTCHA before showing your email address. A free and secure implementation that uses CAPTCHAs to obfuscate an email address can be found at reCAPTCHA MailHide [1].

4. Online Polls: In November 1999, *http://www.slashdot.org* released an online poll asking which was the best graduate school in computer science. As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots using programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote their own program and the poll became a contest between voting "bots." MIT finished with 21,156 votes, Carnegie

Mellon with 21,032 and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll ensures that only humans can vote [1].

5. Preventing Dictionary Attacks: CAPTCHAs can also be used to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring it to solve a CAPTCHA after a certain number of unsuccessful logins. This is better than the classic approach of locking an account after a sequence of unsuccessful logins, since doing so allows an attacker to lock accounts at will [1].

6. Search Engine Bots: It is sometimes desirable to keep webpages unindexed to prevent others from finding them easily. There is an html tag to prevent search engine bots from reading web pages. The tag, however, doesn't guarantee that bots won't read a web page; it only serves to say "no bots, please." Search engine bots, since they usually belong to large companies, respect web pages that don't want to allow them in. However, in order to truly guarantee that bots won't enter a web site, CAPTCHAs are needed [1].

7. Worms and Spam: CAPTCHAs also offer a plausible solution against email worms and spam: "I will only accept an email if I know there is a human behind the other computer." A few companies are already marketing this idea [1].

1.3 Problem Statement

This project proposes a CAPTCHA recognizer that interprets the characters from text based CAPTCHA of selected websites. This is done using various Image Processing techniques and Optical Character Recognition (OCR) algorithms of Neural Networks.

CAPTCHA images are collected from each of the websites under consideration. These images are preprocessed using different techniques of Image Processing and finally these are segmented into images such that every image is a binary image contains exactly one character. The database, thus created, contained 80 images for each character. Various feature extraction techniques were implemented, and this data was passed into artificial neural network which trained on the basis of Error Back Propagation Training Algorithm. The neural network created in this way was used to identify characters from test images.

The objective of this system is to show that image based CAPTCHAs are not as secure as their authors claim. This automatically leads to insecurity for the different applications using the image CAPTCHA. This idea has been chosen in order to show concerns in today's world where the security methods developed to preserve confidentiality in online systems, of which image based CAPTCHA represent of the latest developments, are not only insecure but are prone to attacks by hackers with high success rates.

1.4 Scope of the Project

The proposed system primarily aims at breaking a wide variety of text based CAPTCHA. Since a CAPTCHA is an image with randomly generated characters that are heavily distorted, it is difficult even for humans to always interpret it correctly. The aim is to break the CAPTCHA with the best possible accuracy within the knowledge and time limit.

The CAPTCHA under consideration are from websites of Vodafone, PayPal, Wikipedia, EZ Gimpy, Indian Passport, Gojiyo, MP3Raid, Authorize and Register and IIM Kozhikode. The types of CAPTCHA under consideration are the ones in which all the characters in the image are disjointed. The characters include 50 alphanumeric characters that include both upper case and lower case characters and digits.

1.5 Organization of Project Report

The report is separated into chapters which are organized into the following parts

Chapter 2: Review of Literature

The project lies within the domain of Image Processing, Computer Vision, Fuzzy logic and Neural Networks. Within these domains, concepts like noise removal, segmentation, feature extraction and character recognition are implicitly included. These concepts play a very significant role throughout the project wherein each technology helps to proceed towards the goal to provide an efficient implementation.

Chapter 3: <u>Analysis and Design</u>

This chapter gives a general description of the system as a whole. The functional and non-functional requirements are explained in this chapter followed by a detailed system analysis which includes theoretical considerations, assumptions and an in depth description of every module of the proposed system.

Chapter 4: <u>Implementation and Results</u>

This chapter gives details about the overall system algorithm and algorithms for individual modules in the project. Multiple algorithms, if used at any stage are also mentioned distinctly. It is followed by statistical results of each module.

Chapter 5: <u>Testing</u>

This chapter gives the testing results on every component individually and also testing on the entire system. The system is tested for CAPTCHA of 10 websites and the obtained accuracy is stated.

Chapter 6: <u>Conclusion and Further Work</u>

This chapter concludes the report by giving the summary of the entire project and discussing about the areas of development and research in future.

2. REVIEW OF LITERATURE

This chapter describes the current technologies and methodologies for breaking CAPTCHA and the technologies and methodologies used by DECAPTCHER-A CAPTCHA Solver and Analysis Tool.

2.1 CAPTCHA

A CAPTCHA is a type of challenge-response test used in computing as an attempt to ensure that the response is generated by a person. The process usually involves a computer asking a user to complete a simple test which the computer is able to grade. These tests are designed to be easy for a computer to generate, but difficult for a computer to solve, so that if a correct solution is received, it can be presumed to have been entered by a human. A common type of CAPTCHA requires the user to type letters or digits from a distorted image that appears on the screen, and such tests are commonly used to prevent unwanted internet bots from accessing websites [3].

A CAPTCHA is sometimes described as a reverse Turing test, because it is administered by a machine and targeted at a human, in contrast to the standard Turing test that is typically administered by a human and targeted at a machine.

For the project, CAPTCHA from 10 different websites were considered. These varied in their background color, noise, font style, orientation, foreground color, image size, image file format, etc. The types of CAPTCHA considered for the project report are as discussed below. The properties of each type of CAPTCHA have been discussed below.

2.1.1 EZ Gimpy

As seen in figure 2.1, this CAPTCHA has skewed characters. The font color of the characters is generally black. Completely random noise exists in this type of CAPTCHA. Grid lines exist in a few variations. It is one of the oldest CAPTCHA, and is rarely used these days.

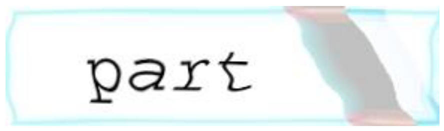


Figure 2.1 EZ Gimpy CAPTCHA

2.1.2 Register

This CAPTCHA has a multicolored background without a definite crisp boundary. Figure 2.2 shows that the characters are non- linearly arranged. Also, the thickness of the characters is not uniform.

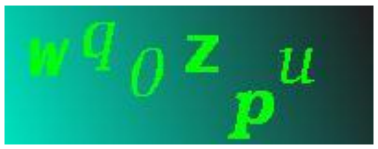


Figure 2.2 Register CAPTCHA

2.1.3 PayPal

This CAPTCHA has a fixed background noise as seen in figure 2.3. The characters “PayPal” are used as noise, and exist in large number in the CAPTCHA. Characters and noise are of same color; but the color is not fixed (either blue or white, the other being used for the background). Characters may be slightly rotated.

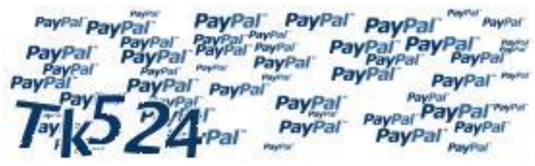


Figure 2.3 PayPal CAPTCHA

2.1.4 MP3Raid

This type of CAPTCHA has large amount of stray lines of a different intensity than the character text as seen in figure 2.4. Therefore, correct thresholding of the image can remove the noisy lines. The CAPTCHA always consists of 6 characters and are always vertical.

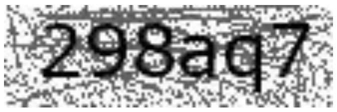


Figure 2.4 MP3Raid CAPTCHA

2.1.5 Passport

This class of CAPTCHA is the only one in which grid lines exist. As seen in figure 2.5, a peculiar characteristic is that the first character is always an alphabet and it is followed by 5 numbers. Grid lines are black in color, whereas characters are blue. A design consideration is that removal of grid lines may result in periodic breaks in the characters.

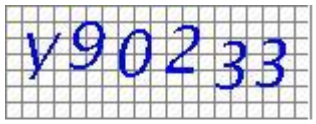


Figure 2.5 Passport CAPTCHA

2.1.6 Wikipedia

Characters are highly distorted in this type of CAPTCHA. Thickness is non-uniform. No noise exists but pre-processing may induce some noise. This CAPTCHA has maximum number of characters ranging from 5 to 12.Figure 2.6 shows a Wikipedia CAPTCHA.

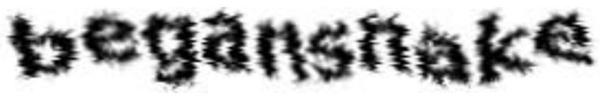


Figure 2.6 Wikipedia CAPTCHA

2.1.7 Vodafone

From figure 2.7, it can be easily inferred that this is the simplest of CAPTCHA considered. It has red colored characters and gray background. No noise exists. Characters do not have any variations in different CAPTCHAs i.e. an ‘a’ in one Vodafone CAPTCHA is an exact copy of any other ‘a’ encountered for the same.

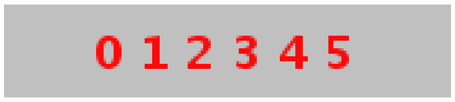


Figure 2.7 Vodafone CAPTCHA

2.1.8 GoJiyo

This CAPTCHA contains non-linearly arranged characters without rotation. There exists slight variation in the font size of the characters. Background consists of thin strips of red color. Figure 2.8 shows a GoJiyo CAPTCHA.

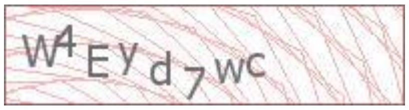


Figure 2.8 GoJiyo CAPTCHA

2.1.9 IIM Kozhikode

From the figure 2.9,it is seen that IIM Kozhikode CAPTCHA has no noise and always has black background and white text with only 4 characters. The characters are rotated at huge angles.

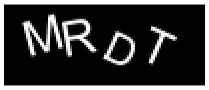


Figure 2.9 IIM Kozhikode CAPTCHA

2.1.10 Authorize

The Authorize CAPTCHA, as seen in figure 2.10, has salt and pepper noise. Characters may have standard skew in either left or right direction. Image dimensions are very large in case of this CAPTCHA.

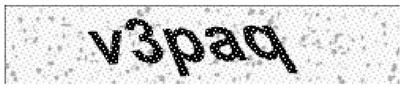


Figure 2.10 Authorize CAPTCHA

2.2 Current Methodology and Technology

Currently, there are three methods to defeat text based CAPTCHA. These are widely used and have yielded good accuracy, if properly implemented. Among these methods, the Template Matching technique is applicable only for a single class of CAPTCHA, i.e. a separate template is required for every new font of CAPTCHA. The technologies used are as follows

2.2.1 Insecure Implementation

Like any security system, design flaws in a system implementation can prevent the theoretical security from being realized. Many CAPTCHA implementations, especially those which have not been designed and reviewed by experts in the fields of security, are prone to common attacks. Some CAPTCHA protection systems can be bypassed without using OCR simply by re-using the session ID of a known CAPTCHA image. A correctly designed CAPTCHA does not allow multiple solution attempts at one CAPTCHA. This prevents the reuse of a correct CAPTCHA solution or making a second guess after an incorrect OCR attempt. Other CAPTCHA implementations use a hash (such as an MD5 hash) of the solution as a key passed to the client to validate the CAPTCHA. Often the CAPTCHA is of small enough size that this hash could be cracked. Further, the hash could assist an OCR based attempt. A more secure scheme would use an HMAC. Finally, some implementations use only a small fixed pool of CAPTCHA images.

Eventually, when enough CAPTCHA image solutions have been collected by an attacker over a period of time, the CAPTCHA can be broken by simply looking up solutions in a table, based on a hash of the challenge image [3].

2.2.2 Template Matching

A number of research projects have attempted to beat visual CAPTCHAs by creating programs that contain the following functionality:

1. Pre-processing: Removal of background clutter and noise.
2. Segmentation: Splitting the image into regions which each contain a single character.
3. Classification: Identifying the character in each region using template matching.

Among the above steps, the only step where humans still outperform computers is segmentation. If the background clutter consists of shapes similar to letter shapes, and the letters are connected by this clutter, the segmentation becomes nearly impossible with current software. Hence, an effective CAPTCHA should focus on the segmentation[3].

Template matching is one of the most common classification methods. Here, every image pixel is used as a feature. Classification is performed by comparing an input character with a set of templates (or prototypes) from each character class. Each comparison results in a similarity measure between the input characters with a set of templates. One measure increases the amount of similarity when a pixel in the observed character is identical to the same pixel in the template image. If the pixels differ the measure of similarity may be decreased. After all templates have been compared with the observed character image, the character’s identity is assigned the identity of the most similar template. Template matching is a trainable process as template characters can be changed. All attempts to break CAPTCHA have been concentrated only on a class of CAPTCHA. As a result, even if slight variations are seen, the accuracy rate of these systems is reduces drastically. No implementation has been able to break all types of CAPTCHA until now [10].

2.2.3 Human Solvers

CAPTCHA is vulnerable to a relay attack that uses humans to solve the puzzles. One approach involves relaying the puzzles to a group of human operators who can solve CAPTCHA. In this scheme, a computer fills out a form and when it reaches a CAPTCHA, it gives the CAPTCHA to the human operator to solve [3].

Spammers pay about $0.80 to $1.20 for each 1,000 solved CAPTCHA to companies employing human solvers in Bangladesh, China, India, and many other developing nations. Others cite a price tag of as low as $0.50 for each 1,000 solved.

Another approach involves copying the CAPTCHA images and using them as CAPTCHA for a high-traffic site owned by the attacker. With enough traffic, the attacker can get a solution to the CAPTCHA puzzle in time to relay it back to the target site [3].

2.3 Methodology and Technology used – Character recognition using Artificial Neural Networks

The methodology followed for breaking CAPTCHA includes character recognition use of artificial neural networks and feature extraction methods. Neural network such as feed forward Backpropagation neural network requires long training time just to "memorize" all possible input vectors that are fed into the network. However, there is always a possibility that the network will give false results due to poor generalization capability. The recognition accuracy of the handwritten characters depends a lot on the exemplars that are used for recognition. In general, the overall recognition process can be divided into 3 main sections, namely preprocessing, segmentation and classification [5]. Preprocessing requires noise removal and other processing such as converting image to binary, etc. before the images are fed to the segmentation unit. Segmentation is isolating the characters individually before they are passed to the feature extraction unit where the important features of characters are identified. Finally, classification process is done by determining the category or the group of each character used during the recognition process.

Initially, 125 images for each of the 10 CAPTCHA classes were obtained. Different schemes in Image Processing were applied depending upon the type of CAPTCHA. These preprocessed images were then segmented into individual characters. Now, the entire database was segregated into separate characters. Folders were created and each folder contained images of exactly one character. These images were refined to 80 images for each of the 50 characters under consideration. Folders of images were separated on the basis of their Euler number- pre classification. Finally, sets of these images were taken for training in the neural networks using the Error Back Propagation Training Algorithm.

2.3.1 Optical Character Recognition

Optical character recognition, usually abbreviated to OCR is a conversion of scanned images of handwritten, typewritten or printed text into machine-encoded text. A OCR system is usually implemented either by using simple patter matching or by using self-learning intelligent systems.

Some of the self-learning systems are Self-Organizing Map (SOM), Support Vector Machine (SVM) and supervised Artificial Neural Networks (ANN), etc. The most common system used are supervised ANN while others are rarely used but still give good results for some of the applications.

Self-Organizing Map: A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space [11].

Support Vector Machine: A support vector machine (SVM) is a concept in statistics and computer science for a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the input, making the SVM a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on [12].

2.3.2 Artificial Neural Networks

The term neural network was traditionally used to refer to a network or circuit of biological neurons. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, are used to extract patterns and detect trends that are too complex to be noticed by either humans or other naive computer techniques. Neural Networks are used for learning wherein the computer can decide efficiently what can be the output for a specific set of input. The true power and advantage of neural networks lies in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modeled. It modifies itself for every training data to correctly classify the input set. Neural Networks, combined with Fuzzy Logic is considered to be the best bet for such types of problems where the input is highly variable and the weights have to be constantly updated accordingly [13].

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in 2 ways:

1. Knowledge is acquired by the network through a learning process.

2. Interconnection strengths known as synaptic weights are used to store the knowledge.

Basically, learning is a process by which the free parameters (i.e., synaptic weights and bias levels) of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place. In a general sense, the learning process may be classified as supervised and unsupervised learning.

Supervised learning has been implemented in the project. In this type of learning, a desired output is provided to the network during the training of the network. Given the training sample, the requirement is to compute the free parameters of the neural network so that the actual output of the neural network due to input vector is close enough to desired output in a statistical sense. For example, we may use the mean-square error as the index of performance to be minimized.

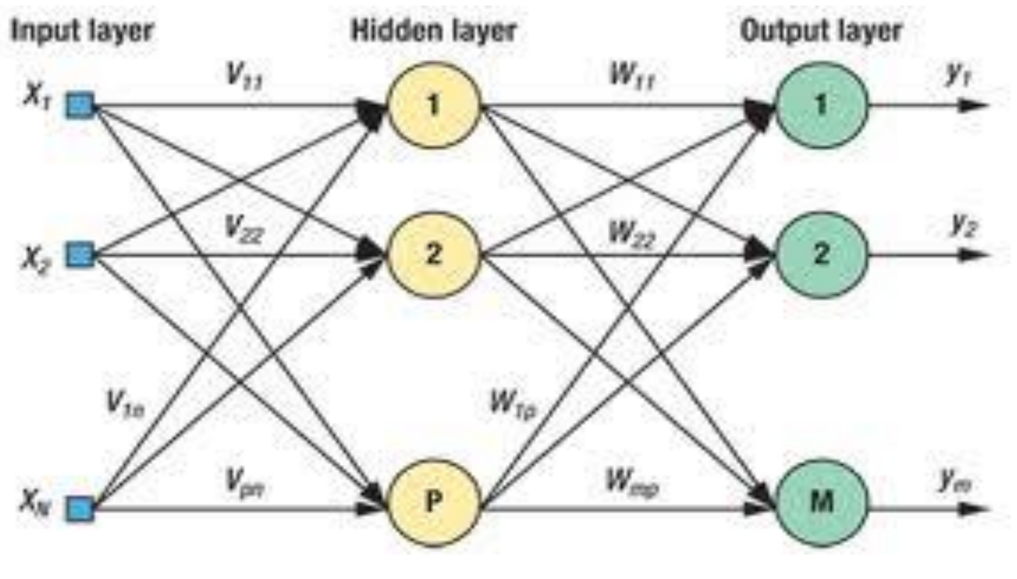


Figure 2.11 Multilayered Feed forward Network

The back-propagation algorithm has emerged as the workhorse for the design of a special class of layered feed forward networks known as multilayer perceptrons (MLP). A multilayer perceptron has an input layer of source nodes and an output layer of neurons (i.e., computation nodes); these two layers connect the network to the outside world. In addition to these two layers, the multilayer perceptron usually has one or more layers of hidden neurons, which are so called because these neurons are not directly accessible. The hidden neurons extract important features contained in the input data.

The back-propagation algorithm has emerged as the workhorse for the design of a special class of layered feed forward networks known as *multilayer perceptrons* (MLP). As shown in Fig. 2.11, a multilayer perceptron has an input layer of source nodes and an output layer of neurons (i.e., computation nodes); these two layers connect the network to the outside world. In addition to these two layers, the multilayer perceptron usually has one or more layers of hidden neurons, which are so called because these neurons are not directly accessible. The hidden neurons extract important features contained in the input data.

2.4. Hardware and Software Requirements

The hardware and software requirements for the DECAPTCHER-A CAPTCHA Solver and Analysis Tool System are discussed in this section.

2.4.1 Hardware Requirements

The minimum system hardware requirements for MATLAB are

1. Processors: Intel Pentium IV (minimum).

2. Memory: 1 GB (minimum).

2.4.2 Software Requirements

MATLAB 7.12(R2011a) or higher (Neural Network Toolbox)

2.4.3 Operating System Requirements

Windows XP, Windows Vista, Windows 7

3. ANALYSIS AND DESIGN

This chapter gives a general description of the system as a whole. The functional and non-functional requirements are explained in this chapter followed by a detailed system analysis which includes theoretical considerations, assumptions and an in depth description of every module of the proposed system.

3.1 Requirement Analysis

This chapter gives details about the functional and non-functional requirements of the system.

3.1.1 Functional Requirements

The primary function that the system provides is to correctly decode the text from the CAPTCHA image. It should provide a great accuracy in correctly identifying the characters in the image. The Use Case diagram is as follows.

The system consists of the following major functionalities

1. The user provides the path of the saved image file. Alternatively, he/she may directly provide the URL of the image.

2. This will result in the system trying to solve the CAPTCHA using the default parameters.

3. An advanced user may provide parameters explicitly. Parameters include threshold value to convert image to binary, removal of grid noise, etc. and all these can be altered only for pre-processing functions.

4. Once the pre-processing is completed, the system segments the CAPTCHA image by detecting the connected components, and eliminating the unnecessary ones.

5. Individual segmented images are passed to the system, which tests it with the neural network and gives a single character as an output for every image.

6. The outputs are combined to provide the solution to the input image.

3.1.2 Non- Functional Requirements

Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation. The text CAPTCHA recognition consists of three stages. These stages are: preprocessing, segmentation and recognition. These three stages are interconnected. The segmentation efficiency depends on the efficiency of the preprocessing. And the efficiency of the whole system depends on the efficiency of the segmentation and the recognizer. This can be shown by the following:

SYS = Seg*Rec^N

where SYS is a relative frequency of the correctly recognized text CAPTCHA, Seg is the relative frequency of the correctly segmented text-CAPTCHA, Rec is the relative frequency of a correctly recognized symbol, N is a number of symbols on the text CAPTCHA. The term "relative frequency" is used when dealing with limited size sets without knowing the probability distribution function. The relative frequency insignificantly differs from the probability due to the Bernully theorem which says that with the large size of set the relative frequency is approximately equal to the probability. Some of the non-functional requirements of the system can be broadly classified according to their states i.e. run time and design time [14].

Execution qualities (which are observable at run time):

1. Accessibility: The system should be easily accessible to the end user and should not produce any type of complexity during the use.

2. Dependability: Since the solving of the CAPTCHA completely depends on the system, a high dependability factor should be present that could develop a factor of trust between the user and the system.

3. Accuracy: The system should be as accurate as possible to always provide correct results to all challenges put forward.

4. Throughput: The time to solve each CAPTCHA should be as low as possible and hence the aim to keep the average throughput of maximum ten seconds per CAPTCHA.

Evolution qualities (which are embodied in the static structure of the software system):

1. Performance: The performance of the system highly depends upon both the accuracy and the average time. It should be reliable enough to provide high performance.

2. Maintainability and scalability: It should also not possess high maintainability issues wherein the whole system has to be rebuilt for a minimal amount of change. It should be built in such a way that scalability should not be a problem when new types of CAPTCHA are encountered in future.

3. Legality: The software should always be used for legal purposes and should never violate the terms and services of any website taken into consideration.

3.2 Design Considerations

While designing the system, all the previous developments made in the domain, and the potential researches that can be effectively considered, have been taken into account. Since breaking a CAPTCHA is considered to be an HARD AI problem mainly because of its segmentation stage. The only challenging task that separates a usual OCR algorithm to an algorithm that breaks a CAPTCHA image is the segmentation of the individual characters. Hence an optimum system should more focus on the most important challenge of segmentation while designing a CAPTCHA breaker.

The major challenges that exist in the system are:

1. Efficiently segmenting the CAPTCHA where characters are highly distorted and connected. Hence, the algorithm should efficiently be able to separate the characters for correct recognition.

2. Taking intelligent decisions while recognizing characters that resemble all most the same like when recognizing characters like “b” and “6”, “0” and “o”, “i”, ”l” and “1” and so on.

3. Maintaining a decent accuracy rate of breaking all types of CAPTCHA while balancing it with the amount of time taken to recognize the CAPTCHA.

4. Finally the task of bringing it all together at one place and integrating the system with the web to actually bring it into live use of solving the CAPTCHA while using the web.

3.2.1 Assumptions

While implementing the system following two assumptions are considered

1. A total set of 50 alphanumeric characters are used. This consists of the ten numerals from 0 to 9, and others are alphabetic characters. Among the remaining 40 characters, for cases in which the small case and upper case alphabets are identical (for e.g. s and S), only 1 character is considered. There are 12 such cases. This allows in reducing the classification complexity by about 20%

2. In order to recognize the characters correctly it is considered that the characters are segmented and not joint. Only CAPTCHA with disjoint characters are considered.

3.3 Project Design

The system block diagram, shown in figure 3.1, shows different modules of the project. Each module is further divided into various sub-modules.

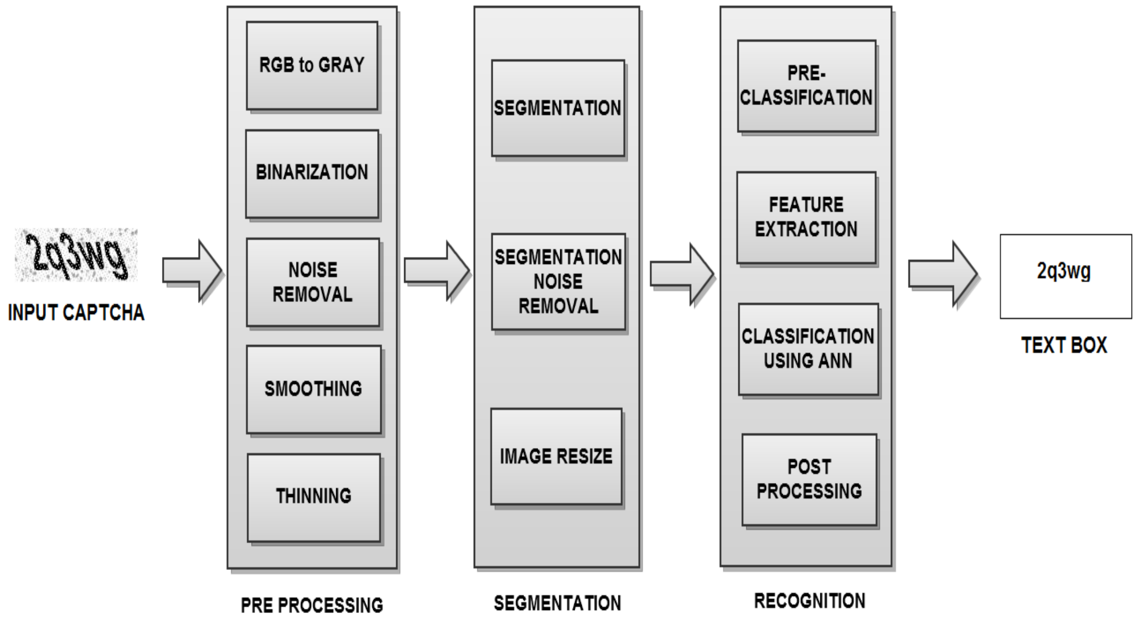


Figure 3.1 System Block Diagram

The system will perform the following tasks.

3.3.1 Accepting the input

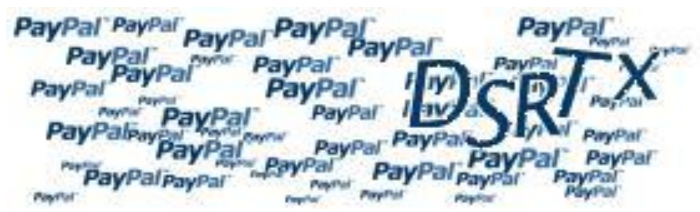


Figure 3.2 Input image

The original CAPTCHA image obtained from the web is generally in RGB format and contains background noise as shown in figure 3.2. Hence to make the remaining process simple it is necessary to preprocess the image efficiently. This is obtained from the GUI either by passing the URL or file location on computer memory.

3.3.2 Pre-processing the CAPTCHA image

The input CAPTCHA image will be first preprocessed i.e. converted into gray scale followed by thresholding, noise removal and optional smoothening. The characters in the binary image will then be further processed for thinning and segmentation to separate out the individual characters. These steps have been discussed below.

3.3.2.1 RGB to Gray

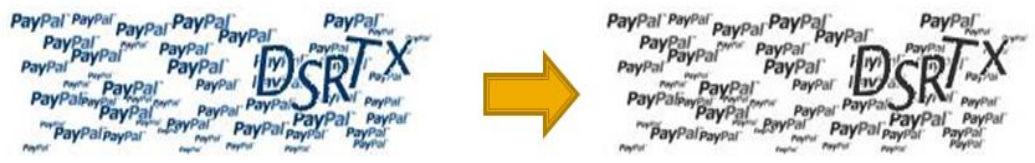


Figure 3.3 RGB to Grayscale

Figure 3.3 shows the conversion from RGB to Grayscale of PayPal CAPTCHA. First, the original RGB CAPTCHA image is converted into gray scale because to work on each color in the image is very difficult. So, converting it into gray scale helps to work on 256 intensity values. Scan the entire image pixel by pixel.

3.3.2.2 Binarization

After the grayscale image is obtained, the image is binarized; i.e. the image is retained with only two possible intensity values – Black(0) and White(1). Figure 3.4 shows the conversion of a Grayscale image to a Binary image. Binary images are often produced by thresholding a grayscale or color image, in order to separate an object in the image from the background.

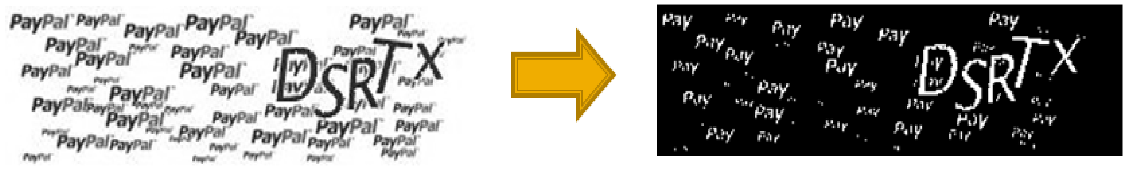


Figure 3.4 Grayscale to Binary

The color of the object is referred to as the foreground color. The rest is referred to as the background color. If intensity of current pixel <threshold value, then set intensity of current pixel to 0 (Black) else set it to 1 (White).Here, the threshold values differ for different classes of CAPTCHA.

3.3.2.3 Noise Removal

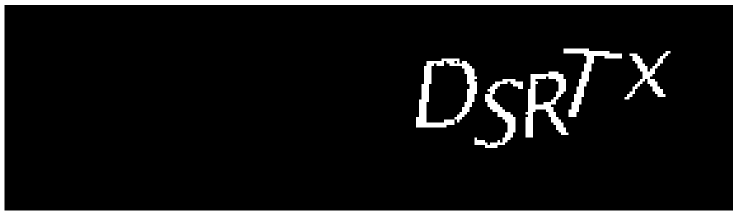


Figure 3.5 Noise Removal

After binarization, a CAPTCHA image may contain unnecessary set of white pixels i.e. noise. Hence noise is removed using various low pass filters like median filter as shown in figure 3.5.

3.3.2.4 Smoothing

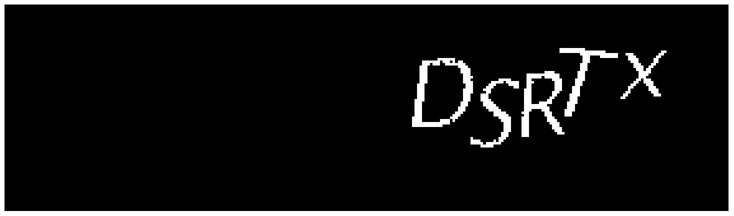


Figure 3.6 Smoothing

Smoothing may be classified as a part of Noise removal module in a normal OCR problem. However, in case of CAPTCHA images, text may be highly distorted, as in the case of Wikipedia CAPTCHA. The effect of smoothing is as seen in figure 3.6.

3.3.2.5 Thinning

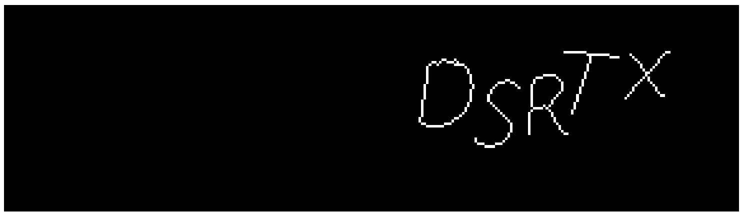


Figure 3.7 Thinning

Thinning provides a tremendous reduction in data size, thinning extracts the shape information of the characters. It can be considered as conversion of off-line handwriting to almost on-line data. From the above figure 3.7, it can be easily seen that thinning is the process of reducing thickness of each line of pattern to just a single pixel. In this project, morphology based thinning algorithm has been used for better symbol representation.

3.3.3 Segmentation

After the image passes the preprocessing stage, character edges are first thinned before performing segmentation algorithms. In some CAPTCHA classes, all the characters are distant so it is not very difficult to segment them. But in some other cases, alternate algorithms have to be implemented to separate the characters in order to apply OCR techniques on it efficiently. These classes of CAPTCHA are, however, out of the scope of project. The segments of original image are as shown in figure 3.8.

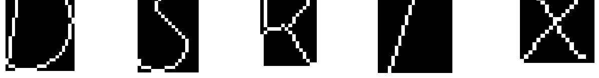


Figure 3.8 Segmentation

The boundaries of segments are extracted using region properties. Using these boundary points, individual characters are extracted from the entire image. Such images are of varied sizes. They are resized into various sizes and passed into the next module of feature extraction.

3.3.3.1 Segmentation Noise Removal

Using flood-fill algorithm, determine all the connected components in the image. This includes characters as well as noise. Compute the area of each connected component. Set a threshold value (t_v) and remove all those components which have area $<t_v$. This threshold value differs for different classes of CAPTCHA, because the size of CAPTCHA image is not constant and hence also the size of individual characters. The segmentation noise removal process is shown in figure 3.9.

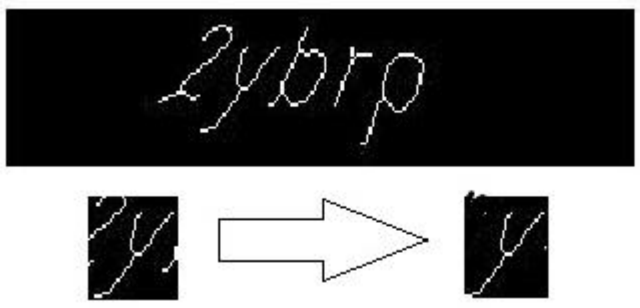


Figure 3.9 Segmentation Noise Removal

3.3.3.2 Image Resize

Image resize or normalization is required as the size of the numeral varies from character to character. Also, some feature extraction techniques implemented in the project require a fixed image size. As a result, the input image is normalized to size 16 X 16 pixels after finding the bounding box of each segment of the original image.

3.3.4 Recognition

The segmented characters can now be used for recognition. The system breaks CAPTCHA by recognizing the text character-by-character. The recognition is done with the help of an Artificial Neural Network. Before the input is fed to the neural network, the input is pre-classified to decide which network to forward it to.

3.3.4.1 Pre-classification

Pre-classification is done based on a property known as Euler number. The Euler number of a character is the difference between the number of connected areas and the number of holes in the character. There are 16 characters with euler number 0, 32 characters with euler number 1 and 2 characters with euler number -1. Hence accordingly the incoming character is preclassified.

3.3.4.2 Feature Extraction

Relevant features will be extracted from the characters which forms an input vector to a trained Artificial Neural Network which will classify the individual images to their corresponding characters.

The most important aspect of handwriting recognition scheme is the selection of good feature set, which is reasonably invariant with respect to shape variations caused by various styles.

1. Zone Based Feature Extraction (Centroid Method)

The major advantage of this approach stems from its robustness to small variation, ease of implementation and provides good recognition rate. This method provides good result even when certain preprocessing steps are not considered.

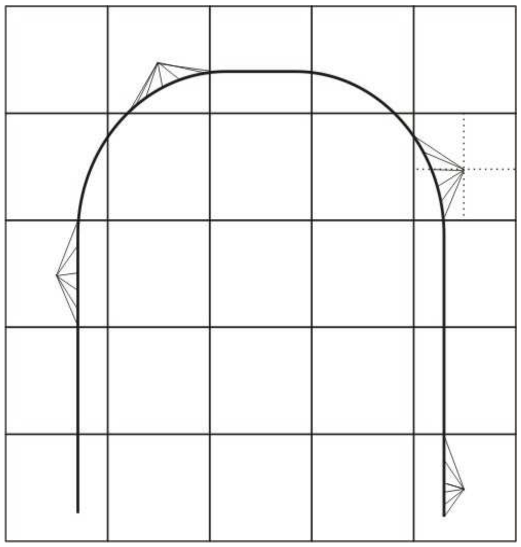


Figure 3.10 ZCZ Feature Extraction

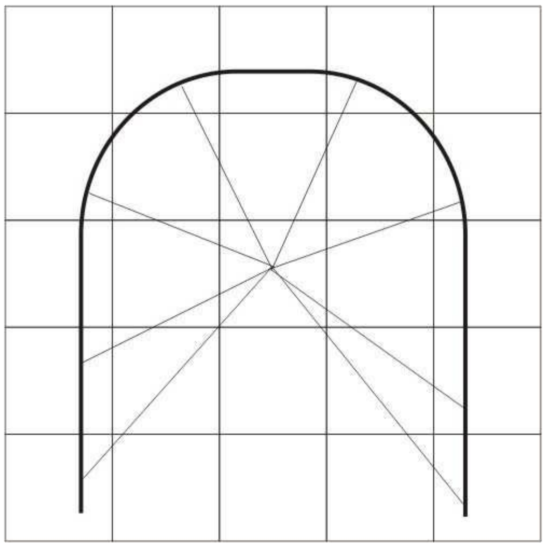


Figure 3.11 ICZ Feature Extraction

The character centroid is computed and the image is further divided into sixteen equal parts or zones. Average distance from the character centroid to the each pixel present in the zone is to be computed. Similarly, zone centroid is computed and average distance from the zone centroid to each pixel present in the zone is to be computed. This procedure is repeated for all the zones present in the numeral image. There could be some zones that are empty, and then the value of that particular zone image value in the feature vector is zero. Finally 32 such features are used for feature extraction. For classification and recognition nearest neighbor classifier and feed forward back propagation neural network classifiers are used. Thus, a hybrid feature extraction system wherein Image centroid and Zone (ICZ) based distance metric feature extraction system [7], shown in figure 3.11, and Zone Centroid and Zone (ZCZ) based Distance metric feature extraction system [8], shown in figure 3.10, are implemented.

2. Zone Based Feature Extraction (Contour Method)

In this method the geometric features of the character contour are extracted. These features are based on the basic line types that form the character skeleton. It extracts different line types that form a particular character. It also concentrates on the positional features of the same. The universe of discourse is formed which is nothing but the shortest matrix that fits the entire character skeleton. The Universe of discourse is selected because the features extracted from the character image include the positions of different line segments in the character image. So every character image should be independent of its Image size. After the universe of discourse is selected, the image is divided into windows of equal size, and the feature is done on individual windows. The image was zoned into 9 equal sized windows. Feature extraction was applied to individual zones rather than the whole image. This gives more information about fine details of character skeleton. Also positions of different line segments in a character skeleton become a feature if zoning is used. This is because, a particular line segment of a character occurs in a particular zone in almost cases [15]. For instance, the horizontal line segment in character 'A' almost occurs in the central zone of the entire character zone.

To extract different line segments in a particular zone, the entire skeleton in that zone should be traversed. After line segments have been extracted from the image, they have to be classified into any one of the following line types (i) Horizontal line, (ii) Vertical line, (iii) Left diagonal line, (iv) Right diagonal line.

For this, a direction vector is extracted from each line segment which will help in determining each line type. After the line type of each segment is determined, feature vector is formed based on this information [15]. Every zone has a feature vector corresponding to it. Under the algorithm proposed, every zone has a feature vector with a length of 9. The contents of each zone feature vector are as mentioned ahead.

1. Number of horizontal lines.

2. Number of vertical lines.

3. Number of Right diagonal lines.

4. Number of Left diagonal lines.

5. Normalized Length of all horizontal lines.
6. Normalized Length of all vertical lines.
7. Normalized Length of all right diagonal lines.
8. Normalized Length of all left diagonal lines.
9. Normalized Area of the Skeleton.

3. Feature Extraction using Region properties

Various properties of the region with respect to the characters were extracted and from them 8 properties were selected. Following are the image properties selected in feature vector.

Area: This gives the actual number of pixels in the region [16].

Perimeter: Perimeter is calculated as the distance between each adjoining pair of pixels around the border of the region [16].

Minor Axis Length: The length (in pixels) of the minor axis of the ellipse that has the same normalized second central moments as the region. This property is supported only for 2-D input label matrices [16].

Major Axis Length: The length (in pixels) of the major axis of the ellipse that has the same normalized second central moments as the region. This property is supported only for 2-D input label matrices [16].

Eccentricity: It specifies the eccentricity of the ellipse that has the same second-moments as the region. The eccentricity is the ratio of the distance between the foci of the ellipse and its major axis length. The value is between 0 and 1. (0 and 1 are degenerate cases; an ellipse whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment.) This property is supported only for 2-D input label matrices [16].

Equivalent Diameter: It specifies the diameter of a circle with the same area as the region. Computed as sqrt(4*Area/pi). This property is supported only for 2-D input label matrices [16].

Orientation: Orientation is the angle between the x axis and the major axis of the ellipse that has same secondary moments as that of region. Orientation value is the angle between the horizontal dotted line and the major axis [16].

Solidity: It is the scalar specifying the proportion of the pixels in the convex hull that are also in the region. It is computed as Area/Convex Area. This property is supported only for 2-D input label matrices [16].

Reflection X coefficient and Reflection Y coefficient: Some of the characters like 0, M, N are symmetric about X Y axes. This information can be used to distinguish different characters especially 0 and D. This symmetry information can be quantified by a coefficient called as reflection symmetry coefficient [17].

4. Feature Extraction using Horizontal and Vertical Profiles

It includes projection information about X and Y axes. Projection is mapping of binary image into one dimensional array. A horizontal projection is sum of white pixels (background being black) projected onto vertical axis x, while vertical projection is the sum of white pixels onto horizontal axis y. Projection data forms major part of feature vector and is unique for each character.

3.3.4.3 Classification using Artificial Neural Network

The features extracted from all the images are used to create a Neural Network which is trained using Error Back Propagation Training Algorithm. The number of input nodes, the number of neurons in the hidden layer, the number of neurons in the output layer, the goal, the number of epochs, the learning rate and the momentum constant and other such parameters of the neural network are decided and the neural network is created.

EBPTA is a supervised training algorithm. As a result, the desired output is provided along with the feature vector during the training phase . The network is trained until either of the following conditions is satisfied

1. The maximum number of epochs (repetitions) is reached.

2. The maximum amount of time is exceeded.

3. Performance is minimized to the goal.

4. The performance gradient falls below min_grad [18].

Recall

The training set feature vector is again provided to the trained EBPTA neural network. The aim of performing Recall is to ensure that the network does not give any error in classifying the characters which it has already seen. If all the characters in the training set are classified correctly, then the training has completed successfully.

Generalization

New set of character database is provided to the trained EBPTA neural network. Based on the set weights it tries to classify each character to the nearest possible match. The accuracy of the whole CAPTCHA majorly depends on the classification accuracy of individual characters.

3.3.4.4 Post Processing

Once results from the recognizer are obtained, a post processing step is done on the output to verify the results. Post processing is required because the accuracy of the classifier is limited due to the fact that the features used to classify the characters are not providing enough information for the classification. But once a classification has been made by the recognizer, the testing results can be used to check if there is an error or not.

Since the type of CAPTCHA is known, properties of the CAPTCHA can be used to check for any possible errors. For example, Passport CAPTCHA have an alphabet followed by 5 numerals, CAPTCHA from Authorize have exactly 5 characters, and so on. Using this information, some misclassifications can be corrected and the accuracy of the system can be improved. The output of the post processing stage is sent to GUI as output of the recognizer.

4. IMPLEMENTATION AND RESULTS

This chapter gives details about the algorithms for individual modules in the project. Multiple algorithms, if used at any stage are also mentioned distinctly. It also presents the implementation issues and results of each module.

4.1 Implementation Details

In this section, the overall system algorithm and the algorithms used in each of the module of the system have been discussed.

4.1.1 Algorithm for DECAPTCHER

Input: CAPTCHA Image

Output: String of recognized characters

Algorithm:

Step 1: Read the path of the CAPTCHA image.

Step 2: Determine the type of CAPTCHA from the source path and read the image.

Step 3: Send the CAPTCHA image and its type to the preprocessing engine and get back the preprocessed image.

Step 4: The preprocessed image is then segmented to get back an array of segmented images.

Step 5: For each character, the Euler number (en) of the character image is determined.

Step 6: IF en==0 then the character is simulated on the network net0 which is trained for characters of Euler number 0.

ELSE IF en==1 then the character is simulated on the network net1 which is trained for characters of Euler number en 1.

ELSE IF en==-1 then the character is simulated on the network net_1 which is trained for characters of Euler number en -1.

Step 7: The result of each simulation is concatenated to get a string of recognized text.

4.1.2 Algorithm for Pre-processing

Step 1: Convert a color image from RGB scale to grayscale.

Input: RGB image

Output: Grayscale image

Algorithm:

1. Check if the image is RGB or not. If no, then return else move to step 2.

2. Pass the image to the rgb2gray() method that computes the grayscale values by forming a weighted sum of the *R*, *G*, and *B* components:

$$0.2989 * R + 0.5870 * G + 0.1140 * B \tag{4.1}$$

Step 2: Converting grayscale image to Binary image

Input: Grayscale image

Output:Binary Image

Algorithm:

1. Scan image pixel by pixel

2. If intensity of current pixel < 127 set intensity of current pixel to 0 (Black) else set it to 255 (White).

The threshold level considered varies for different classes of CAPTCHA because of variation in font color and background colors.

Step 3: Remove salt and pepper noise.

Input: Binary image with noise

Algorithm:

1. Select a seed pixel, p.

2. Use flood-fill algorithm to label all the pixels in the component containing p.

3. Search for the next unlabeled pixel.

4. Repeat steps 2 and 3 until all the pixels are labeled.

5. Compute the area of each connected component.

6. Set a threshold value (t_v) to remove all those components which have area<$t_{v.}$

The bwareaopen (BW, P) removes from a binary imageBW, all connected components (objects) that have fewer than P pixels, producing another binary image. The default connectivity is 8 for a two dimensional image [19].

Step 4:Thin the binary image.

Input: Binary image with thick lined characters

Output: Binary image with thin lined characters

Algorithm:

1. Divide the image into two distinct subfields in a checkerboard pattern.

2. In the first subiteration, delete pixel *p* from the first subfield if and only if the conditions G_1, G_2, and G_3 are all satisfied.

3. In the second subiteration, delete pixel *p* from the second subfield if and only if the conditions G_1, G_2, and G_3' are all satisfied.

The pixels x_1, x_2, ..., x_8 are the values of the eight neighbors of *p* and are collectively denoted as *N*(*p*), starting with the east neighbor and numbered in counter-clockwise order. The number of white pixels in *N*(*p*) is denoted by *w(p)*. Hilditch [22] defined the crossing number $X_H(p)$ as the number of times from a black point to a white point in *N*(*p*).

Condition G1:

$$X_H(p) = 1 \tag{4.2}$$

where

$$X_H(p) = \sum_{i=1}^{4} w_i \tag{4.3}$$

$$w_i = \begin{cases} 1\,, if\ x_{2i-1} = 0 \text{ and } x_{2i} = 1\ or\ x_{2i+1} = 1 \\ 0, otherwise \end{cases} \tag{4.4}$$

Condition G2:

$$2 \leq min\{n1(p), n2(p)\} \leq 3 \tag{4.5}$$

where

$$n1(p) = \sum_{k=1}^{4} x_{2k-1} \vee x_{2k} \tag{4.6}$$

$$n2(p) = \sum_{k=1}^{4} x_{2k} \vee x_{2k+1} \tag{4.7}$$

Condition G3:

$$(x_2 \vee x_3 \vee \bar{x}_8) \wedge x_1 = 0 \tag{4.8}$$

Condition G3':

$$(x_6 \vee x_7 \vee \bar{x}_4) \wedge x_5 = 0 \tag{4.9}$$

4. The two sub iterations together make up a single iteration of the thinning algorithm. When the user specifies an infinite number of iterations (n=Inf), the iterations are repeated until the image stops changing [19].

4.1.3 Algorithm for Segmentation

Input: Preprocessed Binary Image

Output: Array of segmented images.

Algorithm:

Step 1: Determine the connected components in the preprocessed image using the method bwconncomp(BW).

Step 2: Uniquely Label each component using the method bwlabel(BW).

Step 3: Determine the top left and the bottom right coordinates of each component using the Bounding Box property.

Step 4: Extract the box formed from the coordinates.

Step 5: Resize each segmented character using imresize(img,[M N]).

4.1.4 Feature Extraction Algorithms

This section explains input and output various algorithm used for Feature Extraction

1. Algorithm for Zone Based Feature Extraction (Centroid Method)

Input: Each segmented character image

Output: Feature Vector of size 32

Algorithm:

Step 1: Compute the input image centroid

Step 2: Divide the input image into n equal zones.

Step 3: Compute the distance from the image centroid to each pixel present in the zone.

Step 4: Repeat step 3 for the entire pixel present in the zone.

Step 5: Compute average distance between these points.

Step 6: Compute the zone centroid.

Step 7: Compute the distance between the zone centroid to each pixel present in the zone.

Step 8: Repeat step 7 for the entire pixel present in the zone

Step 9: Compute average distance between these points.

Step 10: Repeat the steps 3-9 sequentially for the entire zone.

Step 11: Finally, 2*n such features will be obtained for classification and recognition.

2. Algorithm for Zone Based Feature Extraction (Contour Method)

Input: Each segmented character image

Output: Feature Vector of size 81

Algorithm:

Step 1: Find out the universe of discourse so that the character image is size independent.

Step 2: Check for number of rows and columns of the image and ensure it is multiple of 9 and if not add extra 0 value rows and columns so that a 3X3 window can be used.

Step 3: Divide the whole image into 9 equal sized zones.

Step 4: For each zone, determine the line type and the following 9 features.

1. Number of horizontal lines.
2. Number of vertical lines.
3. Number of Right diagonal lines.
4. Number of Left diagonal lines.
5. Normalized Length of all horizontal lines.
6. Normalized Length of all vertical lines.
7. Normalized Length of all right diagonal lines.
8. Normalized Length of all left diagonal lines.
9. Normalized Area of the Skeleton.

Step 5: Concatenate the 9 features of all the zones to form a vector of total 81 features consisting of 9 features from each of the 9 zones.

2. Algorithm for Region properties

Input: Each segmented character image

Output: Feature Vector of size 52

Algorithm:

Step 1: Determine the connected component in the preprocessed image using the method bwconncomp(BW).

Step 2: Label the component using the method bwlabel(BW).

Step 3: Calculate Area as the product of the number of rows and columns of the bounding box of the character.

Step 4: Calculate solidity as

S= Area/ (Convex Area);

Where: Area= rows*columns

Convex Area= the area of the minimum convex polygon enclosing the character

Step 5: Orientation is calculated as the angle between the Major axis and the x-axis.

Step 6: To calculate Perimeter, check for 4-connected pixels. Add each 4-connected pixel that has at least one neighbor as background and one neighbor as object. The total count is the perimeter [16].

Step 7: To calculate Major Axis Length, find 2 pixels on the character with the maximum distance. The line joining them is the Major axis and the distance between them is its length.

Step 8: The line perpendicular to the major axis at its midpoint is the minor axis. Find the distance between 2 intersections at both ends of the perpendicular bisector. This is the Minor axis.

Step 9: Find the distance between the 2 foci. The ratio of the distance between the foci and the length of the major axis is the Eccentricity.

Step 10: EquivDiameter is computed as sqrt (4*Area/pi). It specifies the diameter of the circle with the same area as region [16].

Step 11: Determine the Reflection X coefficient and Reflection Y coefficient [17].

Algorithm:

1. Consider Cinas the original character matrix where information is stored in binary format. The center of matrix is considered as the origin. Cx and Cy are the transformed matrix obtained after flipping about the X and Y axes respectively.

2. Calculate the area of original matrix Cin.

3. Determine the union of matrixes Cx and Cy with original matrix Cin.

4. Calculate the areas of regions (Cin∪Cx) and (Cin∪Cy).

5. The reflection coefficient can be determined as

Xcoefficient=Area (Cin)/ Area (Cin∪Cx);

Ycoefficient=Area (Cin)/ Area (Cin∪Cy);

Step 12: Calculate the horizontal and the vertical profile of the character image using the mean method over the image matrix.

4.1.5 Classification using Error Back Propagation Training Algorithm

Input: Feature vector database, desired outputs

Output: String output for individual image segmented character

Algorithm:

Given are P training pairs $\{z_1, d_1, z_2, d_2, \cdots, z_p, d_p\}$

Where $z_i$is $(I\ x\ 1)$, d_i is $(K\ x\ 1)$, and $i = 1,2,\dots,P$. Ith component of each z_i is of value -1 since input vectors have been augmented. Jth component of y is of value -1, since hidden layer outputs have also been augmented; y is $(J\ x\ 1)$ and o is $(K\ x\ 1)$ [20].

K = 16 (for Euler number 0), J = 15 (for best trained network), K = 16 (for Euler no. 0)

Step 1: η > 0.05, E_{max} =0.01.

Weights W and V are initialized at small random values; W is$(K\ x\ J)$, $V$is $(J\ x\ I)$.

$$q \leftarrow 1, p \leftarrow 1, E \leftarrow 0$$

Step 2: Training step starts here. Input is presented and the layers' output computed using

$$f(net) \triangleq sgn(net) = \begin{cases} +1, & net\ > 0 \\ -1, & net\ < 0 \end{cases} \tag{4.8}$$

$$z \leftarrow z_p, d \leftarrow d_p$$

$$y_i \leftarrow f\left(v_j^t z\right), \text{for } j = 1, 2, \dots, J \tag{4.9}$$

where v_j, a column vector, is the jth row of V, and

$$o_k \leftarrow f(w_k^t y), \text{for } k = 1, 2, \dots, K \tag{4.10}$$

where w_k, a column vector, is the kth row of W.

Step 3: Error value is computed:

$$E \leftarrow \frac{1}{2}(d_k - o_k)^2 + E, \text{ for } k = 1, 2, \dots, K \tag{4.11}$$

Step 4: Error signal vectors δ_o and δ_y of both layers are computed.

Vector $\delta_o$is $(K\ x\ 1)$, δ_y is $(J\ x\ 1)$.

Error signal terms are computed as follows

$$\delta_{ok} = (d_k - o_k)(1 - o_k)o_k, \text{ for } k = 1, 2, \dots, K \tag{4.12}$$

$$\delta_{yj} = y_j\left(1 - y_j\right)\sum_{k=1}^{K} \delta_{ok} w_{kj}, \text{ for } j = 1, 2, \dots, J \tag{4.13}$$

Step 5: Output layer weights are adjusted:

$$w_{kj} \leftarrow w_{kj} + \eta\delta_{ok}y_j, \text{ for } k = 1, 2, \dots, K \text{ and j} = 1, 2, \dots, J \tag{4.14}$$

Step 6: Hidden layer weights are adjusted:

$$v_{ji} \leftarrow v_{ji} + \eta\delta_{yj}z_i, \text{ for } j = 1, 2, \dots, J \text{ and i} = 1, 2, \dots, I \tag{4.15}$$

Step 7: If $p\ < P$then$p \leftarrow p + 1$, $q \leftarrow q + 1$, and go to Step 2; otherwise go to step 8.

Step 8: The training cycle is complete.

For $E < E_{max}$ terminate the training session. Output weights are$W$, V, q and E.

If$E > E_{max}$, then$E \leftarrow 0$, $p \leftarrow 1$, and initiate new training cycle by going to Step 2 [20].

4.1.6 Algorithm for Post-processing

Input: The string output generated by the neural network classifier.

Output: String output with corrections based on properties of CAPTCHA.

Algorithm:

Step 1: Find the class of CAPTCHA from the path provided.

Step 2: Check if the output string is of the correct length according to the properties of the CAPTCHA.

Step 3: Check the output for any possible errors by consulting the testing results for frequently misclassified characters. Correct those characters to the probable output characters.

4.1.7 Steps in Recall and Generalization

Step 1: For each character in the set, generate the FV required for the network using which the character is to be classified.

Step 2: Simulate the character's output by providing the feature vector to the neural network using sim() function:

```
sim(net,FV)
```

Where net is the network and FV is the feature vector.

Step 3: Calculate the errors between the obtained output and the desired output.

Step 4: Calculate the performance of the trained network based on the desired outputs and the obtained outputs.

Step 5: If the obtained simulation results are same as the desired output for that particular character, then the character is getting classified accurately. Otherwise, the character is not classified properly and hence, contributes to the error.

4.2 Implementation Issues

The first issue is to segment characters that are joint to each other. Since we find out the connected components in the image and segment it accordingly hence the joint characters are considered as one component and segmented incorrectly.

The background color of the noise varies from a very low intensity to a very high intensity in the CAPTCHA image; hence there can’t be a single thresholding value on which the image can be binarized. Hence 2 thresholds are considered one of the first half of the image and other for the second half.

The characters on which the neural network is trained are highly distorted in all sense and hence obtaining excellent generalization accuracy is very difficult.

4.3 Results

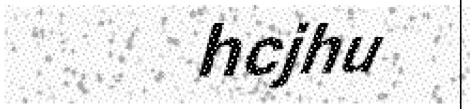
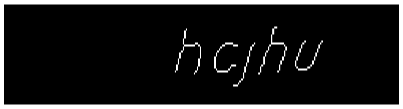
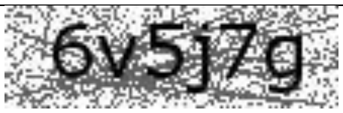
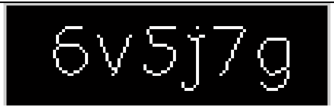
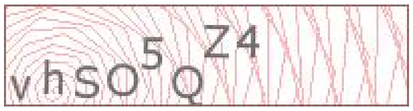
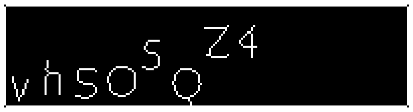
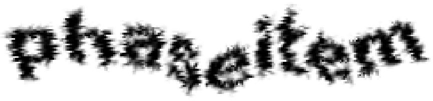
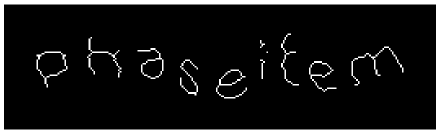
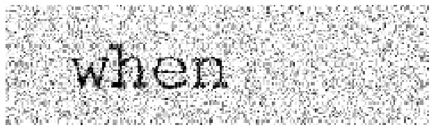
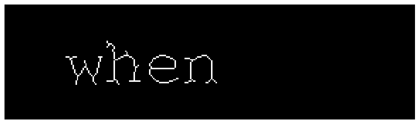
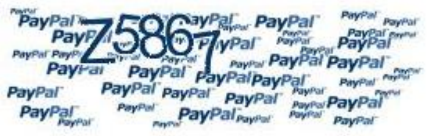
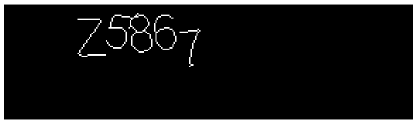
In this section, the results of the pre-processing phase, segmentation phase, feature extraction and the training phase are shown.

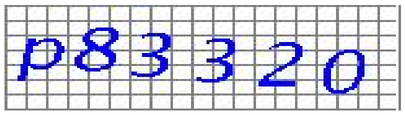
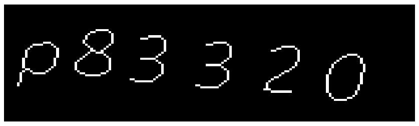
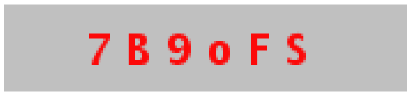
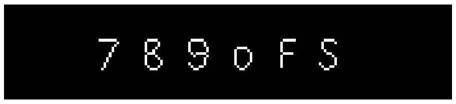
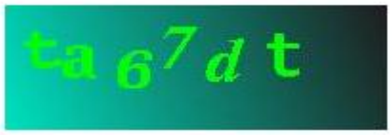
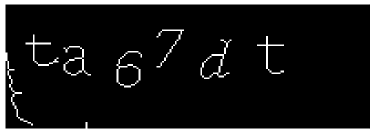
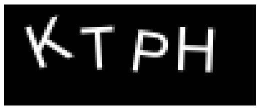
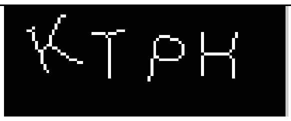
4.3.1 Pre-processing Results

The original CAPTCHA image is preprocessed into a thin, binarized and a noise free image ready to be segmented easily. The accuracy of the CAPTCHA to be recognized correctly starts right from its preprocessing phase. If the noise in the original image is retained, it degrades the accuracy of the later phases. The preprocessing engine for 10 different CAPTCHA images as each CAPTCHA has its own unique characteristic noise and distortion implementations.

The original image and its preprocessed image are shown for different CAPTCHAs in table 4.1.

Table 4.1 Pre-processing Results

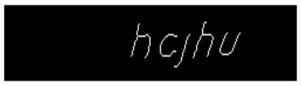
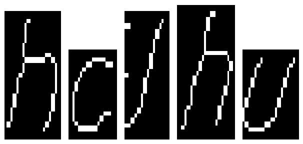
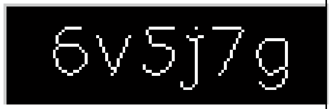
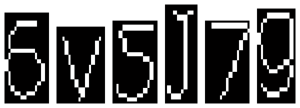
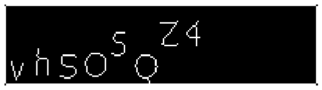
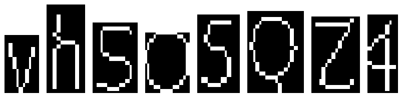
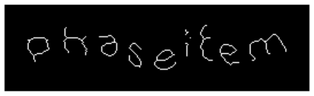
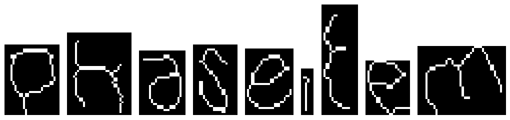
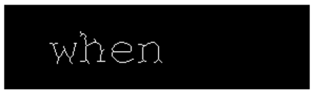
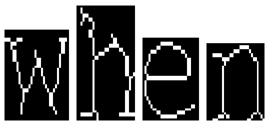
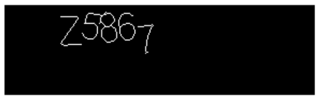
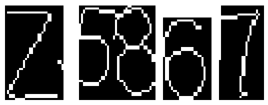
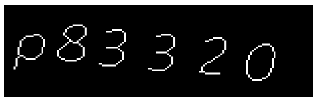
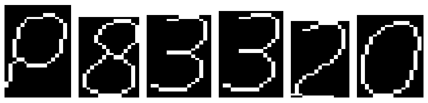
CAPTCHA	Original Image	Preprocessed Image
Authorize		
MP3Raid		
GoJiyo		
Wikipedia		
EZ Gimpy		
PayPal		

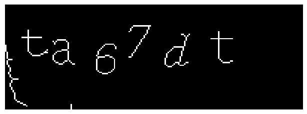
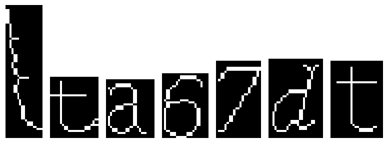
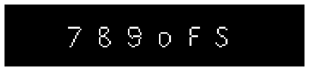
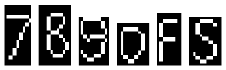
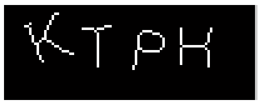
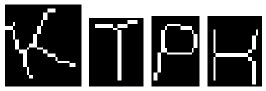
Passport		
Vodafone		
Register		
IIM Kozhikode		

4.3.2 Segmentation Results

The accuracy of the classifier vastly depends upon the accuracy of the segmentation phase. Each CAPTCHA is segmented using a universal algorithm of applying a bounding box over the each connected component i.e. character. The characters in the image are segmented accurately only if they are disjoint. Segmentation was tested on 100 images of 10 CAPTCHAs and the following results were obtained. The segmentation results for all the classes of CAPTCHA considered are as displayed ahead in table 4.2.

Table 4.2 Segmentation Results

CAPTCHA	Preprocessed Image	Segmented Images	Accuracy %
Authorize			90
MP3Raid			100
GoJiyo			100
Wikipedia			85
EZ Gimpy			70
PayPal			75
Passport			99

Register			90
Vodafone			100
IIM Kozhikode			75

4.3.3 Feature Extraction Results

This section gives the feature extraction results for Zone based feature extraction, Region Properties based feature extraction, Horizontal and Vertical Profiles based feature extraction. All the reults have been discussed for intra character similarities and inter character similarities.

1. Intra Character Similarity

Stem plots for 4 images of the same character is plotted to analyze the intra character similarity in the different of various feature extraction techniques.

In figure 4.1, the X-axis represents 4 images of character ‘0’ and Y-axis represents the range of the feature vector values using the centroid based zoning. The range of values is from 0 to 20. It is seen that corresponding features have almost the same range of values.

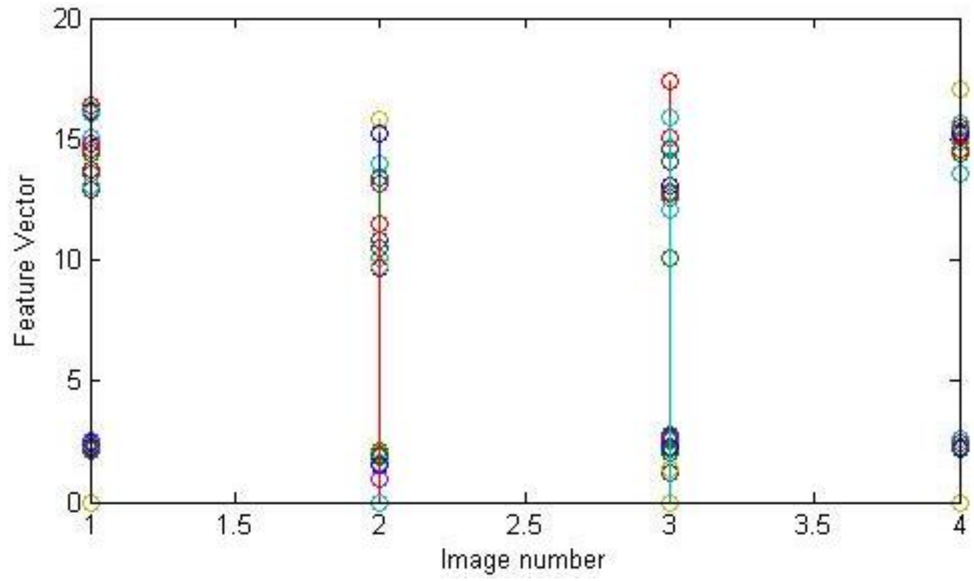


Figure 4.1 Intra Character Similarity in Centroid Based Zoning

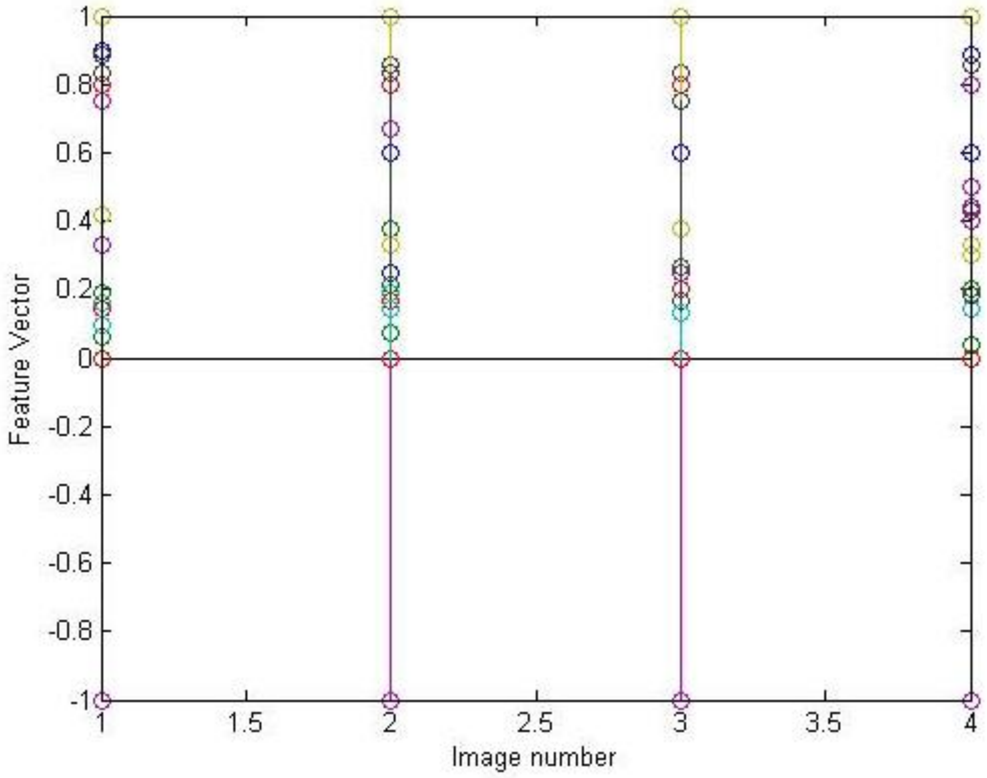


Figure 4.2 Intra Character Similarity in Contour Based Zoning

In figure 4.2, the X-axis represents 4 images of character '1' and Y-axis represents the range of the feature vector values using the contour based zoning. The range of values is from -1 to +1. It is seen that corresponding features have almost the same range.

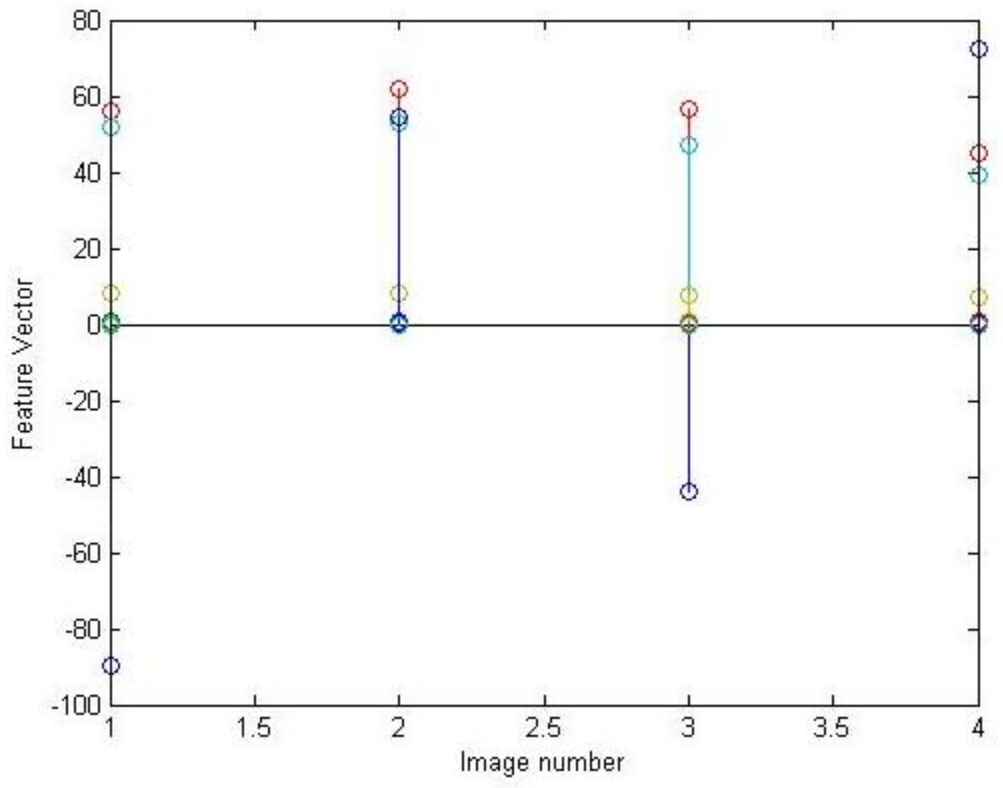


Figure 4.3 Intra Character Similarity in Region Properties + Horizontal and Vertical Profiles

In figure 4.3, the X-axis represents 4 images of character '0' and Y-axis represents the range of the feature vector values using the region properties and profiles. The range of values is from 0 to 80. It is seen that corresponding features have almost the same range of values.

2. Inter Character Differences

Stem plots for 4 different character images are plotted to analyze the inter character differences in the different of various feature extraction techniques.

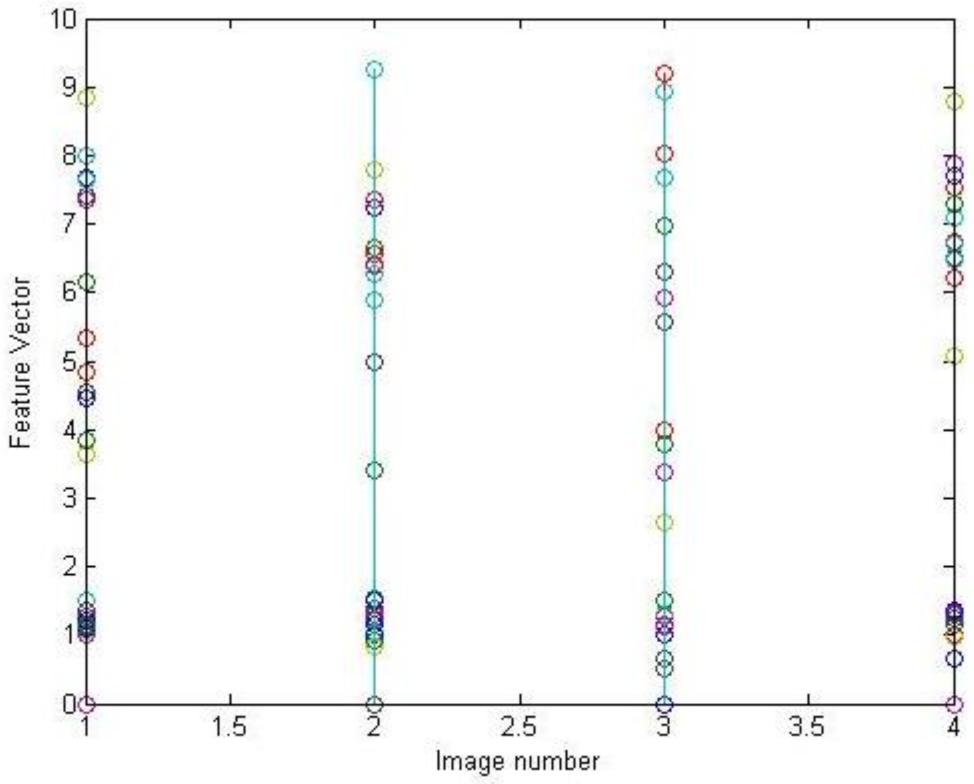


Figure 4.4 Inter Character Similarity in Centroid based Zoning

The X-axis represents 4 images of characters ‘d’, ’e’, ’g’, ’o’ and Y-axis represents the range of the feature vector values using the centroid based zoning as shown in figure 4.4. The range of values is from 0 to 10. It is seen that corresponding features have different range of values.

In figure 4.5 ahead, the X-axis represents 4 images of characters ‘7’, ’c’, ’f’, ’h’ and Y-axis represents the range of the feature vector values using the centroid based zoning. The range of values is from -1 to +1. It is seen that corresponding features have different range of values.

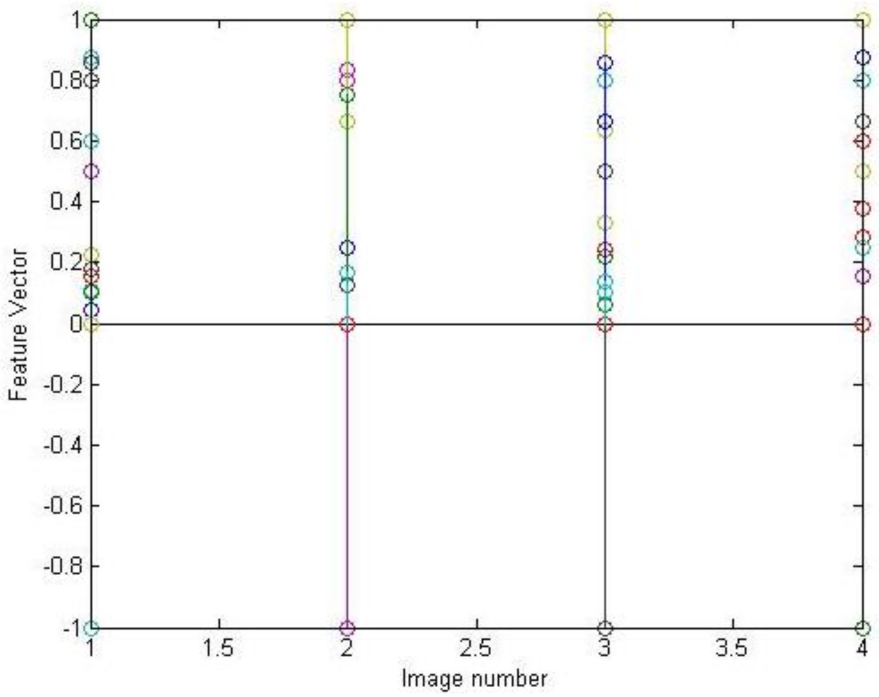


Figure 4.5 Inter Character Similarity in Contour based zoning

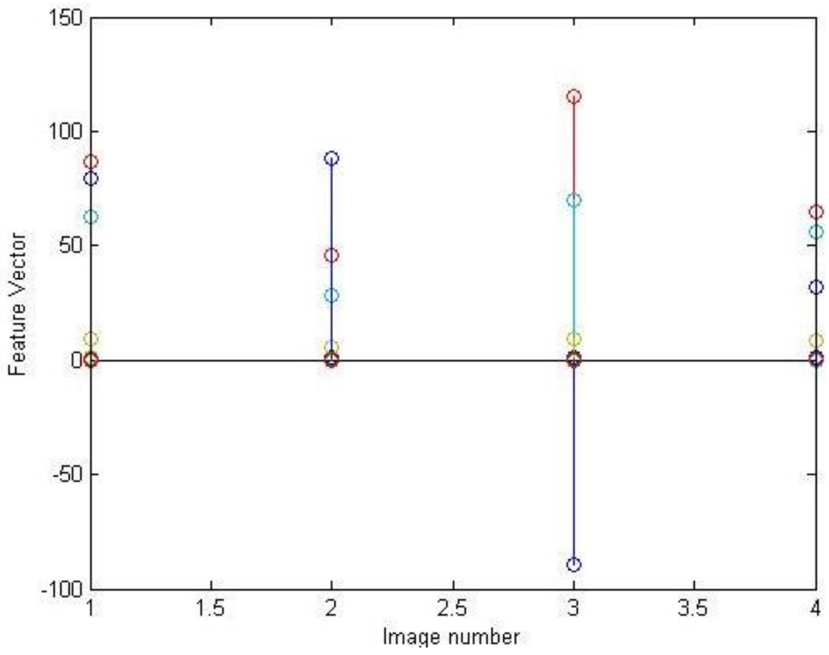


Figure 4.6 Inter Character Similarity in Region Properties + Horizontal and Vertical Profiles

In the figure 4.6, the X-axis represents 4 images of characters ‘d’, ’e’, ’g’, ’o’ and Y-axis represents the range of the feature vectors using the region properties and profiles. The range of values is from -100 to 150. It is seen that corresponding features have different range of values.

4.3.4 Training

Training results for various combinations of parameters have been discussed below.

1. Characters having Euler Number -1

Training was done on 40 images per character and testing was done on 10 images per character and the 100% of the characters were correctly classified. The table 4.3 shows the performance of X different networks trained on characters having Euler number -1 and with different feature vectors and different network parameters with constant goal 0.01.

Table 4.3 ANN Training Results (Euler number -1)

Feature Extraction technique	Feature vector size	Training function	Hidden neurons	Lear ning Rate	Momen tum	Goal State	Epochs	Recall
Zoning (contour)	81	traingd	1	0.4	0.1	0.0079	576	100%

Performance Plot:

The performance plot of the network is shown ahead in figure 4.7. It indicates that the performance is not steady. It increases and decreases rapidly with epochs and finally reaches its goal at epoch no 576.

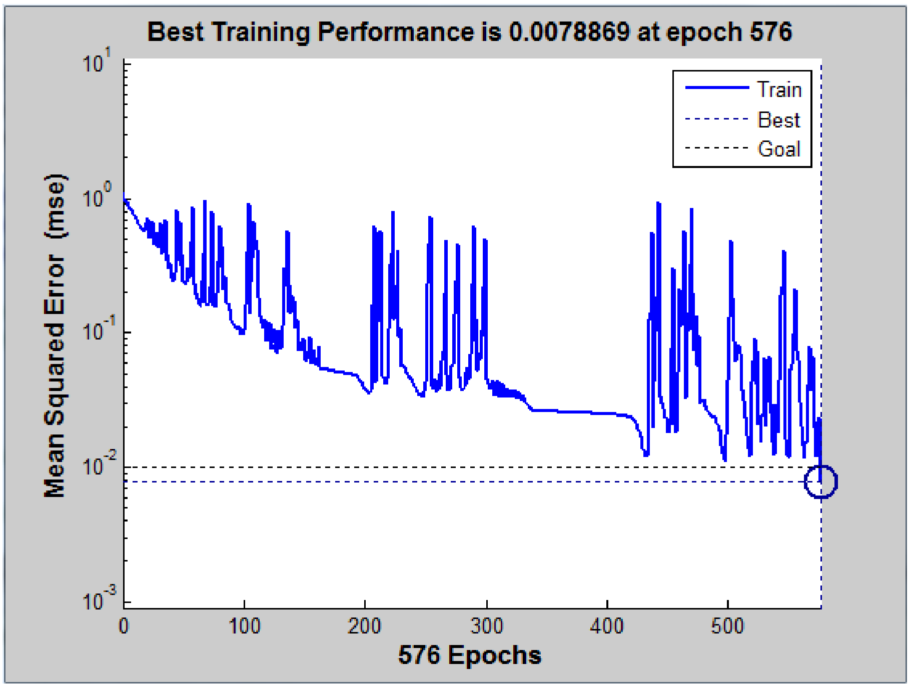


Figure 4.7 Performance Plot for best trained network of Euler no. -1 characters

<u>Training State Plot:</u>

The training state of the network is plotted as shown in figure 4.8 and it indicates frequent changes in the gradient value. The minimum gradient value that is reached is 0.118.

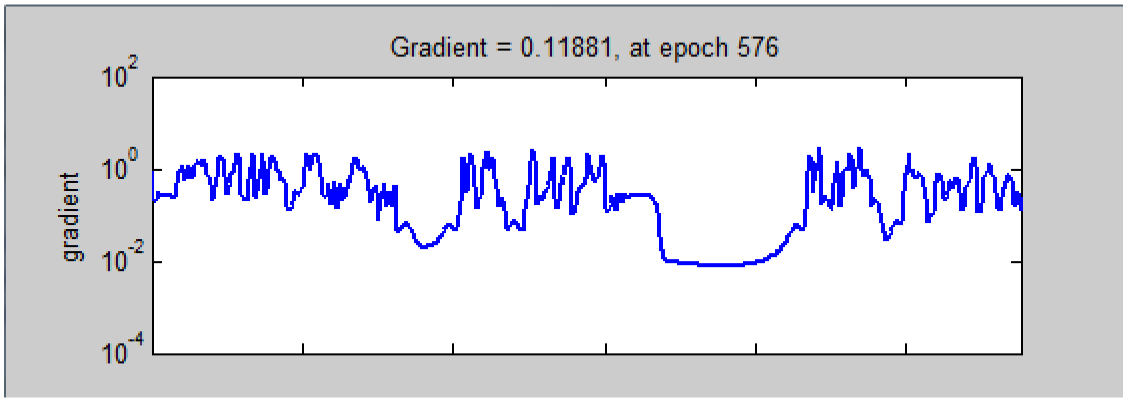


Figure 4.8 Training State Plot for best trained network of Euler no. -1 characters

2. Characters having Euler Number 0

Table 4.4 ANN Training Results (Euler number 0)

Goal = 0.01								
Feature Extraction technique	Feature vector size	Training function	Hidden neurons	Learning Rate	Momentum	Goal State	Epochs	Recall %
<u>Regionprops + profiles</u>	<u>52</u>	<u>traingdx</u>	<u>15</u>	<u>0.6</u>	<u>0.2</u>	<u>0.0099</u>	<u>15771</u>	<u>97.81</u>
Regionprops + profiles	52	traingd	15	0.6	0.2	0.0173	347444	93.33
Zoning (centroid) + Regionprops	40	traingd	15	0.6	0.2	0.0165	319876	91.33
Zoning (centroid) + Regionprops	40	traingd	14	0.7	0	0.0202	318377	90
Zoning (contour)	32	traingd	14	0.6	0.2	0.0266	288674	90.11
Zoning (centroid)	32	traingd	15	0.5	0	0.0290	177674	89.06
Zoning + profiles	76	traingd	15	0.5	0.2	0.0341	149293	85.34

Training was done on 40 images per character and testing was done on 10 images per character and the 60% of the characters were correctly classified. The table 4.4 shows the performance of X different networks trained on characters having Euler number 0 and with different feature vectors and different network parameters with constant goal 0.01.

Performance Plot:

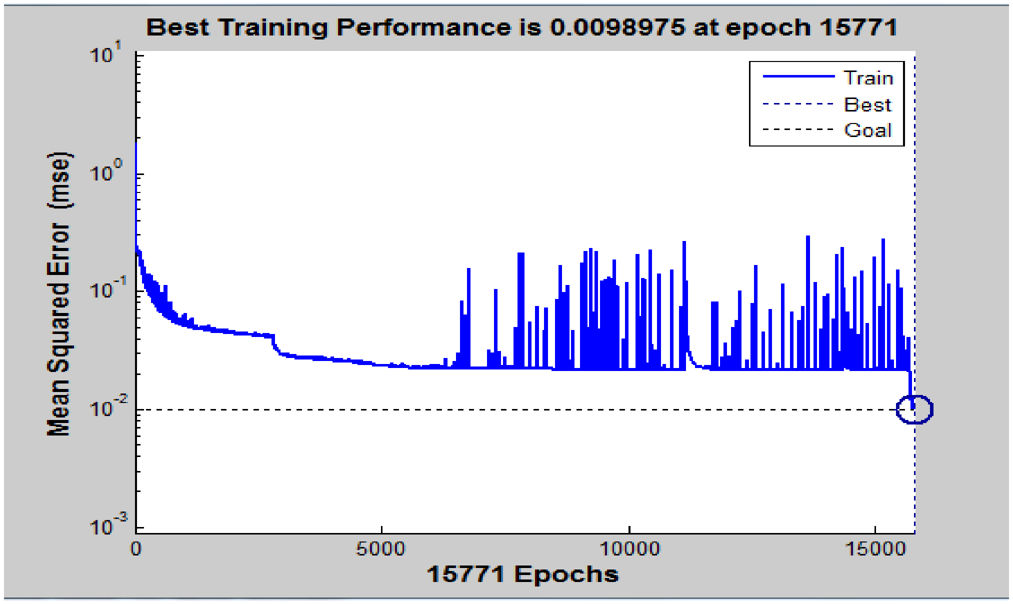


Figure 4.9 Performance Plot for best trained network of Euler no. 0 characters

The performance plot of the network is shown in Figure 4.9. It indicates that the mean squared error has a steep drop at around 3000 epoch and since adaptive learning rate is considered The performance graph is very irregular and the goal is met at 15771 epoch.

Training State Plot:

The training state of the network is shown below in figure 4.10 wherein the rise and fall of the gradient value. It also indicates the increase and decrease of the learning rate. The learning rate is increased if the performance value goes closer to the goal else the learning rate decreases.

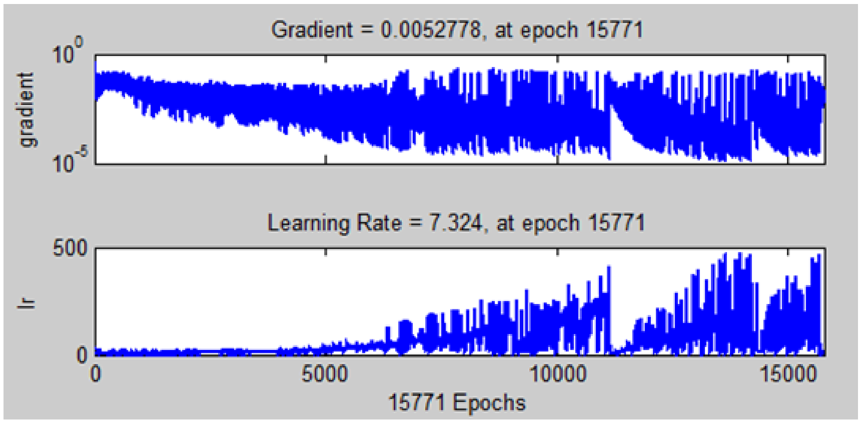


Figure 4.10 Training State Plot for best trained network of Euler no. 0 characters

3. Characters having Euler Number 1

Training was done on 40 images per character and testing was done on 10 images per character and the 60% of the characters were correctly classified. The table 4.5 shows the performance of X different networks trained on characters having Euler number 1 and with different feature vectors and different network parameters with constant goal 0.01.

Table 4.5 ANN Training Results (Euler number 1)

Goal = 0.01								
Feature Extraction technique	Feature vector size	Training function	Hidden neurons	Learning Rate	Momentum	Goal State	Epochs	Recall %
Zoning (contour)	81	traingd	31	0.7	0.3	0.0210	257870	84
Zoning (centroid)+ regionprops	40	traingd	30	0.5	0.2	0.0324	156944	74

Zoning (centroid)	32	traingd	29	0.7	0.3	0.0301	274531	73
Hybrid	55	traingd	31	0.6	0.2	0.0393	268788	68.98
Zoning + profiles	76	traingd	31	0.6	0.3	0.0598	110034	52
Regionprops + profiles	52	traingd	31	0.5	0.3	0.0712	43455	59

<u>Performance Plot:</u>

The performance plot of the network is shown in figure 4.11. It can be seen that the performance value gradually decreases and remains constant at 0.0210 at Epoch number 257869.

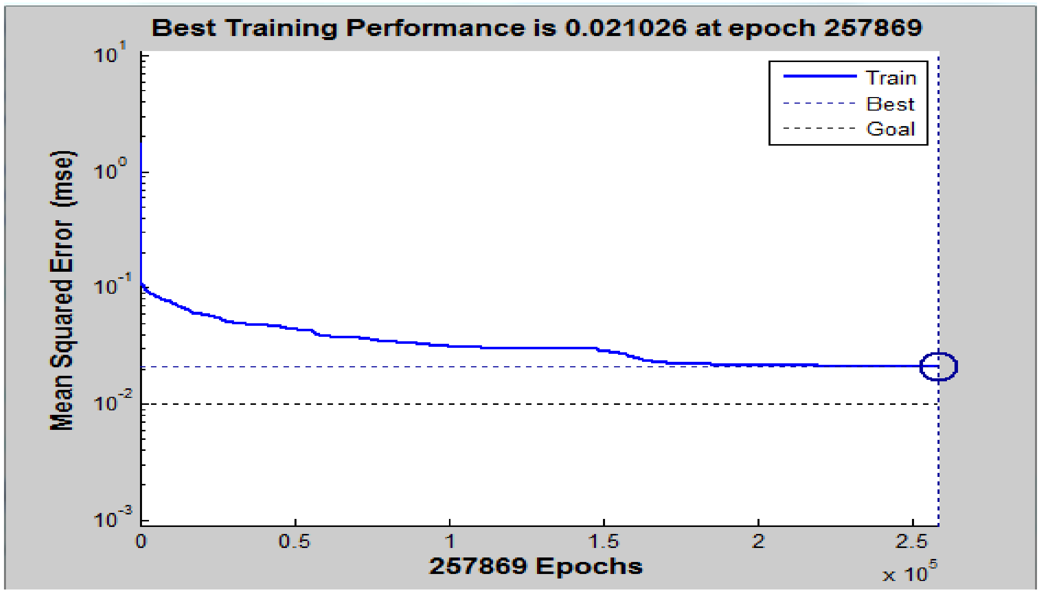


Figure 4.11 Performance Plot for best trained network of Euler no. 1 characters

Training State Plot:

The training state of the network is plotted as shown in figure 4.12 and it indicates frequent changes in the gradient value. The minimum gradient value that is reached is 7.1032e-005.

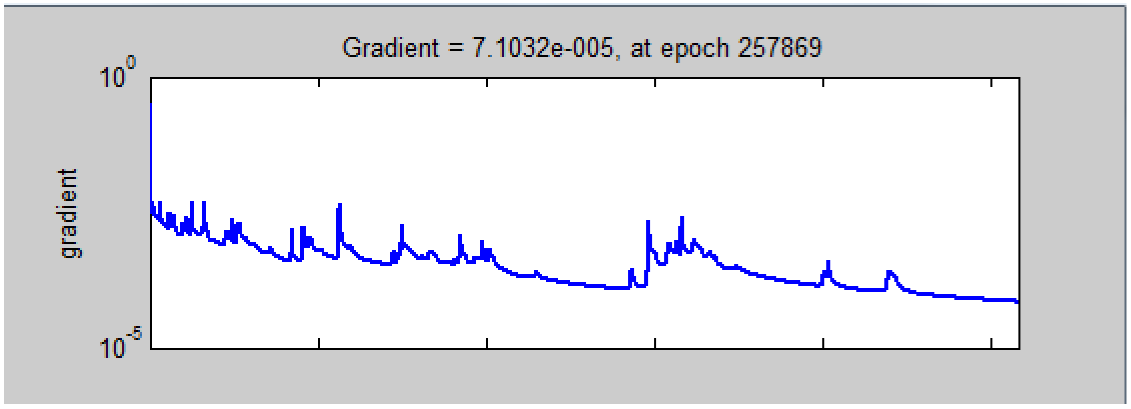


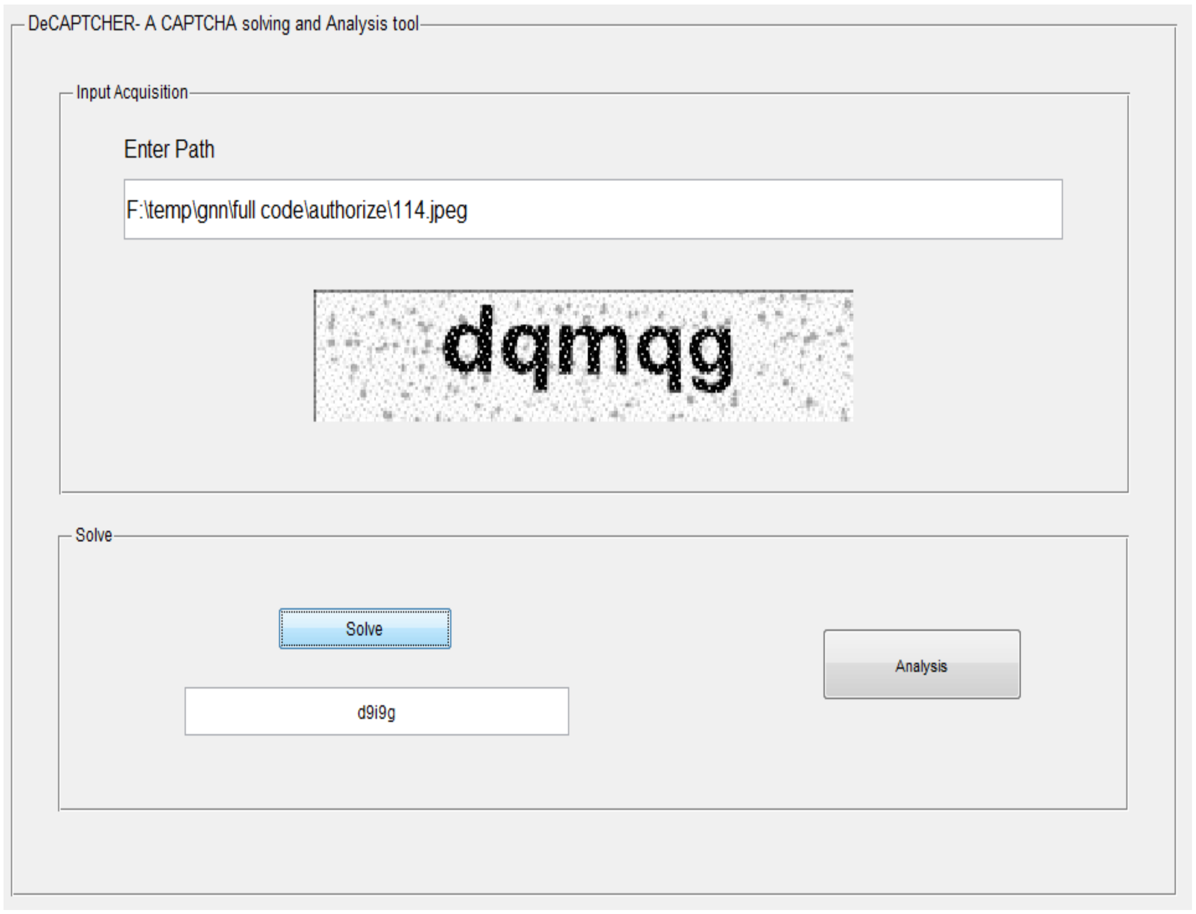
Figure 4.12 Training State Plot for best trained network of Euler no. 1 characters

4.3.5 Graphical User Interface (GUI)

The GUI of the system has 2 screens

1. The Solver screen:

The Solver screen is as shown below in figure 4.13. It takes the path of the image as input, reads the image and shows the original image on the screen. The solve button takes a few seconds to process the image through the system and displays back the result in the provided text box. A button to the optional analysis screen is presented for further analysis of the CAPTCHA image.



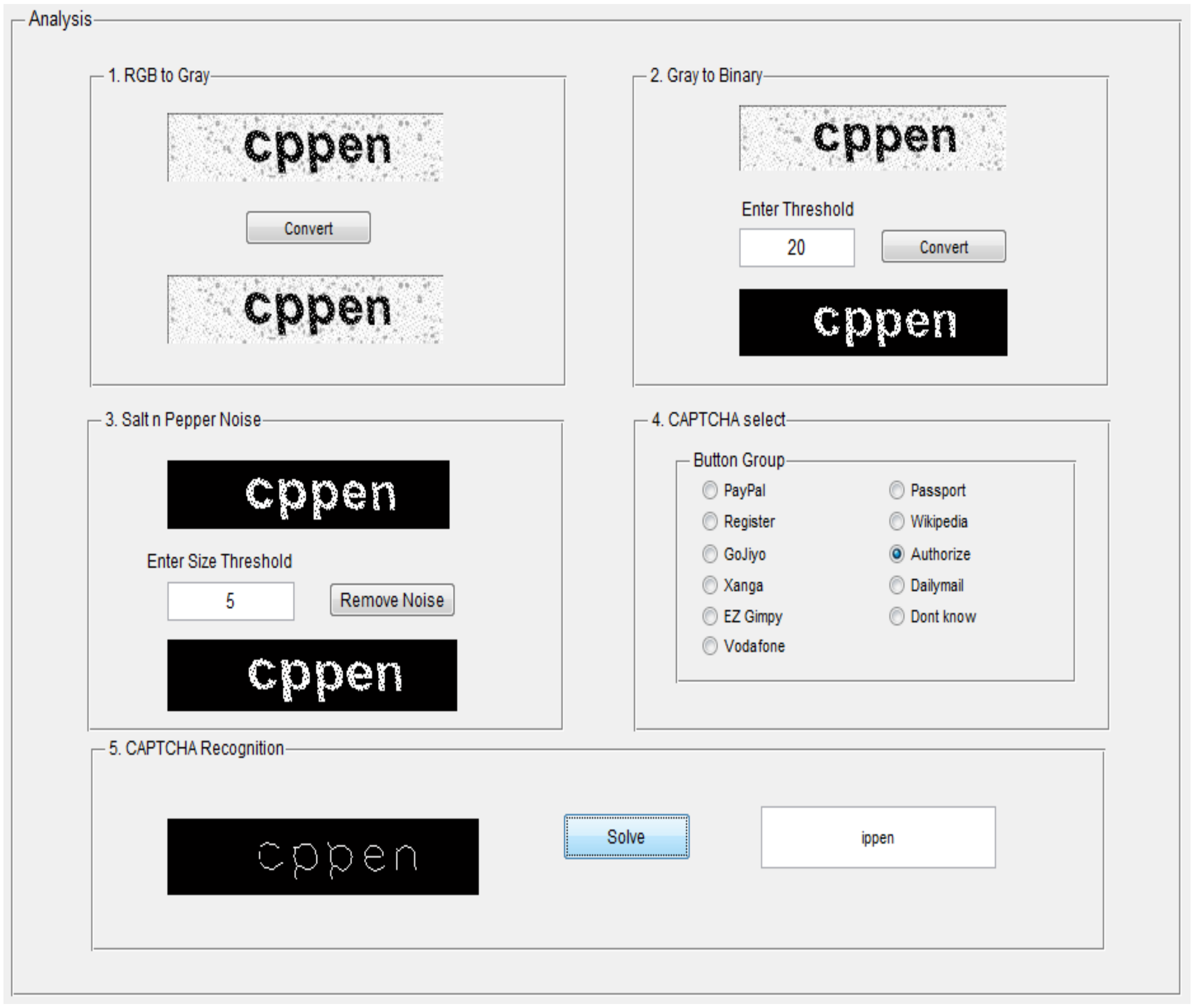
F:\temp\gnn\full code\authorize\114.jpeg

d9i9g

Figure 4.13 Solver screen

2. The Analysis screen:

The second screen is an analysis screen where the CAPTCHA image can be analyzed at the preprocessing phase by manually changing the preprocessing parameters of thresholding and noise removal. Figure 4.14 shows the analysis screen where the result of every preprocessing function is displayed.



Enter Threshold: 20

Enter Size Threshold: 5

PayPal [ ] Passport [ ] Register [ ] Wikipedia [ ] GoJiyo [ ] [✓ Authorize] Xanga [ ] Dailymail [ ] EZ Gimpy [ ] Dont know [ ] Vodafone [ ]

ippen

Figure 4.14 Analysis Screen

5. TESTING

Software testing provides an objective, independent view of the software in order to understand the risks and issues of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing can be stated as the process of validating and verifying that a software program/application/product meets the requirements that guided its design and development [21].

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. Testing was carried out in different levels as follows.

5.1 Unit Testing

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other [21]. The following units in the system were tested as individually:

1. <u>Input Acquisition</u>

The CAPTCHA source URL is taken as input and the type of CAPTCHA is identifies by matching for the string in the URL. This test assures that the type of CAPTCHA is always correctly identified for further processing.

Image_path= https://www.paypal.com/cgi-bin/gs_web/6wrRsgg-1iwTgePNndNOtBSF-lfePW8iKAzt.QrjJ3CSaPpK3kRZDQI95Ps0jAwxoU31Gg/secret.jpeg?version=

CAPTCHA type = PAYPAL

2. Preprocessing

Every original CAPTCHA image is preprocessed to binarize it, remove the unwanted noise and finally thin it to be used for segmentation. All CAPTCHAs are tested for preprocessing to accurately remove the noise without distorting the characters any further. Preprocessing for a test image, shown in the figure 5.1, is shown in figure 5.2.

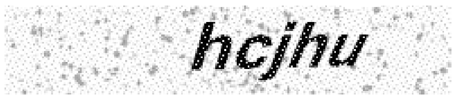


Figure 5.1 Original image

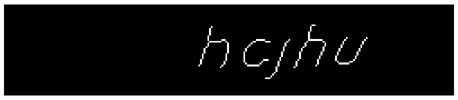


Figure 5.2 Preprocessed image

3. Segmentation

The preprocessed image is the segmented into its distinct characters for individual recognition. It is important that all characters are disjoint since our system does not work for joint characters. For disjoint characters as shown in figure 5.3, 100% accuracy has been achieved. The segmented characters of the CAPTCHAs of joint and disjoint characters are shown below. A test case where joint characters are obtained is shown in figure 5.4.

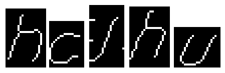


Figure 5.3 CAPTCHA with disjoint characters

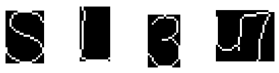


Figure 5.4 CAPTCHA with joint characters

4. <u>Character Recognition</u>

The segmented characters are recognized individually based on the extracted features. The module is separately tested to determine the no of characters that the trained network properly classifies and determine the recall and generalization accuracy of the network.

5. <u>GUI</u>

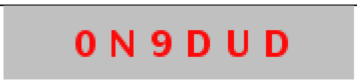
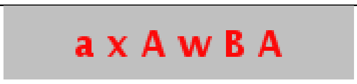
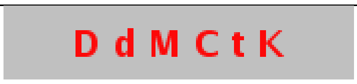
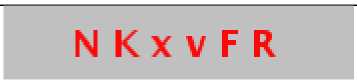
The GUI was tested by keeping the user requirements in mind. It was tested for user friendliness and ease of use.

5.2 System Testing

System testing tests a completely integrated system to verify that it meets its requirements. The system was tested on the following website CAPTCHAs.

1. Vodafone

Table 5.1 System Testing (Vodafone)

CAPTCHA Images	Testing Output
	0N9DuD
	axAwBA
	Ddmctk
	NkxvFR

Character Accuracy:

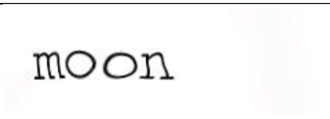
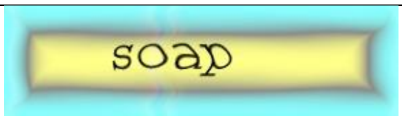
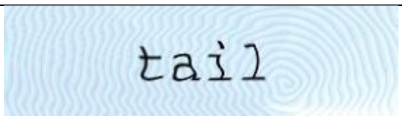
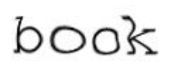
Out of 6 characters in the CAPTCHA, 4 or 5 characters were correctly classified resulting in 80% character accuracy.

CAPTCHA Accuracy:

The system was tested on 50 Vodafone CAPTCHA's and all of them were correctly recognized hence resulting in 75% CAPTCHA recognition accuracy.

2. EZ Gimpy

Table 5.2 System Testing (EZ Gimpy)

CAPTCHA Images	Testing Output
	Meon
	5oap
	ta11
	Booh

Character Accuracy:

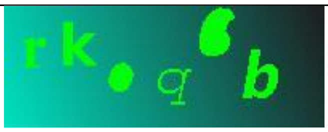
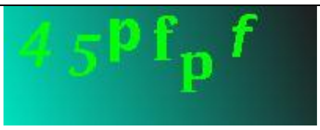
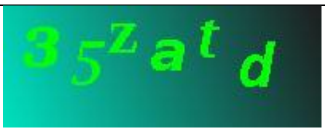
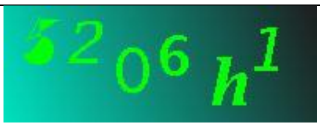
On an average out of 5 characters in the CAPTCHA, 3 of them were correctly classified resulting in almost 60% character accuracy.

<u>CAPTCHA Accuracy:</u>

The system was tested on 50 EZ gimpy CAPTCHAs and 5 were correctly recognized hence resulting in 10% CAPTCHA recognition accuracy.

3. Register

Table 5.3 System Testing (Register)

CAPTCHA Images	Testing Output
	rkqb
	45plpf
	35zar6
	20gh1

<u>Character Accuracy:</u>

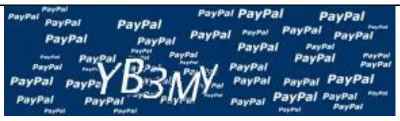
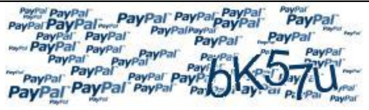
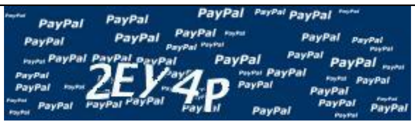
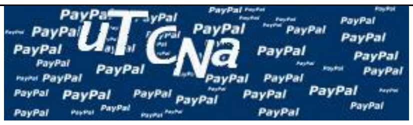
Out of 6 characters in the CAPTCHA, 2 to 3 were correctly classified resulting in almost 40% character accuracy.

<u>CAPTCHA Accuracy:</u>

The system was tested on 50 Register CAPTCHAs and 1 was correctly recognized hence resulting in 2% CAPTCHA recognition accuracy

4. PayPal

Table 5.4 System Testing (PayPal)

CAPTCHA Images	Testing Output
	yB3My
	bk57u
	2Ey4p
	uTcNe

<u>Character Accuracy:</u>

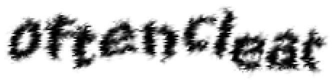
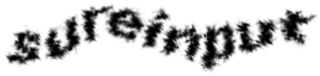
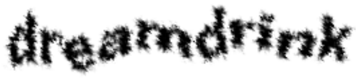
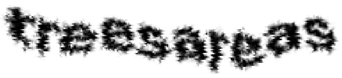
Out of 5 characters in the CAPTCHA, 3 to 4 were correctly classified resulting in almost 75% character accuracy.

<u>CAPTCHA Accuracy:</u>

The system was tested on 50 PayPal CAPTCHAs and 13 were correctly recognized hence resulting in 26% CAPTCHA recognition accuracy.

5. Wikipedia

Table 5.5 System Testing (Wikipedia)

CAPTCHA Images	Testing Output
	ortencjeal
	sureinpuc
	dreerdrink
	treesa1ees

Character Accuracy:

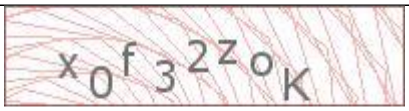
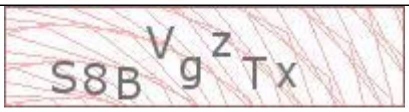
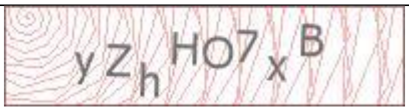
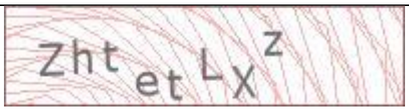
Out of 8-10 characters in the CAPTCHA, 3-4 characters were correctly classified resulting in 40% character accuracy.

CAPTCHA Accuracy:

The system was tested on 50 Wikipedia CAPTCHAs and 1 was correctly recognized hence resulting in 2% CAPTCHA recognition accuracy.

6. GoJiyo

Table 5.6 System Testing (GoJiyo)

CAPTCHA Images	Testing Output
	x0f3tzok
	58BvgzTx
	vzhHo7xB
	zhcecLxz

<ins>Character Accuracy:</ins>

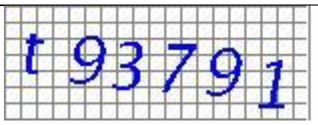
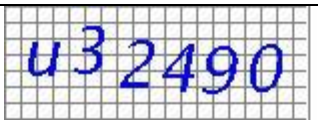
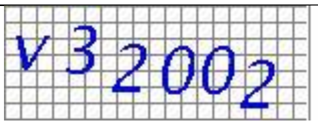
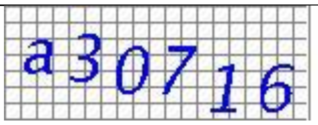
Out of 8 characters in the CAPTCHA, 5 to 6 were correctly classified resulting in almost 70% character accuracy.

<ins>CAPTCHA Accuracy:</ins>

The system was tested on 50 Gojiyo CAPTCHA’s and 2 were correctly recognized hence resulting in 4% CAPTCHA recognition accuracy.

7. Passport

Table 5.7 System Testing (Passport)

CAPTCHA Images	Testing Output
	t93791
	u32490
	v32002
	a30716

<ins>Character Accuracy:</ins>

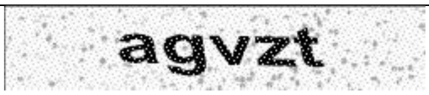
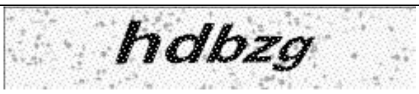
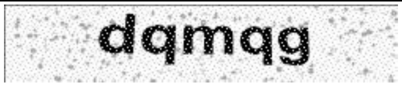
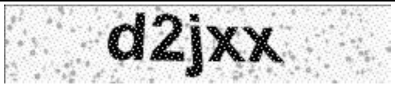
Out of 5 characters in the CAPTCHA, most of the characters were correctly classified resulting in almost 100% character accuracy.

<ins>CAPTCHA Accuracy:</ins>

The system was tested on 50 Passport CAPTCHA's and 48 were correctly recognized hence resulting in less than 96% CAPTCHA recognition accuracy.

8. Authorize

Table 5.8 System Testing (Authorize)

CAPTCHA Images	Testing Output
	agvzt
	hdbzg
	d9m9g
	d2jxx

<ins>Character Accuracy:</ins>

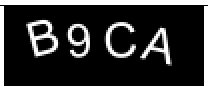
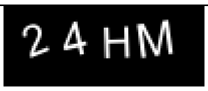
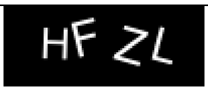
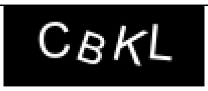
Out of 5 characters in the CAPTCHA, 3 to 4 were correctly classified resulting in almost 75% character accuracy.

<ins>CAPTCHA Accuracy:</ins>

The system was tested on 50 Authorize CAPTCHA's and 20 were correctly recognized hence resulting in 40% CAPTCHA recognition accuracy.

9. IIM Kozhikode

Table 5.9 System Testing (IIM Kozhikode)

CAPTCHA Images	Testing Output
	B9CA
	24HM
	HFZi
	C8KL

<ins>Character Accuracy:</ins>

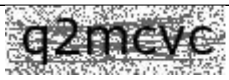
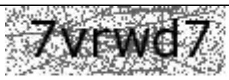
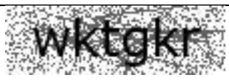
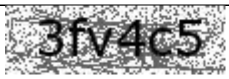
Out of 4 characters in the CAPTCHA, almost 3 were correctly classified resulting in 75% character accuracy.

<ins>CAPTCHA Accuracy:</ins>

The system was tested on 50 IIM Kozhikode CAPTCHA's and 22 were correctly recognized hence resulting in 44% CAPTCHA recognition accuracy.

10. MP3Raid

Table 5.10 System Testing (MP3Raid)

CAPTCHA Images	Testing Output
	gzmcvc
	7vrwd7
	wktgkr
	3fv4c5

Character Accuracy:

Out of 6 characters in the CAPTCHA, almost 5 were correctly classified resulting in almost 83% character accuracy.

CAPTCHA Accuracy:

The system was tested on 50 MP3Raid CAPTCHA's and 18 were correctly recognized hence resulting in 36% CAPTCHA recognition accuracy.

6. CONCLUSION AND FURTHER WORK

This chapter majorly focuses on the conclusion and further work which can be carried out in this domain with a view to enhance this project.

6.1 Conclusion

CAPTCHAs have been one of the most potent mechanisms to protect web applications against automated form submissions. However, a weak CAPTCHA design may only protect against random robots and may not offer any protection against targeted attempts to bypass it. DECAPTHER aims to enable security professionals, developers, and testers to evaluate their CAPTCHA designs and make their implementations more secure. Like cryptographic algorithms, it is in an organization’s best interest to rely on CAPTCHAs that have been thoroughly tested and are proven to be highly secure.

DECAPTCHER was developed using Artificial Neural Networks. Error Back Propagation Training Algorithm was used to train the neural networks. As the characters in the CAPTCHA are highly distorted it is difficult to train a universal network that can perform very accurately while classification. All previous implementation attempts to break CAPTCHA have constrained their development design to one CAPTCHA and have their focus to it. But such a universal Neural Network for all types of characters is first of its kind.

The character set was firstly divided based on its Euler number property and then separate Neural Network was trained on each character set. There are 16 characters with Euler number 0 for which 97% recall accuracy and 70% generalization accuracy was achieved. There are 32 characters with Euler number 1 for which 84% recall accuracy and 65% generalization accuracy was achieved. Also, 2 characters with Euler number -1 were trained and 100% recall accuracy with 98% generalization accuracy was achieved.

6.2 Further Work

The attempt to break a text CAPTCHA is still an active ongoing research in many universities across the globe. Since nowadays all websites have started using reCAPTCHA it is necessary to focus the research on it. Also many anti segmentation techniques like joining all characters in the CAPTCHA have been adopted by developers. Hence, efficient segmentation techniques need to be developed in order to separate the characters. The accuracy of recognition of characters needs to be improved by trying different modifications and type of Artificial Neural Networks. Since CAPTCHA recognition is related to Web CAPTCHAs, hence the system should be integrated with the browser to automatically detect the CAPTCHA image and recognize it.