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.

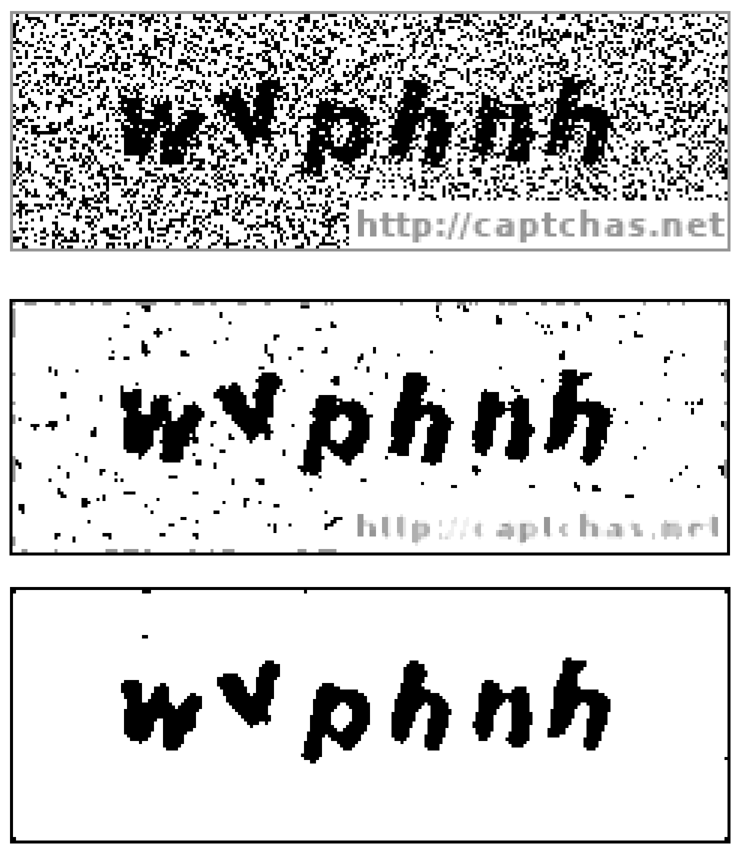
Background

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. A common example is the visual CAPTCHA, which is an image with a series of letters and/or numbers, and a user is prompted to enter its value into a box. The image often contains distortions to make it difficult for a computer to read. These distortions include 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. Although each component is simple, because the entire network is highly connected, neural networks can model highly complex, nonlinear systems and can be proficient in classification and pattern recognition.

Research on neural networks has been in existence for several decades. In particular, the use of neural networks for classification has been used. Le Cun et al at AT and T laboratories has demonstrated that with a particular set of connections with a multi-layered perceptron, handwritten digit recognition can be done extremely efficiently.



```
Width: 26 Height: 21
 @@@@@
@@@@@@@      @@
@@@@@@@    @@@@@
@@@@@@@    @@@@@
 @@@@@@   @@@@@@       @@
 @@@@@@   @@@@@@      @@@@
  @@@@@  @@@@@@@     @@@@@
  @@@@@@@@@@@@@@@   @@@@@@
  @@@@@@@@@@@@@@@@ @@@@@@@
 @@@@@@@@@@@@@@@@@@@@@@@@@
 @@@@@@@@@@@@@@@@@@@@@@@@
 @@@@@@@@@@@@@@@@@@@@@@@@
 @@@@@@@@@@@  @@@@@@@@@@@
 @@@@@@@@@@   @@@@@@@@@@@
 @@@@@@@@@@   @@@@@@@@@@@
 @@@@@@@@@@   @@@@@@@@@@
 @@@@@@@@@    @@@@@@@@
  @@@@@@@@    @@@@@@@
   @@@@@@     @@@@@@@
      @       @@@@@@@
              @@@@@@
```

Procedure and Methods

The general procedure for this endeavor consists of several steps. The first step is the acquisition of the image, which is done by downloading them from captchas.net. This particular website provides a free CAPTCHA service, with a formula 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, it does not blur the image, thus saving fine details of the image while removing noise. This is done in Java.

The next step is segmentation – separating out the letters from the background. This must happen after the removal of the background clutter. It is performed in this project using flood-fill. Flood-fill has the advantage that it is simple to code. However, it will not be useful if two letters are conjoined or if a single letter is broken up into multiple parts. In this scenario, these are not significant problems and cases in which this happens are thrown out. This step is also done in Java.

The last step is the identification of the characters that have been segmented. This will be done with a neural network – a three-layer backpropagation neural network. It will first be trained on a training set, so that it learns how to identify characters. Afterwards, it will be run to identify images using what it has learned. A key feature of this neural network must be to save the network into a file, so that it can be loaded and trained multiple times.

Testing

The image processing – noise removal – is tested on sample CAPTCHA images. It works. Saving the neural network works, too.

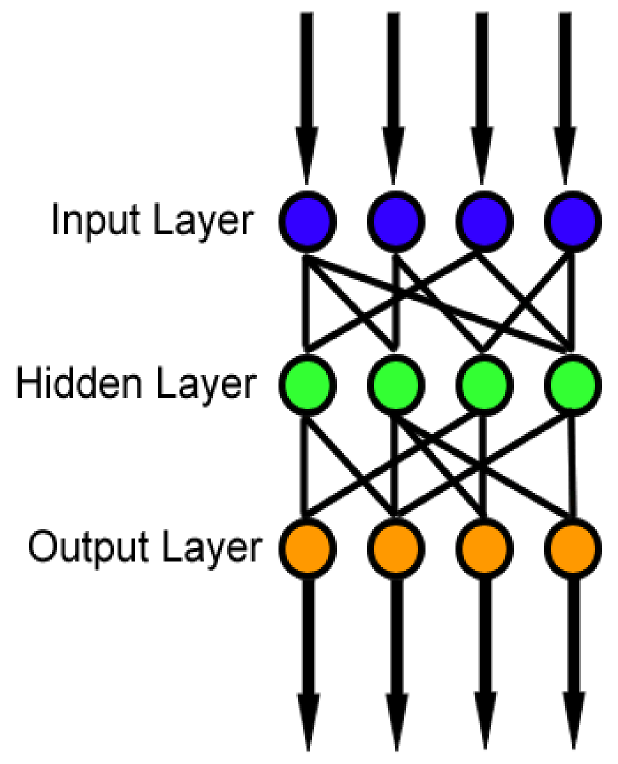
A set of 50 training images have been downloaded with filename as the letters depicted in the images. This works.

The neural network is tested by testing it on simple inputs and outputs, such as AND, OR, and XOR. This works, too.

The next step is to use the outputs of the image processing and segmentation as the inputs to the neural network for training, and then for testing.

Results and Conclusion

This section cannot be completed at the moment.


Input Layer
Hidden Layer
Output Layer