Acknowledgement

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain. I consider myself privileged to express gratitude and respect towards all those who guided us through the completion of this project.

I convey thanks to my project guide **Prof. Vishal Barot** of Computer Engineering Department for providing encouragement, constant support and guidance which was of a great help to complete this project successfully.

Last but not the least, we wish to thank for our **parents** for financing our studies in this college as well as for constantly encouraging us to learn engineering. This personal sacrifice in providing this opportunity to learn engineering is gratefully acknowledged.

ABSTRACT

In this project we present application of machine learning techniques to the problem of reading text from CAPTCHAs. **CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart)** are distortions of texts or images so that they are still recognizable to most humans but can become difficult for a computer to recognize. Additionally, many CAPTCHAs also inject noise or additional structures such as blobs or lines into the image to further make it difficult to recognize. They are primarily employed as security measures on websites to prevent bots from accessing or performing transactions on the site.

On the other hand, machine learning (ML) algorithms, in particular deep learning (DL) neural networks have been trained with significant success on similar problems such as handwritten digit recognition. This motivates us to build an ML-based CAPTCHA breaker that maps CAPTCHAs to their solutions. We systematically used a CAPTCHA dataset of varying complexity and used different libraries to avoid overfitting on one library. We trained on both fixed and variable length CAPTCHAs and will able to get accuracy level **99% and 90%**, respectively.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Introduction

Online services such as webmail, social media, cloud storage, file sharing, and content creation platforms are often abused by bots. Websites are using CAPTCHAs (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part) as one of their main defence mechanisms against such bots. CAPTCHAs are challenges sent to users and permission is granted only to those that are able to solve them correctly within a certain time frame. The challenges are based on tasks that current state of the art algorithms do not perform well but that are fairly easy for people.

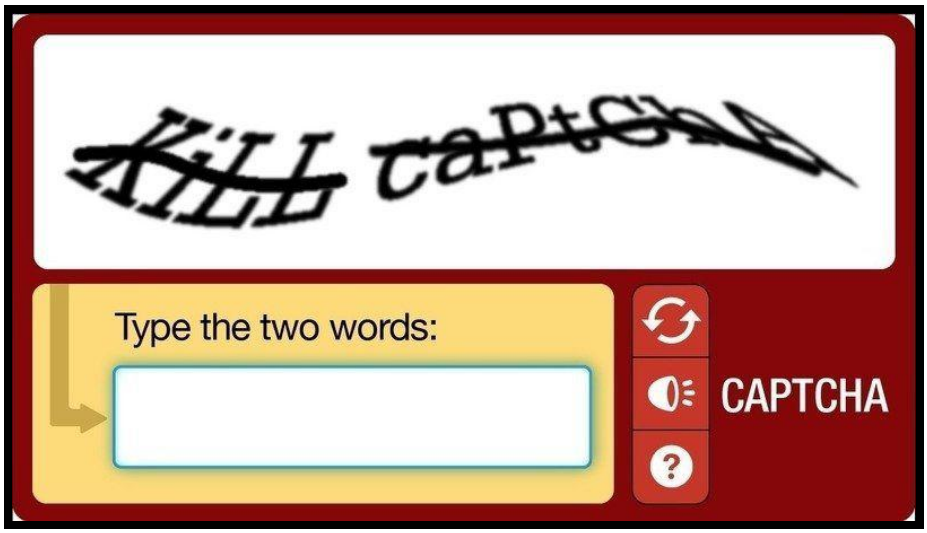

Fig. 1 (a) Example of Text Based CAPTCHA

The term "CAPTCHA" was coined in 2000 by Luis Von Ahn, Manuel Blum, Nicholas J. Hopper (all of Carnegie Mellon University, and John Langford (then of IBM). They are challenge-response tests to ensure that the users are indeed human. The purpose of a CAPTCHA is to block form submissions from spam bots – automated scripts that harvest email addresses from publicly available web forms. A common kind of CAPTCHA used on most websites requires the users to enter the string of characters that appear in a distorted form on the screen.

CAPTCHAs are ubiquitous on the Internet. They are intuitive for users and are a cheap and fast way to secure a site and to ward off spam. A lot of major websites use them for security on the Internet. When CAPTCHAs were designed, there were no AI programs that could recognize them using computer vision. Recently, deep neural networks have brought major advancements in the field of AI and computer vision. They have reached state-of-the-art or better performance as compared to other methods in the fields of speech recognition [1], computer vision, natural language processing [2], and language translation [3]. We tried using deep neural networks to break CAPTCHAs to assess how secure CAPTCHA-based security systems are.

Traditionally, for object/character recognition tasks in computer vision, separate modules were created for preprocessing (noise reduction), segmentation (character segmentation), and character recognition and sequence generation, where the sequence of characters with highest probability was generated.

1.2 Challenges

Challenges include recognition of distorted words, identification of the context of an image, logic questions, mathematical questions and understanding speech. A good candidate task is one such that

- challenges can be automatically generated
- there is a very large (ideally infinite) pool of challenges
- humans (even naive users) perform it easily
- bots perform the task poorly or only with substantial resource overheads

A CAPTCHA is secure if, in the long run, the total cost of automated attacks is higher than their expected gain. Hence the likelihood of a successful attacks is a measure of the security of a

CAPTCHA. It was suggested that if more than 0.01% of the challenges can be successfully solved by a computer program then the scheme is broken, but in the literature a threshold of 1% is more commonly adopted. The threshold was decided based on the cost of the attack and the gains of the hacker for every successful attack. Since the use of CAPTCHAs, as well as the underlying security economics could have changed since these earlier studies were published, it would be useful to have a more recent and representative metric. In the absence of a widely used baseline metric, we will use the much more conservative 1% accuracy criterion that is becoming more widely used in the literature.

1.3 Problem Definition

In this project, we tried to make it easy for a computer to solve a string of character CAPTCHAs. There are so many different types of CAPTCHAs,that we will discuss later,that can be break by bots. So we are also making similar kind of that bot.

1.4 Overview of the Project

CAPTCHA, any user entering a correct solution is presumed to be human. Thus, it is sometimes described as a reverse Turing test, as it is administered by a machine and targeted to a human, in contrast to the standard Turing test that is typically administered by a human and targeted to machine. A common type of CAPTCHA requires the user to type letters or digits from a distorted image that appears on the screen.

A number of research projects have attempted (often with success) to beat visual CAPTCHAs by creating programs that contain pre-processing, segmentation and classification. The pre-processing and classification steps are easy tasks for computers. 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.

Most of the previous work in this area has been done focussing on the CAPTCHA which increase complexity by curvilinear distortion and overlapping of images. Little work has been done on CAPTCHA where complexity arises due to cluttering in the background which connects the text. In such case, with the current software techniques, segmentation becomes almost impossible.

1.5 Project Purpose

Why would anyone need to create a test that can tell humans and computers apart? It's because of people trying to game the system -- they want to exploit weaknesses in the computers running the sites. While these individuals probably make up a minority of all the people on the Internet, their actions can affect millions of users and Web sites. For example, a free email service might find itself bombarded by account requests from an automated program. That automated program could be part of a larger attempt to send out spam mail to millions of people. The CAPTCHA test helps identify which users are real human beings and which ones are computer programs.
Spammers are constantly trying to build algorithms that read the distorted text correctly. So strong CAPTCHAs have to be designed and built so that the efforts of the spammers are thwarted.

1.6 Summary

In our project, we will combine the idea of two papers [4], [5] and are building a model that uses convolutional neural network to learn CAPTCHA image features and then use those features in a

recurrent neural network to output the sequence of characters in the image. We will combine both of the networks to get a single deep neural network that performs end-to-end CAPTCHA recognition.

We have focused mostly on the segmentation part of the algorithms and we will use a generic character recogniser that has been previously tested in the literature [6, 11]. While this is not an optimal state of the art algorithm, it offers the significant advantages of simplicity, robustness, and an ability of training an adequate set of classifiers in a small enough time period to allow us to check different variations of the segmentation algorithm against the validation test.

This project is divided into nine chapters. Chapter 2 gives a literary reviews of CAPTCHAs, including what they are and why we need them and different types of CAPTCHAs. Chapter 3 discusses tools and technology and the work done in the related field of CAPTCHA recognition and what we planned to do. Chapter 4 covers an introduction to neural networks and the various components we used in our project. Chapter 5 introduces How CAPTCHAs are Breaked and make it business. Chapter 6 covers dataset and methods used in our project . We tried different kinds of models, and the details of these are discussed in Chapter 7. Chapter 8 gives issues in CAPTCHAs. Chapter 9 records all the experiments and results we got after training the huge dataset. Chapter 10 concludes the project.

CHAPTER 2
INTRODUCTION TO CAPTCHAs

2.1 Things To Know

The first step to create a CAPTCHA is to look at different ways humans and machines process information. Machines follow sets of instructions. If something falls outside the realm of those instructions, the machines aren't able to compensate. A CAPTCHA designer has to take this into account when creating a test. For example, it's easy to build a program that looks at metadata – the information on the Web that's invisible to humans but machines can read. If you create a visual CAPTCHA and the images' metadata includes the solution, your CAPTCHA will be broken in no time. Similarly, it's unwise to build a CAPTCHA that doesn't distort letters and numbers in some way. An undistorted series of characters isn't very secure. Many computer programs can scan an image and recognize simple shapes like letters and numbers.

One way to create a CAPTCHA is to pre-determine the images and solutions it will use. This approach requires a database that includes all the CAPTCHA solutions, which can compromise the reliability of the test. According to Microsoft Research experts Kumar Chellapilla and Patrice Simard, humans should have an 80 percent success rate at solving any particular CAPTCHA, but machines should only have a 0.01 percent success rate [source: Chellapilla and Simard]. If a spammer managed to find a list of all CAPTCHA solutions, he or she could create an application that bombards the CAPTCHA with every possible answer in a brute-force attack. The database would need more than 10,000 possible CAPTCHAs to meet the qualifications of a good CAPTCHA.

Other CAPTCHA applications create random strings of letters and numbers. You aren't likely to ever get the same series twice. Using randomization eliminates the possibility of a brute-force

attack the odds of a bot entering the correct series of random letters are very low. The longer he string of characters, the less likely a bot will get lucky. CAPTCHAs take different approaches to distorting words. Some stretch and bend letters in weird ways, as if you're looking at the word through melted glass. Others put the word behind a crosshatched pattern of bars to break up the shape of letters. A few use different colours or a field of dots to achieve the same effect. In the end, the goal is the same: to make it really hard for a computer to figure out what's in the CAPTCHA.

Designers can also create puzzles or problems that are easy for humans to solve. Some CAPTCHAs rely on pattern recognition and extrapolation. For example, a CAPTCHA might include series of shapes and ask the user which shape among several choices would logically come next. The problem with this approach is that not all humans are good with these kinds of problems and the success rate for a human user can go below 80%.

2.2 Implementation

Embeddable CAPTCHAs: The easiest implementation of a CAPTCHA to a Website would be to insert a few lines of CAPTCHA code into the Website's HTML code, from an open source CAPTCHA builder, which will provide the authentication services remotely. Most such services are free. Popular among them is the service provided by www.captcha.net 's reCAPTCHA project.

Custom CAPTCHAs: These are less popular because of the extra work needed to create a secure implementation. Anyway, these are popular among researchers who verify existing CAPTCHAs and suggest alternative implementations. There are advantages in building custom CAPTCHAs:

1. A custom CAPTCHA can fit exactly into the design and theme of yoursite. It will not look like some alien element that does not belong there.
2. We want to take away the perception of a CAPTCHA as an annoyance, and make it convenient for the user.
3. Because a custom CAPTCHA, unlike the major CAPTCHA mechanisms, obscure you as a target for spammers. Spammers have little interest in cracking a niche implementation.
4. Because we want to learn how they work, so it is best to build one ourselves.

2.3 CAPTCHA Logic

The CAPTCHA image (or question) is generated. There are different ways to do this. The classic approach is to generate some random text, apply some random effects to it and convert it into an image.

1. The CAPTCHA image (or question) is generated. There are different ways to do this. The classic approach is to generate some random text, apply some random effects to it and convert it into an image.
2. Step 2 is not really sequential. During step 1, the original text (pre-altered) is persisted somewhere, as this is the correct answer to the question. There are different ways to persist the answer, as a server-side session variable, cookie, file, or database entry.
3. The generated CAPTCHA is presented to the user, who is prompted to answer.
4. The back-end script checks the answer supplied by the user by comparing it with the persisted (correct) answer. If the value is empty or incorrect, we go back to step 1: a new CAPTCHA is generated. Users should never get a second shot at answering the same CAPTCHA.
5. If the answer supplied by the user is correct, the form post is successful and processing can continue. If applicable, the generated CAPTCHA image is deleted.

2.4 Characteristics of CAPTCHA

The important characteristics of a CAPTCHA are:

- Easy for a user to solve.
- Difficult for a program or a computer to solve.

To make it happen, the following standard techniques are used. Normally, characters are used to generate CAPTCHAs. The reason why object-based CAPTCHAs are not used in CAPTCHAs is because recognition of objects requires prior knowledge of the scene, and different regions have different names for objects; therefore, it would not be universal. Instead, characters have a unique data set (for example, in the English alphabet, 26 characters and 10 numeric digits), which is present on keyboards as well, making them easier to understand. Hence, they are easy to generate. Figure 2(a) shows an image-based CAPTCHA in which we used the alphabet.

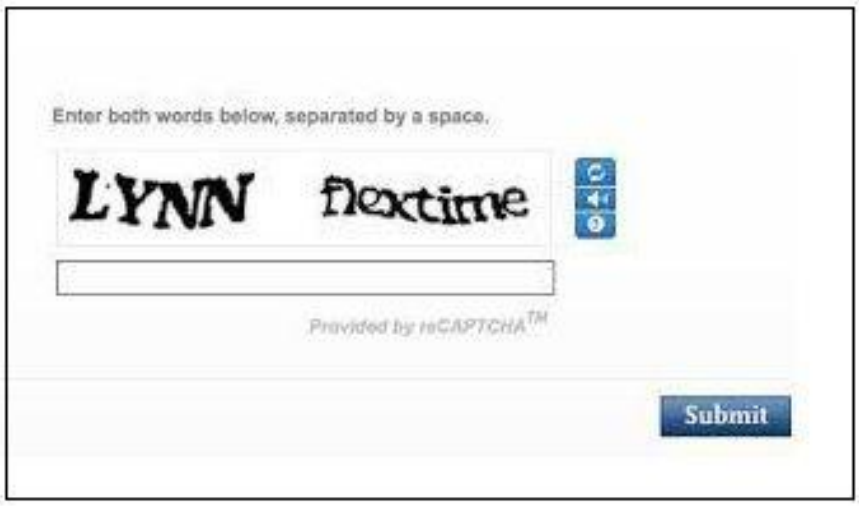

Fig. 2(a) A Sample Image-based CAPTCHA

2.5 Different Types of CAPTCHAs

There are major 9 types of CAPTCHAs exist till now.We will discuss each of them one by one.

2.5.1 The standard word captcha with an audio option

This is the standard captcha available whenever a security check-in is required, where you need to write the word which has been displayed. But some of the distorted word images are hard to solve. To get this pass through it allows you to use the option of**"Recaptcha"**, in order to receive a new one. There is also an audio option if you are unable to visually make out the word. These are the most commonly used while preparing a form in website development or app development.

Fig. 2(b) Standard CAPTCHA

2.5.2 Math Solution

This type of captcha involves basic math problems and if your user cannot solve this basic questions then probably you do not want them to visit your website further. This provides with easy to read numbers and must be solved to get through the captcha.

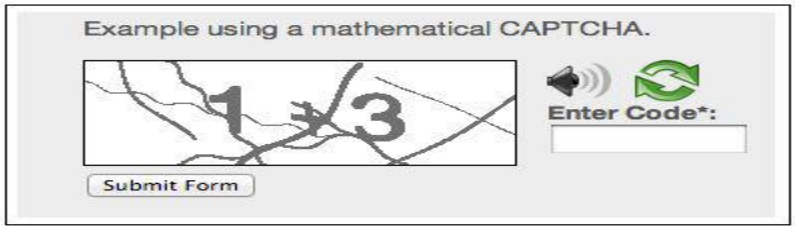

Fig. 2 (c) Math Solution CAPTCHA

2.5.3 Picture identification of captcha

This captcha provides users for selecting the elementary choice of selecting the correct image that they are asked to identify. This type of captcha usually never gets harder than the basic images, so you do not have to worry about your users not being able to depict them.

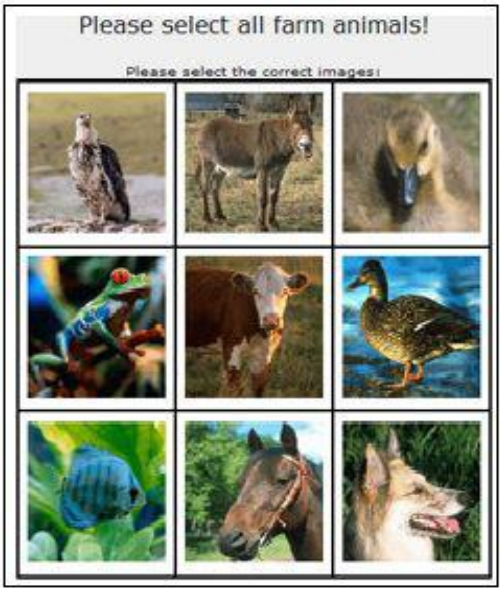

Fig. 2(d) Picture Identification CAPTCHA

2.5.4 3D captcha

These type of captchas are called as **"Super Captcha"** because there are several 3D images which include both images and words and thus becomes hard for one to solve it.

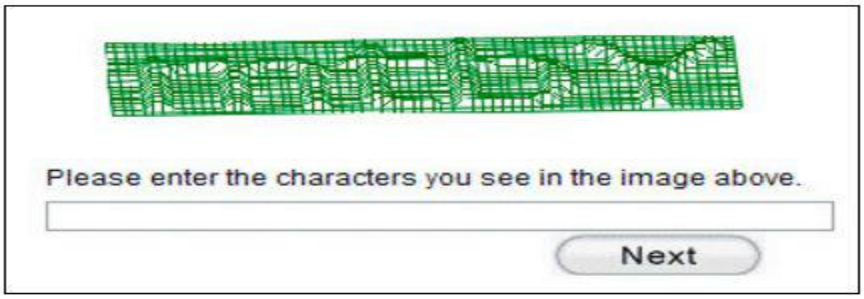

Fig. 2 (e) 3D CAPTCHA

2.5.5 Ad-injected captcha

These type of captcha helps your websites to earn some extra cash by publishing it, which in turn also helps in terms of brand recognition.

Fig. 2(f) Ad-injected CAPTCHA

2.5.6 jQuery slider Captcha

This is a plugin which gives you the ability to add captcha to your forms which are easy to use. This plugin is very useful to keep the spammers away. This plugin lock is disabled until a person slides it to enable it.

Fig. 2(g) Slider CAPTCHA

2.5.7 Drag and drop Captcha

This is also one of the easy to use captchas. It is jQuery based which allows user to drag the required object or shape to pass through the security gate.

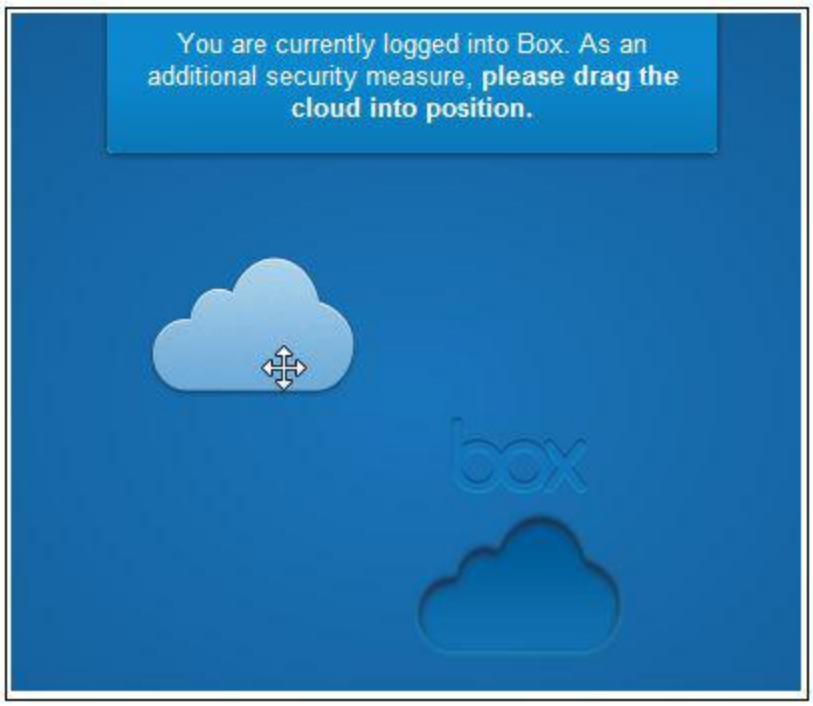

Fig.2(h) Drag And Drop CAPTCHA

2.5.8 Tic Tac Toe Captcha

This captcha which involves gamification was designed for fun and an easy way to ensure that only humans can interact with your website. The captcha that does not hurt that much.

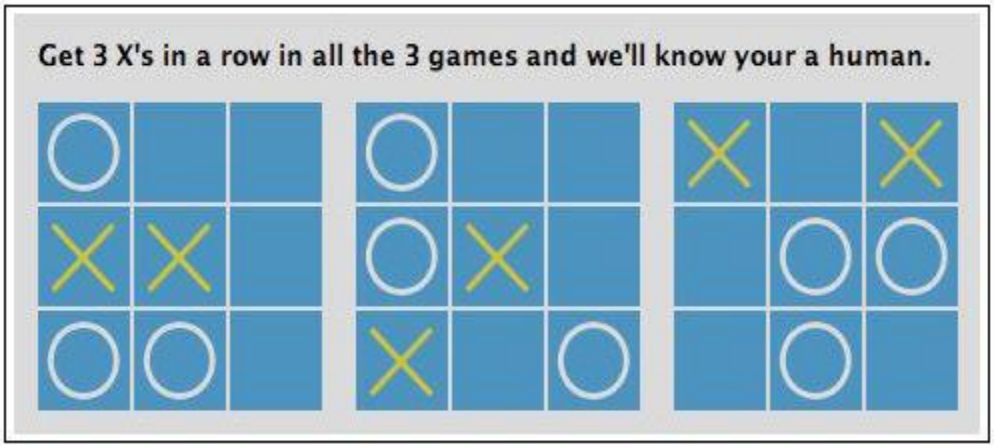

Fig.2(i) Game CAPTCHA

2.5.9 reCAPTCHA

reCAPTCHA is a free service that protects your website from spam and abuse. reCAPTCHA uses an advanced risk analysis engine and adaptive CAPTCHAs to keep automated software from engaging in abusive activities on your site. It does this while letting your valid users pass through with ease.

reCAPTCHA offers more than just spam protection. Every time our CAPTCHAs are solved, that human effort helps digitize text, annotate images, and build machine learning datasets. This in turn helps preserve books, improve maps, and solve hard AI problems.

Fig. 2(j) reCAPTCHA

2.6 Applications of CAPTCHA

CAPTCHAs are used in various Web applications to identify human users and to restrict access to them.Some of them are:

1. Online Polls:

As mentioned before, bots can wreak havoc to any unprotected online poll. They might create a large number of votes which would then falsely represent the poll winner in spotlight. This also results in decreased faith in these polls. CAPTCHAs can be used in websites that have embedded polls to protect them from being accessed by bots, and hence bring up the reliability of the polls.

2. Protecting Web Registration:

Several companies offer free email and other services. Until recently, these service providers suffered from a serious problem – bots. These bots would take advantage of the service and would sign up for a large number of accounts. This often created problems in account management and also increased the burden on their servers. CAPTCHAs can effectively be used to filter out the bots and ensure that only human users are allowed to create accounts.

3. Preventing comment spam:

Most bloggers are familiar with programs that submit large number of automated posts that are done with the intention of increasing the search engine ranks of that site. CAPTCHAs can be used before a post issubmitted to ensure that only human users can create posts. A CAPTCHA won't stop someone who is determined to post a rude message or harass an administrator, but it will help prevent bots from posting messages automatically.

4. Search engine bots:

It is sometimes desirable to keep web pages 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 website, CAPTCHAs are needed.

5. E-Ticketing:

Ticket brokers like TicketMaster also use CAPTCHA applications.These applications help prevent ticket scalpers from bombarding the service with massive ticket purchases for big events. Without some sort of filter, it's possible for a scalper to use a bot to place hundreds or thousands of ticket orders in a matter of seconds. Legitimate customers become victims as events sell out minutes after tickets become available. Scalpers then try to sell the tickets above face value. While CAPTCHA applications don't prevent scalping; they do make it more difficult to scalp tickets on a large scale.

6. Email spam:

CAPTCHAs also present a plausible solution to the problem of spam emails. All we have to do is to use a CAPTCHA challenge to verify that a indeed a human has sent the email.

7. 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.

CHAPTER 3
TOOLS AND TECHNOLOGY REVIEW

3.1 About Tools and Technology

3.1.1 Python 3

Python is a fun programming language with great libraries for machine learning and computer vision.

3.1.2 OpenCV

OpenCV is a popular framework for computer vision and image processing. We'll use OpenCV to process the CAPTCHA images. It has a Python API so we can use it directly from Python.

3.1.3 Keras

Keras is a deep learning framework written in Python. It makes it easy to define, train and use deep neural networks with minimal coding.

3.1.4 TensorFlow

TensorFlow is Google's library for machine learning. We'll be coding in Keras, but Keras doesn't actually implement the neural network logic itself. Instead, it uses Google's TensorFlow library behind the scenes to do the heavy lifting.

3.1.5 OCR (Optical Character Recognition)

Optical Character Recognition, or OCR, is the recognition of printed or written characters by a computer. It enables you to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data. Popular open source OCR tools are Tesseract, GOCR and Ocrad. We will use Tesseract for this project.

3.1.6 Tesseract

Tesseract is an open source OCR engine for various operating systems. It's considered one of the most accurate OCR engines currently available, with the precision depending on the clearness of the image. Google has sponsored its development since 2006.

3.1.6.1 PyTesseract

Python-Tesseract is a python wrapper that helps you use Tesseract-OCR engine to convert images to the accepted format from Python. It can read all image types – png, jpeg, gif, tiff, bmp, etc.

3.2 Related Work

Programmatically, breaking CAPTCHAs is not a new concept. For example, Mori and Malik [7] have broken EZ-Gimpy (92% success) and Gimpy (33% success), CAPTCHAs with sophisticated object recognition algorithms. In comparison to earlier works that were based on sophisticated computer vision algorithms, we are planning to train an end-to-end neural network system that would extract the features needed for classification with minimal hand tuning. Neural networks have shown great results recently in many domains, such as natural language processing [2], speech [1], and image processing [4][5]. Neural networks also have brought down the entry barrier in training such models, as one does not require deep domain knowledge

to massage inputs in order to provide the features that a model could learn from. The hidden layers in neural networks extract the features that are useful during learning. We will be discussing more about neural networks later in the Chapter 5.

In some of the papers, for example, in paper [8], the following steps were used for character recognition in a CAPTCHA:

1. Preprocessing
2. Segmentation
3. Training the model for individual character recognition
4. Generating sequence with highest probability

These steps are very difficult to do because of the following reasons:

- Segmentation is difficult, as some digits could overlap with other digits.
- Deformity of digits is also a major concern. For example, a digit “2” can have a larger loop or just a cusp.
- Unknown scale of characters. We do not know how big a character will be, so it is not known how big the segmentation boxes should be.
- Character orientation. Characters could be rotated at arbitrary angles, making recognition difficult.

Modules for each of the steps mentioned above are optimized independently, so systems combining these modules don’t work well in practice. We can instead learn a deep neural network, a single monolithic system for embedding these modules, and train the entire network together to make sure that that objectives of all the modules are aligned. We can just provide

CAPTCHA images to the network and let it learn image features and how to use these features for recognition.

Relevantly, Google has published a research paper in which they used a convolutional neural network [4] for detecting home addresses using convolutional neural networks. They achieve a 96% accuracy in recognizing complete street numbers. We took the same idea for this project to train our dataset with convolutional neural networks. But in their work [4], they fixed the length of the street number in an image unlike CAPTCHAs, in which length could be different. To tackle this problem, we propose to use Recurrent Neural Networks, which have achieved good results recently, as shown by the "show and tell" [5] paper by Google, where they generate a caption (variable length) for a given image. The real work in decoding a CAPTCHA is to guess all the words in the CAPTCHA correctly. If even one word is wrong, we have to discard the result. Based on the papers [4] and [5], a neural network could be helpful in these scenarios. Using these ideas, we will use CNNs to learn image features and the RNNs to predict a sequence of characters (which could also be variable) in a CAPTCHA.

CHAPTER 4
INTRODUCTION TO NEURAL NETWORK

4.1 What Is NEURAL NETWORK?

In our project, we will use neural networks to break CAPTCHAs. Neural networks are inspired by the brain. In 1943, McCulloch and Pitts [9] laid the formal foundation for the field of artificial neural networks. Because neural networks were computationally expensive and there was no good learning algorithm for training the models, they became unpopular in the 1960's. Marvin Minsky and Seymour Papert made this clear in their paper [12]. Another key advance that came later was the backpropagation algorithm, which effectively solved the neural network problem, given by Werbos [13]. In the early 1980's, researchers showed renewed interest in neural networks. From 2005 onwards, they have again become popular, as computers have now become fast enough to do large computations. People have also achieved success in training neural networks with the SGD (stochastic gradient descent) algorithm, which fortunately works, but does not have clear theoretical justification. In addition, neural network models are big, and thus require a lot of training data. With the recent advances in Big Data, it has become very easy to collect training data. They are the hottest area in the field of machine learning [10].

4.2 Difference Between Machine Learning And Neural Network

Neural networks work very well with different machine learning problems. Neural networks are a type of machine learning algorithm. The basic difference between machine learning and conventional programming languages is that in conventional programming, a computer has to be explicitly programmed. We ourselves have to write and maintain the code. But in case of neural networks, the network adapts itself to the problem during training. In conventional programming style, people have to understand the problem well and research different approaches. Since a clear solution is often elusive in practical problems,people tend to use heuristics, which work for some use cases but do not generalize well. But with machine learning, we have the model itself learn from the data. Machines tend to learn faster (vis a vis core research), so they take less time

in solving problems. Machine learning algorithms like neural networks also do a good job with generalization. Some of the applications of neural networks include facial recognition as used by Facebook to tag photos [14], image captioning [5] by Google and etc.

Neural networks are essentially a bunch of interconnected elements called neurons. They are an information processing paradigm which is inspired by biological nervous systems. The nervous system contains around 1010 neurons. Figure 4(a) shows a single neuron.

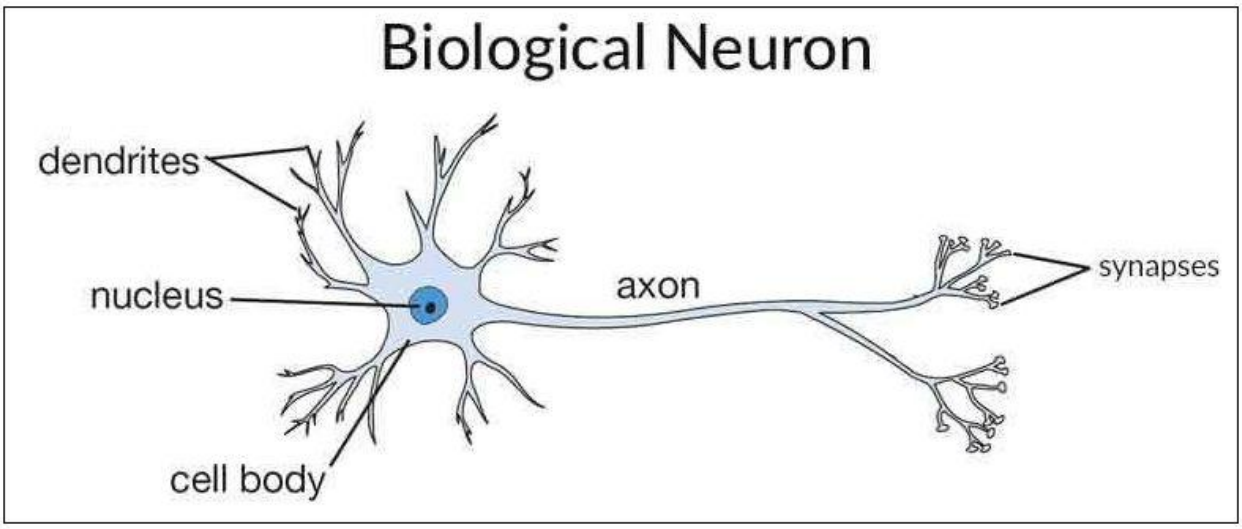

Fig. 4(a) A Biological Neuron

Each biological neuron consists of a cell body. It contains lots of dendrites, which bring electrical

signals into the cell, and an axon, which transmits these signals out of the cell. A neuron fires when the collective effect of its inputs reaches a certain threshold. The axons of neurons are dependent on each other and can influence the dendrites of another neuron. Similarly, a neural network starts with a model, as in Figure 4(b). It consists of several inputs and a single output. Every input is modified by a weight and multiplied with the input value. The neuron combines these weighted inputs and, with the help of the threshold value and activation function,

determines its output. **The objective of the neural network is to transform the inputs into a meaningful output.**

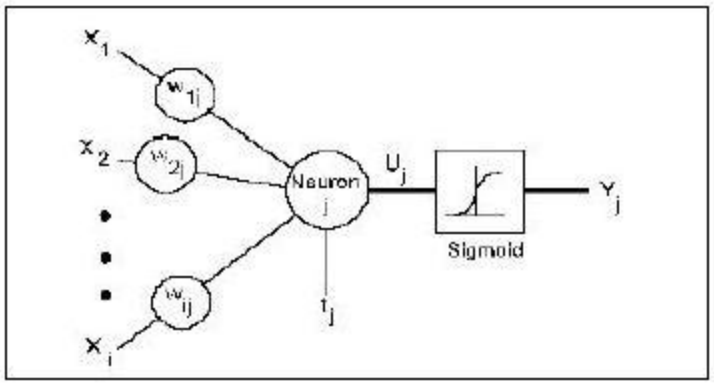

Fig. 4(b) A Simple Neural Network Model

CHAPTER 5
BREAKING CAPTCHAs

5.1 Overview

The challenge in breaking a CAPTCHA isn't figuring out what a message says -- after all, humans should have at least an 80 percent success rate. The really hard task is teaching a computer how to process information in a way similar to how humans think. In many cases, people who break CAPTCHAs concentrate not on making computers smarter, but reducing the complexity of the problem posed by the CAPTCHA.

Let's assume you've protected an online form using a CAPTCHA that displays English words. The application warps the font slightly, stretching and bending the letters in unpredictable ways. In addition, the CAPTCHA includes a randomly generated background behind the word. A programmer wishing to break this CAPTCHA could approach the problem in phases. He or she would need to write an algorithm -- a set of instructions that directs a machine to follow a certain series of steps. In this scenario, one step might be to convert the image in grayscale. That means the application removes all the color from the image, taking away one of the levels of obfuscation the CAPTCHA employs.

Next, the algorithm might tell the computer to detect patterns in the black and white image. The program compares each pattern to a normal letter, looking for matches. If the program can only match a few of the letters, it might cross reference those letters with a database of English words. Then it would plug in likely candidates into the submit field. This approach can be surprisingly effective. It might not work 100 percent of the time, but it can work often enough to be worthwhile to spammers.

What about more complex CAPTCHAs? The Gimpy CAPTCHA displays 10 English words with warped fonts across an irregular background. The CAPTCHA arranges the words in pairs and the words of each pair overlap one another. Users have to type in three correct words in order to move forward. How reliable is this approach?As it turns out, with the right CAPTCHA-cracking algorithm, it's not terribly reliable. Greg Mori and Jitendra Malik published a paper detailing their approach to cracking the Gimpy version of CAPTCHA.

Mori and Malik ran a series of tests using their algorithm. They found that their algorithm could correctly identify the words in a Gimpy CAPTCHA 33 percent of the time [source: Mori and Malik]. While that's far from perfect, it's also significant. Spammers can afford to have only one-third of their attempts succeed if they set bots to break CAPTCHAs several hundred times every minute.

Another vulnerability that most CAPTCHA scripts have is again in their use of sessions; if we're on an insecure shared server, any user on that server may have access to everyone else's session files, so even if our site is totally secure, a vulnerability on any other website hosted on that machine can lead to a compromise of the session data, and hence, the CAPTCHA script. One workaround is by storing only a hash of the CAPTCHA word in the session, thus even if someone can read the session files, they can't find out what the CAPTCHA word is.

5.2 Breaking CAPTCHAs without OCR

Most CAPTCHAs don't destroy the session when the correct phrase is entered. So by reusing the session id of a known CAPTCHA image, it is possible to automate requests to a CAPTCHA-protected page.

Manual steps

1. Connect to CAPTCHA page

2. Record session ID and CAPTCHA plaintext

Automated steps

Resend session ID and CAPTCHA plaintext any number of times, changing the user data. The other user data can change on each request. We can then automate hundreds, if not thousands of requests, until the session expires, at which point we just repeat the manual steps and then reconnect with a new session ID and CAPTCHA text.

Traditional CAPTCHA-breaking software involves using image recognition routines to decode CAPTCHA images. This approach bypasses the need to do any of that, making it easy to hack CAPTCHA images.

5.3 Breaking a visual CAPTCHA

Greg Mori and Jitendra Malik of University of California at Berkeley's Computer Vision Group evaluate image based CAPTCHAs for reliability. They test whether the CAPTCHA can withstand bots who masquerade as humans.

Approach: The fundamental ideas behind our approach to solving Gimpy are the same as those we are using to solve generic object recognition problems. Our solution to the Gimpy CAPTCHA is just an application of a general framework that we have used to compare images of everyday objects and even find and track people in video sequences. The essences of these problems are similar. Finding the letters "T", "A", "M", "E" in an image and connecting them to read the word "TAME" is akin to finding hands, feet, elbows, and faces and connecting them up to find a human. Real images of people and objects contain large amounts of clutter. Learning to deal with the adversarial clutter present in Gimpy has helped us in understanding generic object recognition problems.

Breaking an EZ-Gimpy CAPTCHA: Our algorithm for breaking EZ-Gimpy consists of 3 main steps:

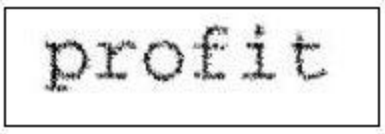

Fig. 5(a) Breaking CAPTCHA

1. **Locate possible (candidate) letters at various locations**: The first step is to hypothesize a set of candidate letters in the image. This is done using our shape matching techniques The method essentially looks at a bunch of points in the image at random, and compares these points to points on each of the 26 letters. The comparison is done in a way that is very robust to background clutter and deformation of the letters. The process usually results in 3-5 candidate letters per actual letter in the image. In the example shown in Fig 5.1, the "p" of profit matches well to both an "o" or a "p", the border between the "p" and the "r" look a bit like a "u", and so forth. At this stage we keep many candidates, to be sure we don't miss anything for later steps.

2. **Construct graph of consistent letters**: Next, we analyze pairs of letters to see whether or not they are "consistent", or can be used consecutively to form a word.

3. **Look for plausible words in the graph:** There are many possible paths through the graph of letters constructed in the previous step. However, most of them do not form real words. We select out the real words in the graph, and assign scores to them based on how well their individual letters match the image. Similar algorithms are also devised by Mori and Malik to evaluate other image based CAPTCHAs like Gimpy, etc.

5.4 Breaking an audio CAPTCHA

Recent research is suggesting that Google's audio capture is the latest in a string of CAPTCHAs to have been defeated by software. t has been theorized that one cost-effective means of breaking audio captures and image captures that have not yet had automated systems developed is to use a mechanical turk and pay low rates for per-CAPTCHA reading by humans, or provide another form of motivation such as access to popular sites for reading the CAPTCHA. However, it always required a significant level of resources to achieve.

The development of software to automatically interpret CAPTCHAs brings up a number of problems for site operators. The problem, as discovered by WintercoreLabs and published at the start of March is that there are repeatable patterns evident in the audio file and by applying a set of complex but straight forward processes, a library can be built of the basic signal for each possible character that can appear in the CAPTCHA. Wintercore point to other audio CAPTCHAs that could be easily reversed using this technique, including the one for Facebook.

The wider impact of this work might take some time to appear, but it provides an interesting proof of breaking audio CAPTCHAs. At the least, it shows that both of Google's CAPTCHA tools have now been defeated by software and it should only be a matter of time until the same can be said for Microsoft and Yahoo!'s offerings. Even with an effectiveness of only 90%, any failed CAPTCHA can easily be reloaded for a second try.

5.5 Social Engineering used to break CAPTCHAs:

Spammers often use social engineering to outwit gullible Web users to serve their purpose. Security firm, Trend Micro warns of a Trojan called TROJ_CAPTCHAR, which masquerades as a strip tease game. At each stage of the game, the user is asked to solve a CAPTCHA. The result is relayed to a remote server where a malicious user is waiting for them. The strip-tease game is a ploy by spammers to identify and match solutions for ambiguous CAPTCHAs from legitimate sites, using the unsuspecting user as the decoder of the said images.

5.6 CAPTCHA Cracking as a Business:

No CAPTCHA can survive a human that’s receiving financial incentives for solving it. CAPTCHA are cracked by firms posing as Data Processing firms. They usually charge $2 for 1000 CAPTCHAs successfully solved. They advertise their business as “Using the advertisement in blogs, social networks, etc significantly increases the efficiency of the business. Many services use pictures called CAPTCHAs in order to prevent automated use of these services. Solve

CAPTCHAs with the help of this portal; increase your business efficiency now!" Such firms help spammers in beating the first line of defence for a Website, i.e., CAPTCHAs.

CHAPTER 6

DATASET AND METHODS

6.1 Creating Our Dataset

To train any machine learning system, we need training data. To break a CAPTCHA system, we want training data that looks like this:

Fig.6(a) CAPTCHA Image with Output

We will use PyCaptcha, a python package for CAPTCHA generation, to make custom CAPTCHA image dataset. This package offers several degrees of freedom such as font style, distortion and noise, which we can exploit to increase the diversity of our data and the difficulty of the recognition task.

For the first step, we will create single-letter CAPTCHA images by feeding PyCaptcha uppercase letters ranging from A to Z from a restricted set of fonts. The resulting images will be labelled by the corresponding letters. This will give us a supervised classification problem with 26 classes.

For an actual CAPTCHA breaker, we need to map an image into a string of letters. Therefore, we will also generate a four-letter CAPTCHA image dataset. However, we cannot take each distinct

four-letter string as a label, as that will give too many classes. We will store the full four-letter string, but as is discussed in a later section, we only used 26 single-letter labels in our multi-letter CAPTCHA algorithms.

A typical four-letter CAPTCHA image that we have generated is shown in Fig.6(b)

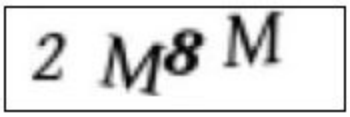

Fig. 6(b) CAPTCHA Image

Finally, we will generate a set of 1580 single-letter and four-letter CAPTCHAs that came from a different library of fonts which further measures how well our algorithms have truly learned the fundamental archetype of each letter based on a restricted set of training data. This is referred to as the "prediction" dataset.

6.2 Methods

6.2.1 Single-letter CAPTCHA recognition

6.2.1.1 Using PyTesseract Library

In this we have used PyTesseract Library, that will automatically recognize text from image. So we have used different types of simple images in which letters and digits are recognized.

6.2.1.2 k-means clustering

We will implement the k-means clustering algorithm using the k-means classifier in the scikit-learn library. The input data were grouped into 26 clusters. To determine the prediction for each cluster, we will pick up the most common label in each group and used it as the prediction for the whole cluster. The k-means result gives a rough indication of how inherently 'learn-able' the dataset is.

6.2.1.3 Support vector machine (SVM)

We built a support vector machine with the LinearSVC classifier in the scikit-learn library. Multi-class classification was done by "one-vs-the rest" strategy. L2 regularization and hinge loss function were used.

6.2.1.4 Convolutional neural network (CNN)

We will use TensorFlow back-end and Keras as a front-end to train our CNN, as these packages are well-designed and easy to get started for a deep learning project. We will design our own network from scratch using these packages, trying out several but ultimately settling on the architecture as shown in Fig.6(c). In our design, we will consider factors such as the number of parameters, layers, activations, filter sizes, etc.

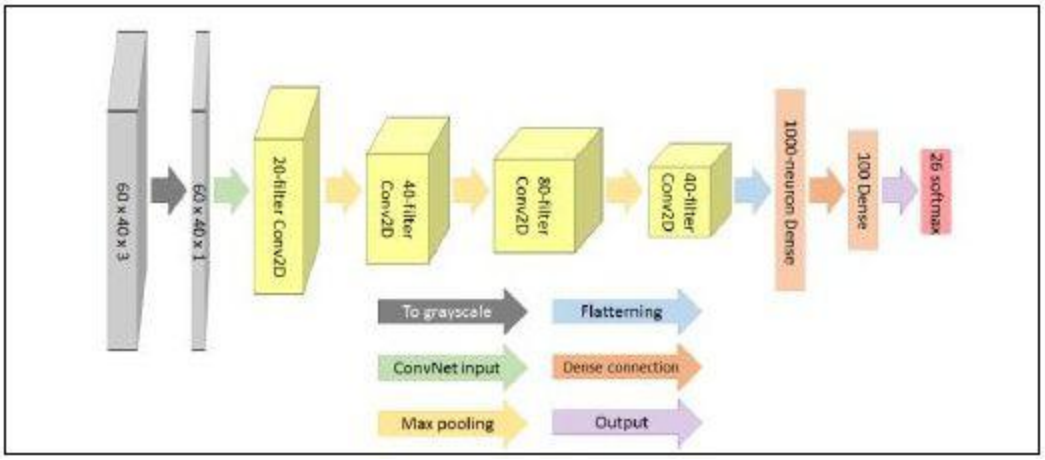

Fig.6(c) Structure of Our Convolutional Neural Network

6.2.2 Multi-CAPTCHA recognition algorithms

6.2.2.1 Moving-window algorithm

We will develop two algorithms to recognize a four-letter CAPTCHA image of 160-by-60 pixels. The first one, referred to as the "moving-window algorithm", directly uses the well-trained single-letter CNN reported above. A subset of the full image defined by a 40-by-60 "window" was revealed to the CNN. This window will translate across the full image, with a prediction made at each position. It is expected that when the window moves across a specific

letter, the softmax probability of that letter will first increase, then plateau at a high confidence when the window is well-centered, and finally decrease. Thus, from the dependence of prediction probability on window position, with some threshold parameters being manually set, we were able to extract the full four-letter string.

6.2.2.2 Multi-CNN algorithm

The second algorithm, referred to as the "multi-CNN algorithm" uses four fixed and overlapping windows to crop out four 70-by-60 sub-images from the full image of 160-by-60. Using our four-letter CAPTCHA dataset, for each of the four sub-image crops, we will train one CNN with the labels being one of the four digits in the strings. To be specific, the first crop was from pixel 0 to 69 in length, and labelled by the first digit; the second crop was from pixel 30 to 99, labelled by the second digit; the third from 60 to 129, labelled by the third digit; and the fourth from 90 to 159, labelled by the last digit. All four crops used the full width. Each of the four trained CNNs will use to predict one digit in the test images.

CHAPTER 7

EXPERIMENT OF MODELS

Before processing on image we require some pre-processing.

7.1 Pre-processing

The purpose of pre-processing is to take the challenges to a state were the segmentation algorithms can extract the letters. Common pre-processing techniques include:

- **Background removal**: This is most useful against CAPTCHAs that use colour as a defence mechanism. It results in a grayscale image with the foreground pixels retaining their intensity and background pixels being white.
- **Up-sampling** Each pixel of the image is divided into sub pixels, thus allowing finer control over the area that is affected by the segmentation algorithm.
- **Blurring**: The image is convolved with Gaussian, mean or median smoothing kernels to reduce the amount of noise in the image.
- **Thresholding**: Removes pixels of low intensity (they are treated as noise), resulting in a binarised image. This is very useful since most segmentation algorithms require a binarised input image.
- **Line removal**: Eliminate straight segments that are not part of characters. Lines are categorised as:

  - lines with **smaller thickness** than the characters. They can be removed using erosions followed by dilations [15]. Black pixels in areas with small black count are removed from the erosion and dilation preserves the thickness of the characters in areas were pixels were not removed.

  - lines with **similar or greater thickness** than the characters. If the lines are longer than the characters then line detection algorithms (*eg.*Hough transforms) are used to identify them. The area around them is then examined to decide which pixels to remove [6].

If the lines substantially overlap with characters, then they are not removed at this stage and instead are dealt with by the segmentation methods.

- **Thinning** of characters: a binary skeleton of the image is created using Zhang's thinning algorithm [16]. This is done in order to reduce the number of character pixels without removing any intrinsic information.

7.2 Single Character Recognition

In this model we will input the CAPTCHA image to PyTesseract Library and we will apply pre-processing to image.The model looks like this:

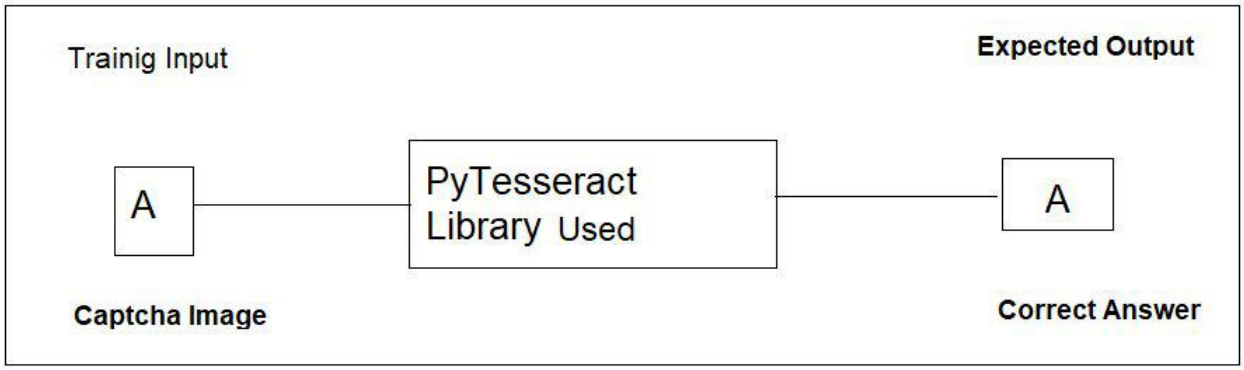

Fig. 7 (a) Single Character Recognition

When we applied pre-processing, the program generates basically two more images of the original images.These two images are "**removed noise image**" and "**threshold image**" respectively.These two images are illustrated below:

Fig. 7 (b) Original Image

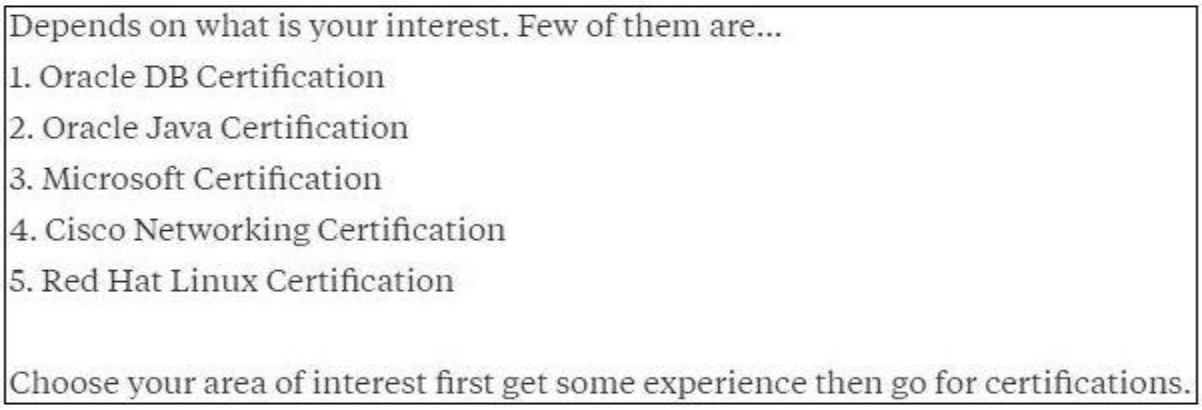

Fig. 7 (c) Removed_Noise Image

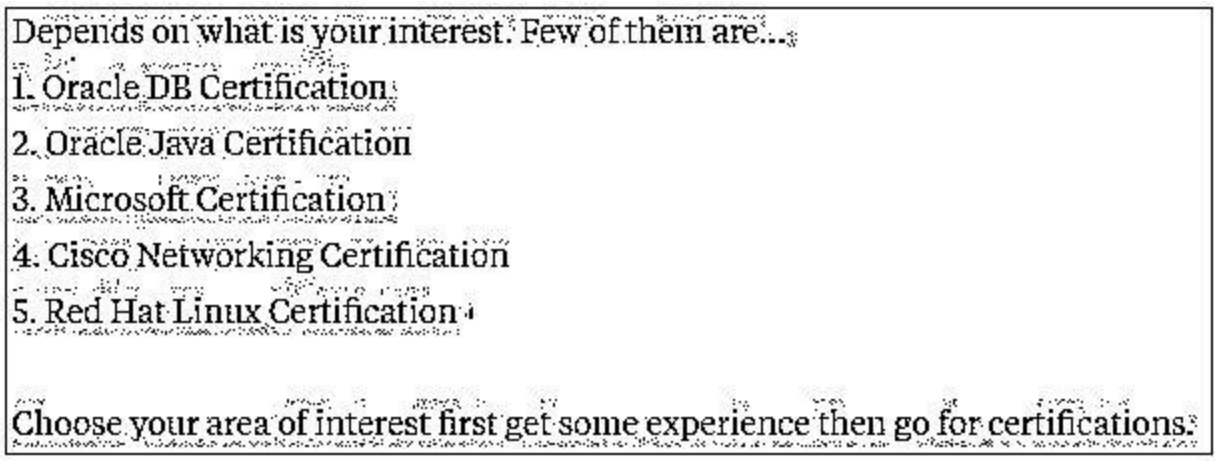

Fig. 7 (d) Threshold Image

So from this we can get all single letters and digits after pre-processing of image.Here we have not used any such algorithm but just used PyTesseract Library Functions which are in built.

7.3 k-means Clustering

Here, we will use k-means clustering algorithm which is machine learning algorithm.In this algorithm the process flow is illustrated in the figure. For single character recognition we can't use this algorithm. So it's accuracy is little bit more than single character recognition algorithm. We can also use Support Vector Machine algorithm instead of k-means clustering.

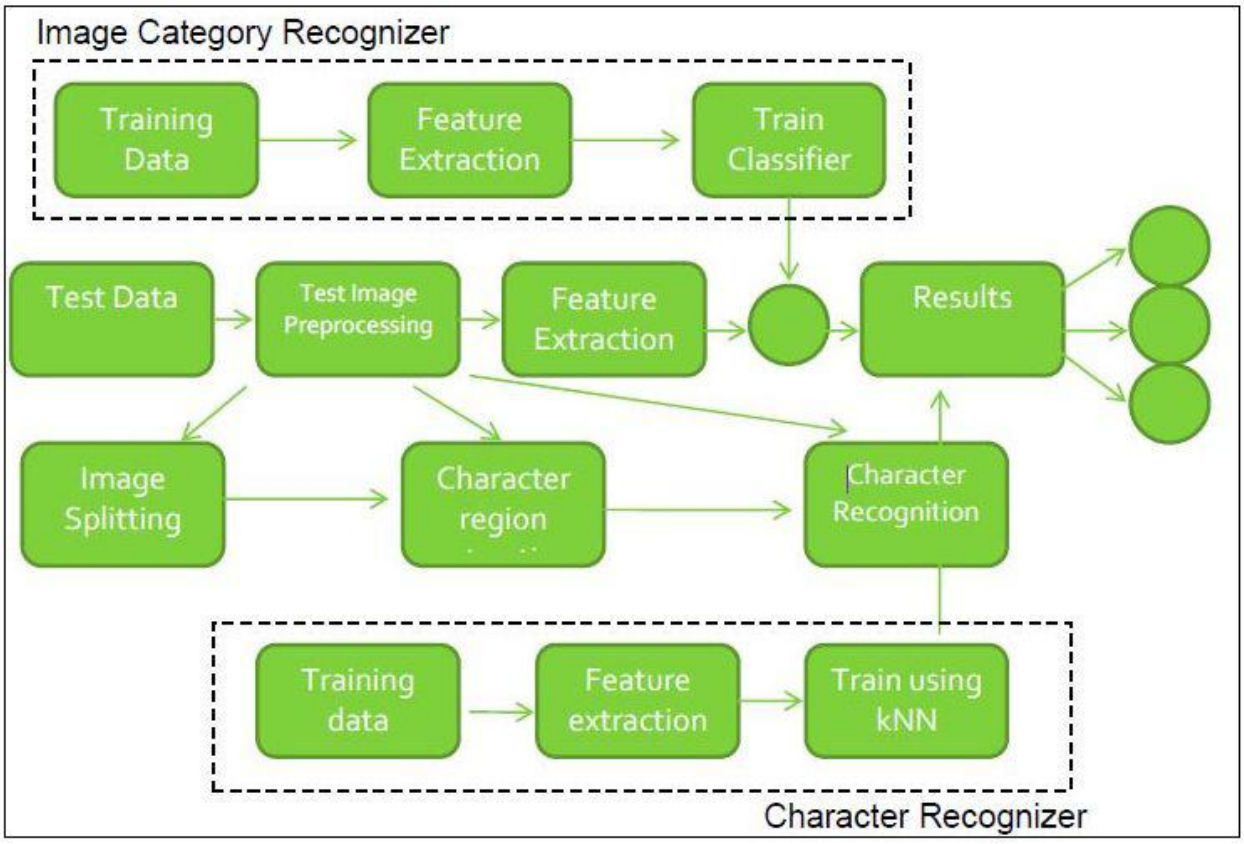

Fig. 7 (e) Process Flow Of The k-Nearest Neighbour

7.4 Convolutional Neural Network

7.4.1 Building and Training the Neural Network

Since we only need to recognize images of single letters and numbers, we don't need a very complex neural network architecture. Recognizing letters is a much easier problem than recognizing complex images like pictures like cats and dogs.

We'll use a simple convolutional neural network architecture with two convolutional layers and two fully-connected layers:

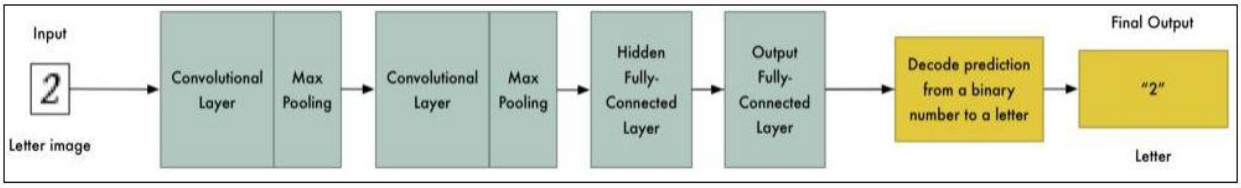

Fig. 7 (f) Internal Working of CNN

7.2.2 Simplifying the Problem

Now that we have our training data, we could use it directly to train a neural network:

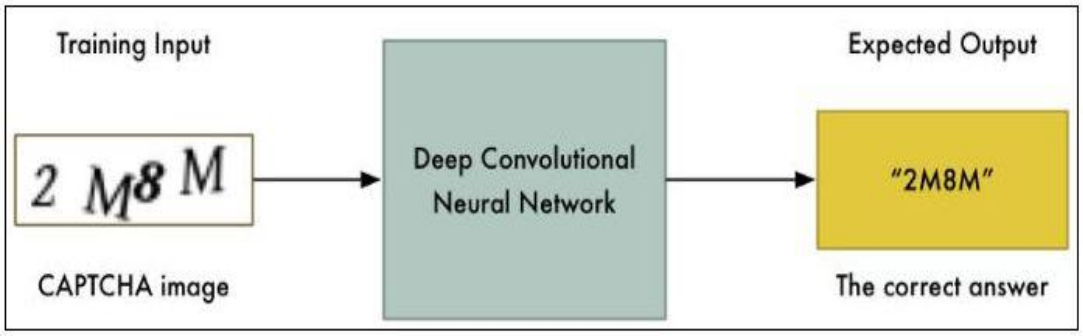

Fig. 7(g) Working of CNN

With enough training data, this approach might even work — but we can make the problem a lot simpler to solve. The simpler the problem, the less training data and the less computational power we'll need to solve it.

Luckily the CAPTCHA images are always made up of only four letters. If we can somehow split the image apart so that that each letter is a separate image, then we only have to train the neural network to recognize a single letter at a time:

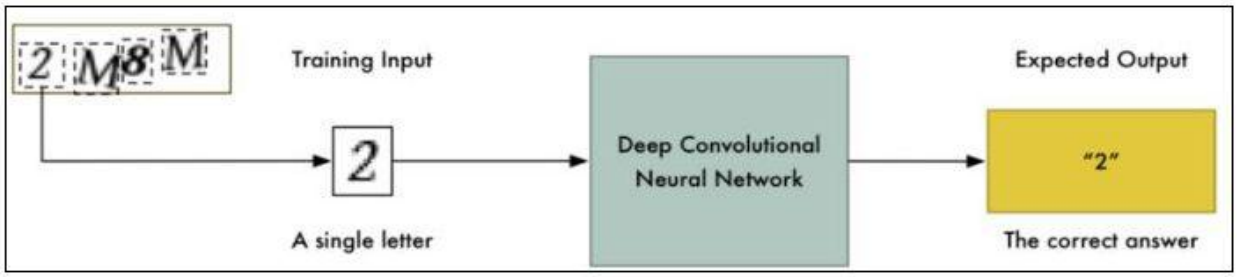

Fig. 7 (h) Single Character Identification using CNN

Luckily, we can still automate this. In image processing, we often need to detect "blobs" of pixels that have the same color. The boundaries around those continuous pixels blobs are called contours. OpenCV has a built-in findContours() function that we can use to detect these continuous regions.

So we'll start with a raw CAPTCHA image:

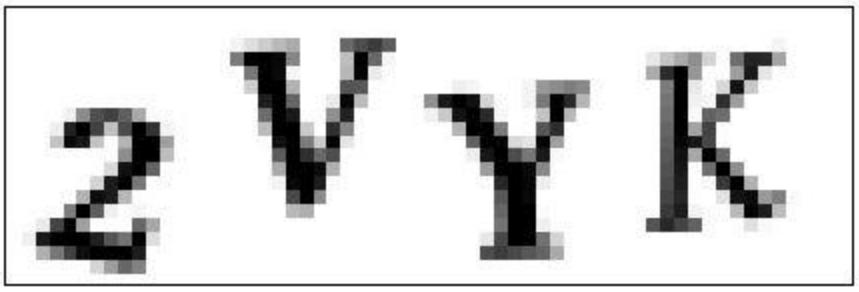

Fig. 7 (i) Raw CAPTCHA Image

And then we'll convert the image into pure black and white (this is called **thresholding)** so that it will be easy to find the continuous regions:

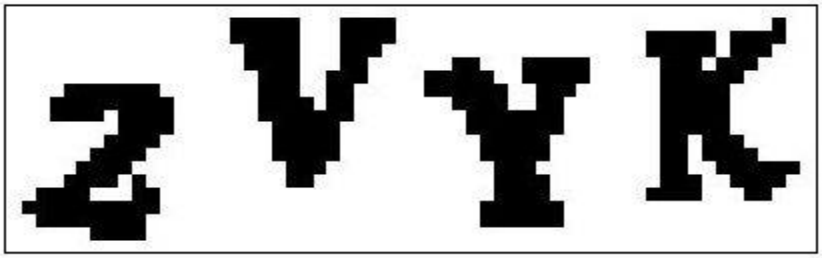

Fig. 7 (j) Threshold Image of Raw CAPTCHA Image

Next, we'll use OpenCV's findContours() function to detect the separate parts of the image that contain continuous blobs of pixels of the same color:

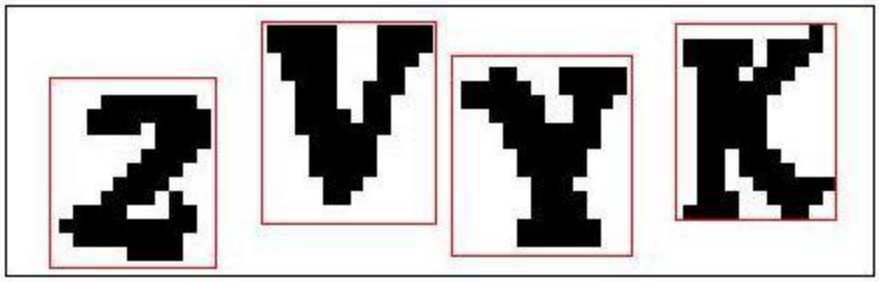

Fig. 7 (k) Split Image in Single Character

Then it's just a simple matter of saving each region out as a separate image file. And since we know each image should contain four letters from left-to-right, we can use that knowledge to label the letters as we save them. As long as we save them out in that order, we should be saving each image letter with the proper letter name.

But, sometimes the CAPTCHAs have overlapping letters like this:

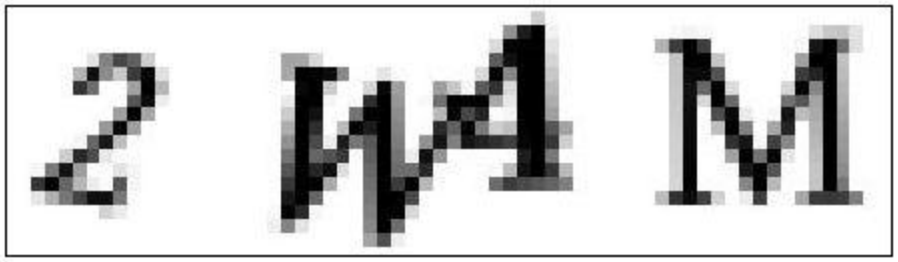

Fig. 7 (l) Overlapping Of Characters

That means that we'll end up extracting regions that mash together two letters as one region:

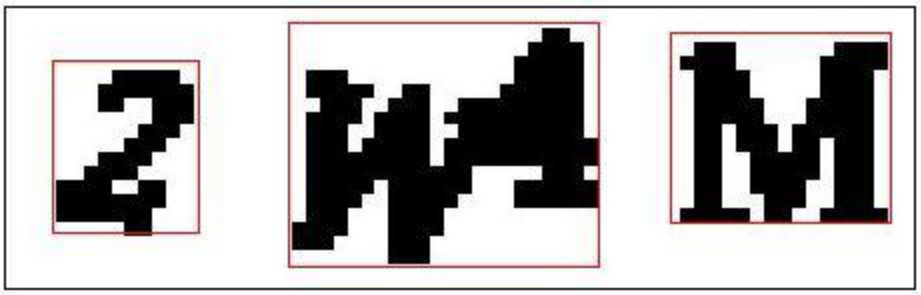

Fig. 7 (m) Make Two letters as A One Region

If we don't handle this problem, we'll end up creating bad training data. We need to fix this so that we don't accidentally teach the machine to recognize those two squashed-together letters as one letter.

A simple hack here is to say that if a single contour area is a lot wider than it is tall, that means we probably have two letters squished together. In that case, we can just split the conjoined letter in half down the middle and treat it as two separate letters:

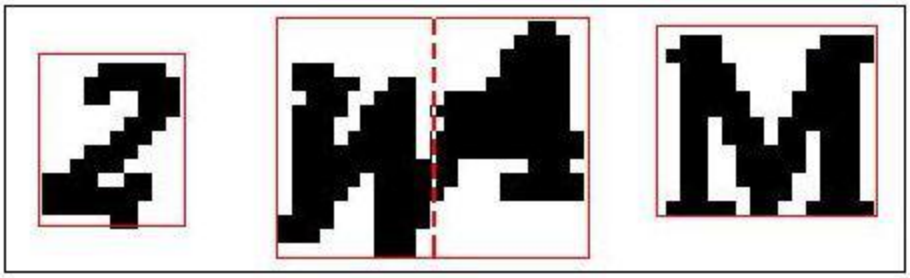

Fig. 7 (n) Split Wider Region in Single Character

Now that we have a way to extract individual letters, let's run it across all the CAPTCHA images we have. The goal is to collect different variations of each letter. We can save each letter in it's own folder to keep things organized.Here's a picture of what our "W" folder looked like after we extracted all the letters:

Fig. 7 (o) Some of the "W" letters extracted from DATASET

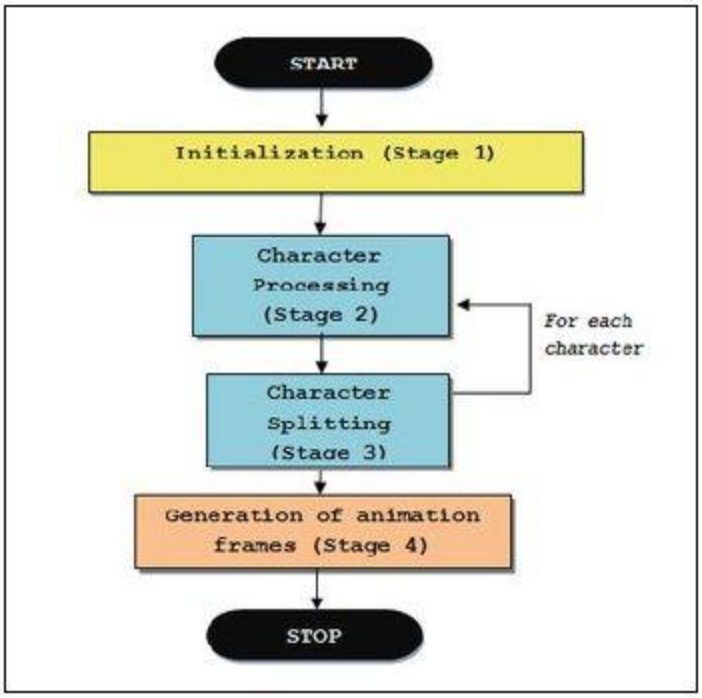

Fig. 7 (p) Flow Chart Of The System

CHAPTER 8
ISSUES IN CAPTCHA BREAKING SYSTEM

There are many issues with CAPTCHAs, primarily because they distort text and images in such a way that, sometimes it gets difficult for even humans to read. Even the simplest, but effective CAPTCHA, like a mathematical equation "What is the sum of three and five?" can be a pain for cognitively disabled people.

8.1 Usability Issues with Text Based CAPTCHAs:

Are text CAPTCHAs like Gimpy, user–friendly? Sometimes the text is distorted to such an extent, that even humans have difficulty in understanding it. Some of the issues are listed in the table 1.

Distortion becomes a problem when it is done in a very haphazard way. Some characters like 'd' can be confused for 'cl' or 'm' with 'rn'. It should also be easily understandable to those who are unfamiliar with the language.

Content is an issue when the string length becomes too long or when the string is not a dictionary word. Care should be taken not to include offensive words.

Presentation should be in such a way as to not confuse the users. The font and colour chosen should be user friendly

Table 1 : Usability Issues in Text Based CAPTCHAs

<table>
<tr><th>CATEGORY</th><th>USABILITY ISSUE</th></tr>
<tr><td>DISTORTION</td><td>Distortion method and level
Confusing characters
Friendly to foreigners?</td></tr>
<tr><td>CONTENT</td><td>Character Set
String Length: How long? / Predictable or not?
Random or dictionary word?
Offensive word</td></tr>
<tr><td>PRESENTATION</td><td>Font type and size
Image size
Use of colour
Integration with web pages</td></tr>
</table>

8.2 Usability of Audio CAPTCHAs:

In audio CAPTCHAs, letters are read aloud instead of being displayed in an image. Typically, noises are deliberately added to prevent such audio schemes from being broken by current speech recognition technologies.

Distortion: Background noises effectively distort sounds in audio CAPTCHAs. There is no rigorous study of what kind of background noises will introduce acceptable sound distortion. However, it is clear that distortion methods and levels, just as in text based CAPTCHAs, can have a significant impact on the usability of audio CAPTCHAs. For example, an early test in 2003 showed that the distorted sound in an audio CAPTCHA that was deployed at Microsoft's Hotmail service was unintelligible to all (four) journalists, with good hearing, that were tested. Due to sound distortion, confusing characters can also occur in audio CAPTCHAs. For example, we observed that it is hard to tell apart 'p' and 'b'; 'g' and 'j', and 'a' and '8'. Whether a scheme is friendly to non-native speakers is another usability concern for audio CAPTCHAs.

Content: Content materials used in audio CAPTCHAs are typically language specific. Digits and letters read in a language are often not understandable to people who do not speak the language. Therefore, unlike text-based schemes, localisation is a major issue that audio CAPTCHAs face.

Presentation: The use of colour is not an issue for audio CAPTCHAs, but the integration with web pages is still a concern. For example, there is no standard graphical symbol for representing an audio CAPTCHA on a web page, although many schemes such as Microsoft and reCAPTCHA use a speaker symbol. More importantly, what really matters for visually impaired users is that the html image alternative text attached to any of the above symbol should clearly indicate the need to solve an audio CAPTCHA.

When embedded in web pages, audio CAPTCHAs can also cause compatibility issues. For example, many such schemes require JavaScript to be enabled. However, some users might prefer to disable JavaScript in their browsers. Some other schemes can be even worse. For example, we found that one audio scheme requires Adobe Flash support. With this scheme, vision-impaired users will not even notice that such a CAPTCHA challenge exist in the page, unless Flash is installed in their computers - apparently, no text alternative is attached to the speaker-like Flash object, either.

8.3 Limitations

1. Since we are performing on dataset,it is difficult for machine to identify where CAPTCHA image is situated in the website.
2. Machine can't refresh the CAPTCHA automatically so there is no way for breaking another CAPTCHA.
3. In this we have considered only text based CAPTCHA. So to solve some math solution CAPTCHAs we require Natural Language Processing of image.
4. Some picture Identification CAPTCHAs are a major issue for us.But we will try to focus in that in future.
5. Google's reCAPTCHA is a big challenge for us because they monitor the mouse events,they use some cookies and biggest one is in 50-55 seconds we have to solve that reCAPTCHA.

CHAPTER 9

EXPERIMENTAL RESULTS

9.1 Results of Single Character Recognition

By using algorithms and methods which we have discussed in Chapter 6 the accuracy and prediction can be determined.

Table 2 : Performances of Different Algorithms on Test Dataset and Prediction Dataset

ALGORITHM	TEST SCORE	PREDICTION
Linear SVM (Single Letter)	60%	0.1%
K-means (Single Letter)	30%	--NA--
CNN (Single Letter)	99%	60%
CNN (VGG-19 transfer ,Single Letter)	95%	70%
CNN moving window (4 letter)	38%	0.50%
Multi -CNN (4-letter)	76%	0.75%

Unsupervised learning and linear classifiers will not perform well in recognizing single-letter CAPTCHAs. The test accuracy of SVM is only 69%, which is reasonable because SVM is susceptible to geometrical noises and not good at generalizing the abstract features of letters. It is also not surprising that the prediction accuracy of SVM is only 0.1% since the letter fonts in the prediction dataset are quite different from the font in the training dataset. For k-means, the performance is even worse,because most of the data cannot be well clustered. The unsatisfying performance of these algorithms shows the drawbacks of non-DL based learning in recognizing CAPTCHAs.

Meanwhile, a convolutional neural network trained on a single letter achieved significantly superior accuracies. The in-house architecture easily achieved test scores of 99% while the transferred VGG-19 achieved test accuracies of 95%. The disparity in the two can be attributed to the fact that we trained a lot of parameters in VGG-19, which probably led to some overfitting. However, we observe that when we try predicting on single-letter CAPTCHAs generated from new fonts, the accuracy of the transfer learned CNN wins out, achieving 70% accuracy vs 60% accuracy for our in-house architecture.

9.2 Multi-letter CAPTCHA Recognition

For multi-letter CAPTCHA recognition, the moving-window algorithm will achieve only half the accuracy of the multi-CNN algorithm. This is mainly because the moving-window algorithm involves explicit character segmentation, which is nontrivial given the fact that the letters have different widths and gaps. As a result, the moving-window algorithm may mis-recognize two letters as one letter, or part of a letter as another letter. In the multi-CNN algorithm, however, each sub-image crop is wide enough to fully contain the target letter. Therefore, it is the neural network's job to identify the single letter from a background of other digits and noise. The downside of this is that it inevitably makes the CNNs harder to train. In fact, in principle we could use the full image to train the four single-letter CNNs, but then the background is even noisier so that the CNNs did not learn well.

The prediction accuracies for both multi-letter CAPTCHA algorithms are below 1%. This is because both algorithms are based on single-letter CNNs. The accuracy of single-letter CNN must be raised to the power of 4 to give an upper bound for the multi-letter algorithms. We are also aware that there is significant error when the algorithms try to identify where a letter is. Assuming 60% accuracy for single-letter CNN and 50% accuracy for position identification, we expect an overall upper bound of (60%×50%)4 = 0.8% accuracy, which will agree with our experimental results.

CHAPTER 10

CONCLUSION

In this project, we have tried to decode an image-based CAPTCHA using python library and deep neural networks. We have used convolutional neural networks and recurrent neural networks instead of using the conventional approach of first cleaning a CAPTCHA image, segmenting the image, and recognizing the individual characters. For machine learning problems, we need a large amount of data, so we have generated a dataset of image-based CAPTCHAs. The programs generating this dataset will be publicly available so that they can be used by other people in their research. We have exploited the capabilities of CNNs to work on the images and RNNs to work with sequences. The model was trained on both simple CAPTCHAs and complex CAPTCHAs. The accuracy achieved on fixed-length CAPTCHA was very impressive (99.8% for simple images and 96% for complex images). We tried both fixed length and variable length CAPTCHAs. Both give 99and 55 % accuracy respectively.

It is clear that the more kinds of CAPTCHAs we include in our training set, the more robust our model will become. We have tried to demonstrate this by using a real dataset in our training set, and were able to achieve 99% accuracy. While it is still in early stage, our model performs better than previous work that relies on manually-generated segmentation oriented models. In the end, we are able to provide end-to-end neural network system. Given an image, we will be able to decode the CAPTCHA in that image.

FUTURE WORK

- In this project we have only considered the text based CAPTCHAs.
- So for math solution CAPTCHAs we will implement Natural Language Processing Algorithms to our project and try to break that kind of CAPTCHAs.

- For identifying the Picture Based CAPTCHAs we will try for Google Reverse Image System that will identify similar kind of images that is given in the CAPTCHAs image and we will try to implement that algorithm to our project.
- Also,if possible,we will try and understand the Google's reCAPTCHA and algorithms that are used in that and make that CAPTCHA also be breakable.

REFERENCES

[1] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, Andrew Y. Ng. Deep Speech: Scaling up end-to-end speech recognition, 17 Dec 2014

[2] Tomas Mikolov Et al. Distributed Representations of Words and Phrases and their Compositionality,Oct 16, 2013.

[3] Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Networks. Sep 10, 2014.

[4] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks,14 Apr 2014.

[5] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan, Show and Tell: A Neural Image Caption Generator, 20 Apr 2015

[6] https://benthamopen.com/FULLTEXT/COMPSCI-1-1#R6

[7] Greg Mori and Jitendra Malik. Recognising Objects in Adversarial Clutter: Breaking a Visual CAPTCHA, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 03), Vol 1, June 2003, pp.134-141.

[8] Kumar Chellapilla, Patrice Y. Simard Using Machine Learning to Break Visual Human Interaction Proofs (HIPs) Microsoft Research, one microsoft way, WA. 2005

[9] F. Azam and H. F. VanLandingham. An efficient dynamic system identification technique using modular neural networks. Artificial Neural Networks for Intelligent Engineering, 7:225-230,1997.

[10] Artificial neural networks are changing the world. What are they?. http://www.extremetech.com/extreme/215170-artificial-neural-networks-are-changing-the-world what-are-they. Accessed in Oct 2015.

[11] https://benthamopen.com/FULLTEXT/COMPSCI-1-1#R11

[12] Minsky, M. S. Papert. An Introduction to Computational Geometry. MIT Press, 1969.

[13] Werbos, P.J. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, 1975

[14] Yaniv Taigman,Ming Yang, Marc'Aurelio Ranzato, Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, June 24, 2014.

[15] https://benthamopen.com/FULLTEXT/COMPSCI-1-1#R23

[16] https://benthamopen.com/FULLTEXT/COMPSCI-1-1#R26