CS 297 Report

NEURAL NET CAPTCHA CRACKER

By

Geetika Garg

SJSU ID: 010021128

Email: Geetikagarg07@gmail.com

Project Advisor :

Dr. Chris Pollett

Department of Computer Science

San José State University

One Washington Square

San José, CA 95112

CONTENTS:

INTRODUCTION:

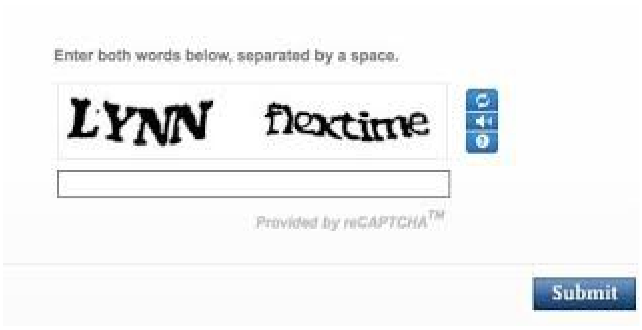
A CAPTCHA (acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart") is a type of challenge-response test used in computing to determine whether or not a user is human. CAPTCHAs were first mentioned in a paper by Moni Naor in 1996. A visual CAPTCHA, often an image with a series of letters and/or numbers, prompts a user to decipher its message. In this project, I am working with Dr. Pollett on neural networks to crack CAPTCHAs. The core idea of the project is to break an image based CAPTCHA. Our goal is to make a program that will remove effects and noise from an image and then use neural nets to recognize a CAPTCHA. We will be training a large sample of data on neural nets to build our OCR.

At present, CAPTCHAs are almost a standard security mechanism for defending against undesirable and malicious bot programs on the Internet. For example, bots that could sign up for thousands of accounts a minute with free email service providers, bots that could send out thousands of spam messages each minute, and bots that could in one act post numerous comments in weblogs (“blogs”) pointing both readers and search engines to irrelevant sites.

The motivation for this project is to show that the security methods used in many online systems are not secure and are prone to attack by hackers. Also, a CAPTCHA is a so called win-win solution, in that if a bot cannot break it, it provides security, but if it is automatically broken then a difficult task in computer vision or related areas has been solved.

Breaking CAPTCHAs is not new. For example, Mori and Malik [5] have broken the EZ-Gimpy (92% success) and the Gimpy (33% success) CAPTCHAs with sophisticated object recognition algorithms. In contrast to such earlier work that relied on sophisticated computer vision algorithms, we are planning to train an end to end neural network system that should extract the features needed for classification with minimal hand tuning. Neural networks have shown great results recently in many domains like linguistics, speech and image processing. Neural networks also have brought down the entry barrier to training such models as one doesn’t need a lot of domain knowledge to massage the inputs to provide the features that a model could learn from. The hidden layers in neural networks extract the features that are useful during learning.

Text-based CAPTCHAs are the most common and widely used form. The image or sound is usually distorted in various ways to make it difficult for a machine to perform the test. When successful, CAPTCHAs can prevent a wide variety of abuses, such as invalid account creation and spam comments on blogs and forums. CAPTCHAs are intended to be easy for humans to perform, and difficult for machines to perform. The Image below shows an example of a CAPTCHA.


Enter both words below, separated by a space.

Neural networks model biological neural systems. I will be using a simple feed forward network which is a DAG of units that are inspired from biological neurons. Each unit outputs a value that is a function (could be arbitrary non-linear function like sigmoid, tanh, relu etc.) of its inputs. A layer is a collection of units that have common inputs. Often, there is an input layer, an output layer, and possibly many hidden layers. The input layer takes in data, the output layer gives out data, and hidden layers do intermediate processing. The hidden layers extract features from inputs and this is where neural networks outshine other machine learning algorithms as they can be used to build an end to end system which requires very little hand tuning and domain knowledge for things like feature extraction.

CAPTCHAS can have variable number of characters. But a feed-forward network produces a fixed number of outputs. For instance a character recognition feed-forward network would only output a single character. So we need to do some input pre processing before it can be fed into a neural network.

I would be talking about the following deliverables in the subsequent sections but the basic flow is as follows:
We first remove the background clutter (Deliverable 1). In this particular case, the CAPTCHAs images contain a lot of black and white noise, which can be removed. Then the letters are segmented from the image (Deliverable 3). And then they are individually fed into a neural network for character recognition (Deliverable 4).

Note that the neural network can classify a segment as no letter. The reason because the image segmenter would use a moving window for segmentation and a lot of segments wouldn’t contain any letters.

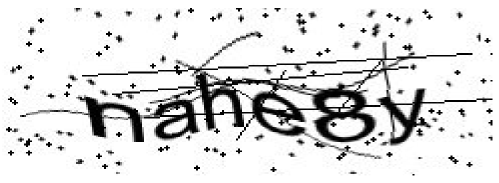
DELIVERABLE 1:

In order to clear the background noise for an image CAPTCHA. We started using an edge detection algorithm.

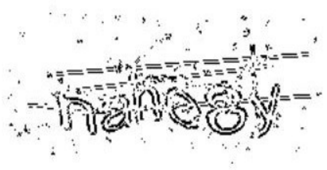
Edges are the straight lines or curves in the image plane across which there is a significant change in brightness. The reason of doing this is because the output displays more compact and abstract representation. Edges corresponds to locations in images where the brightness undergoes a sharp change. An edge detection system works like a high pass filter in frequency domain, so high frequency components pass through while the low frequency components are filtered out. This was the main logic of my program. I used the Python Imaging library to implement the edge detection algorithm. Also, I computed gradients for every pixel which is basically gradient in x and y direction (so a vector with x and y component). And then calculated gradients for neighbor pixels. The goal was to keep the pixels which have gradients in roughly the same direction as that of their neighbors. So we computed cosine distance between gradient vectors of pixels and their neighbors. The algorithm kept pixels which have more than a threshold number of neighbors with low cosine distance gradients.

Input-Output

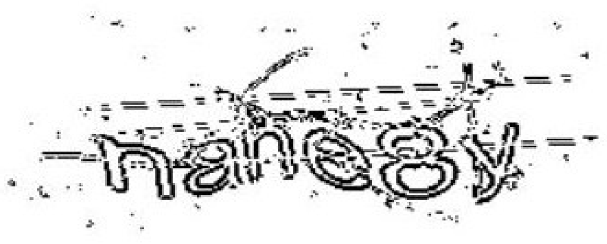
Example:This is what my code outputs on following inputs.
Input Image



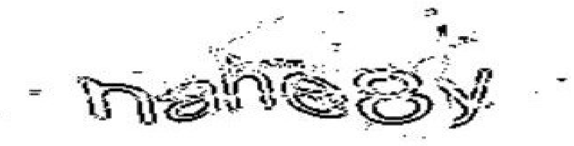
Output Image1



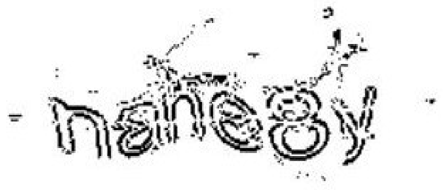
Output Image2



Output Image3



Output Image4



As you can see in the output image, only boundary pixels are shown and the rest are filtered out. I changed different features and got different results as shown in output Image1, Image2, Image 3 and Image4. Further, I am planning to take the coordinates of these boundaries to break the image into different pieces of characters which will serve as input to my neural net model.

DELIVERABLE 2:

While moving forward towards our goal, we spent some time in reviewing about CAPTCHAs and the work done in this field before.

Moni Naor introduced CAPTCHA in 1996 for the first time. The basic idea for CAPTCHA is taken from the cryptography where a sender sends a challenge to a receiver to authenticate its identity. Many Ideas were proposed as to what we could be used for the CAPTCHA problem. For example, one task might be to recognise male or female, check facial expression happy or sad, fill in words or disambiguation like what does “It” refers to in a paragraph.

Initially, it was thought that CAPTCHAs can be used for security because there had not been any AI programs which could perform the problem task. CAPTCHAs are based on the underlying assumption that it is difficult to solve this AI problem. So it was decided to use one of the AI problems as a CAPTCHA as if it is solved then then a difficult AI problem would be solved which would help in research. So, it was a win win situation. CAPTCHAs were first developed by Alta Vista to prevent “bots”.

Where is there a need for CAPTCHAs?
Online Polls - In case of online polls, we have to make sure that there is no bot or computer which is voting automatically. To enforce a CAPTCHA can be used to make sure that only humans are polling.

Free Email Services - There are many free email services available. However, if there are too many accounts then it can lead to poor service. All the customers won’t be served properly. Again, CAPTCHAs could be useful here.

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." Since search engine bots 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.

Prevention of 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.

The Following are the reasons for using Character Recognition in CAPTCHAs:

1) Recognition of objects requires prior knowledge of the scene and different regions have different names of an object. It would not have been universal.
2) Character has unique(say 256 ascii characters) data set and all of it is present on keyboard clearly. No confusion would be there.
3) Also, no prior knowledge is required by the user.
4) Characters were designed by humans for humans and humans have been trained at the task since childhood.
5) Character-based CAPTCHA can be generated quickly.

Steps Used to make a CAPTCHA:

Building an actual reading-based CAPTCHA requires one to make several independent choices.
a) Character set: The character set to be used in the HIP.
b) Affine transformations: Translation, rotation, and scaling of characters.
c) Adversarial clutter: Random arcs, lines, or other simple geometric shapes that intersect with the characters and themselves.
d) Image warp: elastic deformations of the HIP Image at different scales.
e) Background and foreground textures: These textures are used to form a colored HIP image from a bi-level or grayscale HIP mask generated by using a) through d)

Key Points to consider in CAPTCHA breaking:

1) Comparing computer and human performance, People have said that that computers are better than humans at recognition.
2) Decided workflow for project:
    1)Preprocessing
    2)Segmentation
    3)Training the data set
    4)Testing

DELIVERABLE 3:

After removing noise from a CAPTCHA image, we moved forward to segment the image into individual characters.

The aim for this deliverable was to segment an image based on the window size. Right now I have manually set the window width and size to work for a particular type of CAPTCHA so that I can start working on training sample neural network.

Script:

```
img = cv2.imread(filename)
x = 40
y = 10
height = 100
width = 40
i = 1
while width < len(img[0]):
    y = width
    width = y + 30
    crop_img = img[x:height, y:width]
    cv2.imwrite('cropped'+ str(i)+'.png', crop_img)
    i += 1
```

Example:This is what my code outputs on following inputs.

Input Image

Output Image1

Output Image2

Output Image3

Output Image4

Output Image4

Output Image4

Output Image4

DELIVERABLE 4:

Now that we had characters from the image, using a neural network we tried to recognize what these characters were and then recognize the full CAPTCHA

I created a dataset of character images using SkimpyGimpy. It is a synthetically generated dataset. I have tried training a model with perceptron using Theano framework but the results were really bad. It was giving 92% error rate. Now, we are thinking of using convolutional neural net. It is under progress. I will be working on this deliverable over summer and will update it by then.

CONCLUSION

In conclusion I feel that I have done the research part of my project. Now, I know what parts need to be worked on next semester. This paper covers most of the basic algorithm needed for this project. I have researched the open source libraries available for image processing. I succeeded in removing noise from CAPTCHA images. I used canny edge detection technique for edge detection. Also, I took care of disturbance by calculating gradients in x and y direction and keeping only those pixels whose gradients align with that of their neighbors as described in deliverable 1. I have researched on the work done in this field so far. We now have a base for our project ready.

I am also learning about neural nets . I used theano library to train a simple neural network but the results were not good(92% error rate). I train the model by synthetically creating the data set of alphabets using SkimpyGimpy library. I will dig into this more over summer.

Further, I would be applying it on a bigger dataset and test the results. I want to build a standalone system which others can use to break captchas. I think more work needs to be done on training and tuning the parameters of the neural network used for character recognition .

REFERENCES:

1) Kumar Chellapilla, Patrice Y. Simard Using Machine Learning to Break VisualHuman Interaction Proofs (HIPs) Microsoft Research, one microsoft way, WA 98052 -2005

2)Anjali Avinash Chandavale and A. Sapka . Security Analysis of CAPTCHA, 2012

3) Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski-Building Segmentation Based Human-friendly Human Interaction Proofs (HIPs) Microsoft Research, One Microsoft Way, Redmond, WA, USA 98052 ,200**5**

4)David R. Martin, Charless C. Fowlkes, Jitendra Malik(2002).Learning to Detect Natural Image Boundaries Using Brightness and Texture.U.C. Berkeley, Berkeley, CA 94720 Stuart J. Russell, Peter Norvig. Artificial Intelligence: A Modern Approach .

5)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.