CAPTCHA Solving With Neural Networks

Tianhui Cai
TJHSST Computer Systems Lab 2007-2008

Abstract

CAPTCHAs, Completely Automated Public Turing tests to tell Computers and Humans Apart, are tests to determine if a user is a human or a machine. Variations include audio and visual CAPTCHAs, which are often found on registration webpages to prevent automated (spam) registration. The focus of this project is on visual CAPTCHAs, which consist of an image with letters or numbers that are to be typed into a form by the user. The goal of this project is to devise a system to break a particular CAPTCHA that can be found on captchas.net.

Background

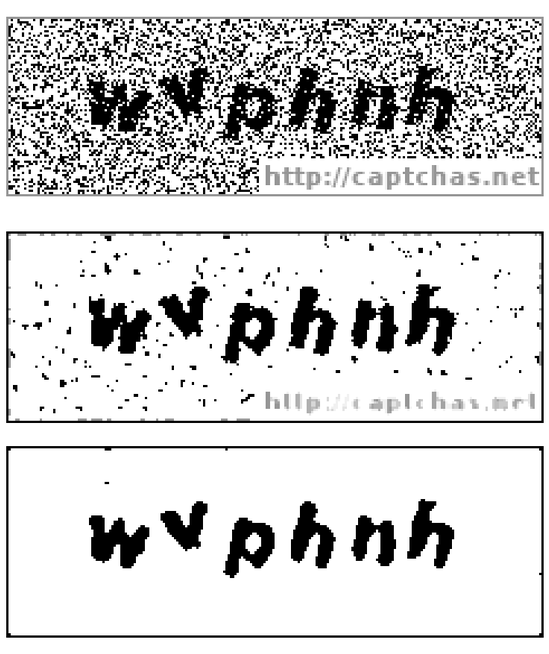
A CAPTCHA's purpose is to distinguish between a computer and a human by presenting a challenge that is easy for most humans, but difficult for computers. The visual CAPTCHA, often an image with a series of letters and/or numbers, prompts a user to decipher its message. The image often contains distortions to make it difficult for a computer to read, including rotation, translation, scaling, background noise, and color.

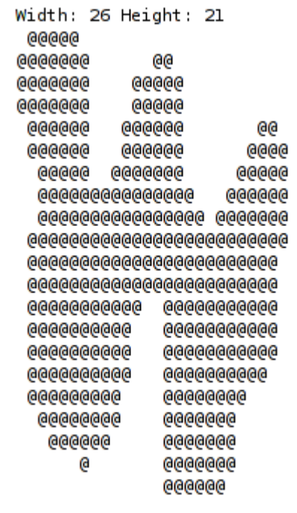
For a computer to beat a CAPTCHA, it must identify which pixels comprise the letters. This is usually done after removing the background clutter. After the letters are separated from the image, they must be identified, which is often done with a neural network.

Neural networks model biological neural systems. 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. Neurons fire with a value between 0 and 1; when a neuron receives input, it weights the inputs by neurons connected to it, and fires depending on the sum of the weighted inputs. Through training, such a network adjusts the weights on each layer depending on the error produced. The rate at which a network corrects its weights is called the learning rate.

Because the entire network is highly connected, neural networks can model highly complex, nonlinear systems and can be proficient in classification and pattern recognition.

Previous research in this subject has been done for the past decades. Sherin M. Youssef and Shaza B. AbdelRahman researched license plate recognition in their paper, *A Smart Access Control Using An Efficient License Plate Location And Recognition Approach*.





Procedure and Methods

The general procedure consists of several steps. The first step is the acquisition of the image, which is done by downloading them from captchas.net. This website provides a free CAPTCHA service, with a formula using the URL to tell you what an image says. The images were downloaded and named with Ruby, with filenames being the sequence of letters depicted in the image.

The second step is to remove background clutter. In this particular case, the CAPTCHAs provided contain a lot of black and white noise, which can be removed with a median filter. In contrast to a Gaussian blur, a median filter does not blur the image, thus saving fine details of the image while removing noise. Java was used.

The next step is segmentation – separating out the letters from the background. It is performed in this project using flood-fill. Although this method has limitations, it is adequate for this purpose. The letters were also scaled and centered in a 9x11 box. This step is also done in Java. The images of the letters are turned into arrays and outputted into a text file as lines of decimals from 0 to 1, representing shades from white to black.

The last step is the identification of the characters that have been segmented. This is done with a three-layer backpropagation neural network. The network is first trained using 1742 letter images, read from a text file, so that it learns how to identify those characters. Afterwards, it is tested on a new set of 600 different letter images, using what it has learned to identify those images. A key feature of the neural network is that it can be saved into a file and reloaded.

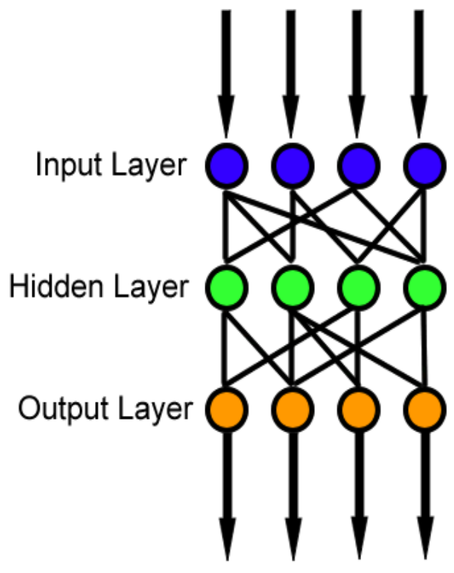
Testing was done to gauge the effects of learning rate and training iterations on accuracy of identification.

Results and Conclusion

A successfully trained neural network was produced at the end of this project. In addition, each component process by itself worked – converting a folder of images into a data training set in the form of a text file, saving and loading a neural network, and training and testing a neural network.

Number of Iterations	Accuracy
10000	53.0%
100000	87.2%
500000	91.5%
1000000	91.5%

Learning Rate	Accuracy
0.05	86.0%
0.1	87.2%
0.2	91.2%
0.4	87.0%



The accuracy improves with number of iterations up to a point after which accuracy stabilizes at a maximum. Accuracy also improves with learning rate up to a point at which it overshoots, reducing accuracy. The program often mistook q's for g's and i's for l's, probably due to the low quality of the images that were fed into it.

Future versions of this program would be able to deal with more distortions, have better segmentation, remove noise more efficiently, and implement more sophisticated neural network structures.