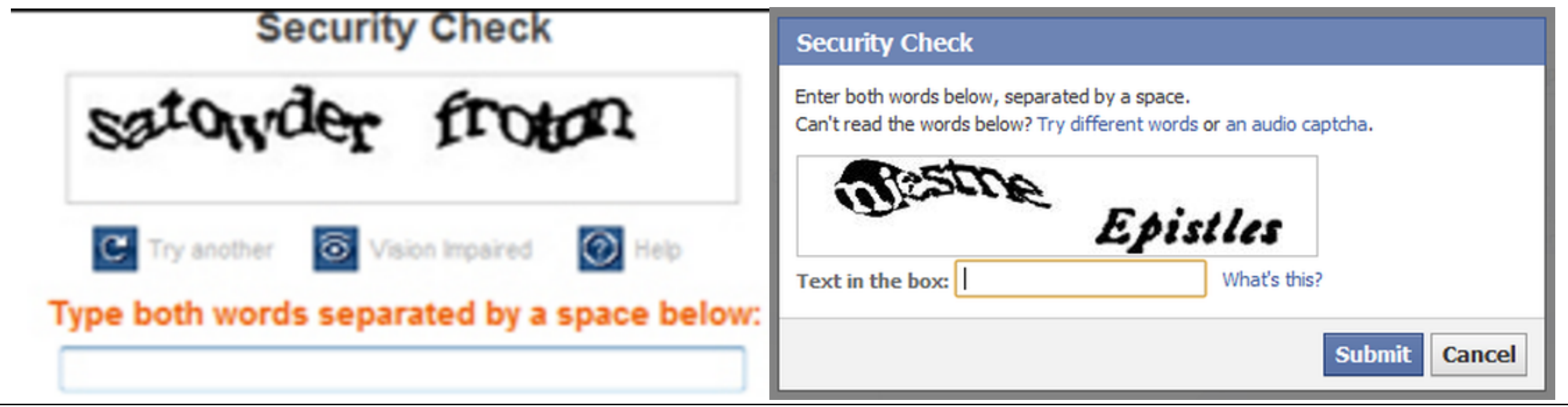
Capturing CAPTCHAS: Captchas Recognition Using Neural Network

Benjamin Hinton[2], Jun Song[1,3], Michael Tong[1]
[1]Department of Computer Science, UC-Berkeley
[2]Department of Bioengineering, UC-San Francisco & UC-Berkeley Joint Program
[3] Department of Statistics, UC- Berkeley

Motivation:

CAPTCHA (Completely Automated Public Turing test to Tell Computers and Humans Apart) images have been used for years to help separate actual human users from computer controlled counterparts online and are important in:

- Preventing spam account creation
- Preventing automated ticket purchases
- Securing actions in email, banking, and other sectors



Security Check

Type both words separated by a space below:

Security Check

Enter both words below, separated by a space.
Can't read the words below? Try different words or an audio captcha.

Text in the box:

What's this?

Figure 1: Sample CAPTCHA Images provided by TicketMaster and Facebook

We hypothesize we can create a Neural Network that can correctly classify CAPTCHA images, which would indicate a major flaw in security methods employed by a number of websites and online services.

Data Acquisition:

We used the python package wsy_captcha to produce n = NUMBER images for training and n = NUMBER images for validation All images have the numbers and letters skewed, rotated, and have several occlusions.

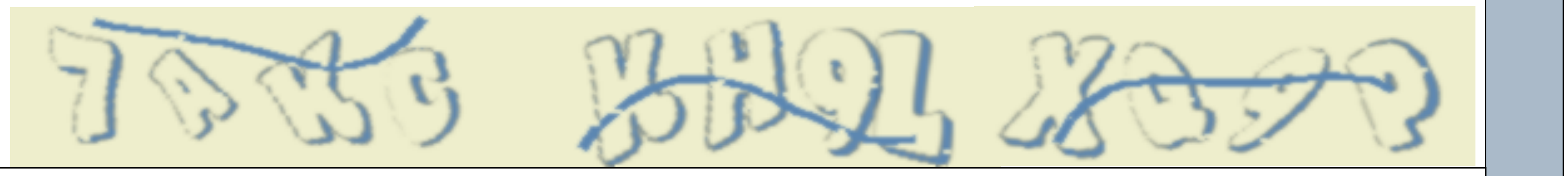


Figure 2: Sample CAPTCHA Images produced by wsy_captcha

Methods: Data Preprocessing

To **enhance contrast** of the images, we converted the images from RGB to grey scale images. This aided in classification performance, reduced memory usage and increased computational speed.

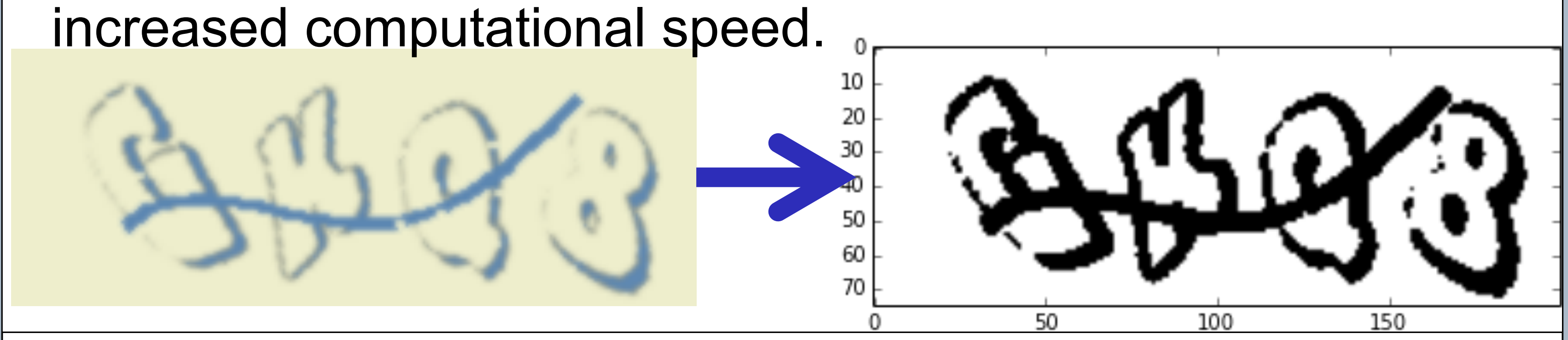


Figure 3: Images after conversion to grey scale images.

Methods: Data Preprocessing Continued

In these CAPTCHA images a mostly horizontal line occludes the image and reduces classification performance. We wrote a script to **remove the horizontal line** and aid in the classification.

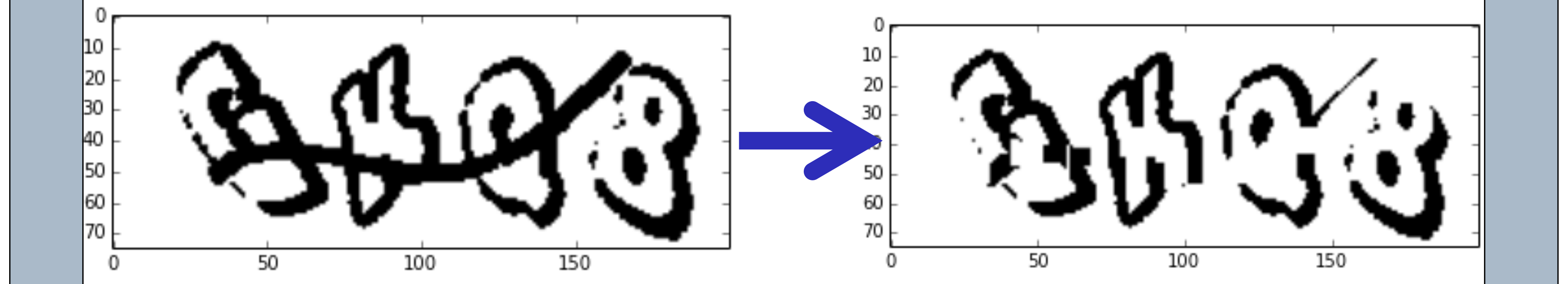


Figure 4: Images after removal of the occluding line.

Algorithms in CAPTCHA solvers typically either classify each letter individually or try to classify an entire word[1]. In our method we chose to solve each letter individually, so we wrote script to separate each character into separate images to be classified:

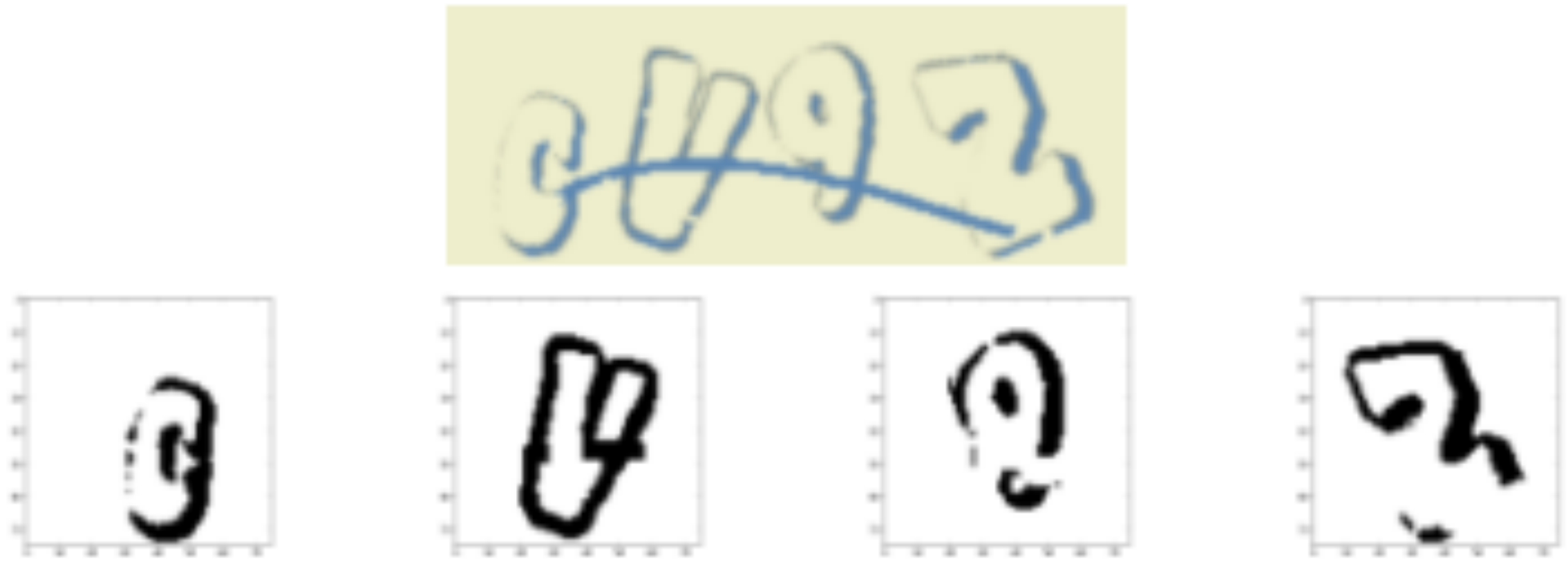


Figure 5: Images after separating each character individually.

Methods: Neural Net:

Our neural net architecture is loosely based off of Alexnet, with additional inspiration from other studies [2,3]. Inputs into the net were 75x75 cropped pixel sections from the above segmented character images.

layer name	layer type	nonlinearity
conv1	5x5x8 convolutional layer	RELU
pool1	2x2 max pooling layer	none
conv2	5x5x16 convolutional layer	RELU
pool2	2x2 max pooling layer	none
conv3	3x3x32 convolutional layer	RELU
pool3	2x2 max pooling layer	none
fc1	7x7x32 → 1024 fully connected layer	RELU
fc2	1024x36 fully connected layer	RELU

Figure 6: Architecture of the neural net used for classification.

Results:

Our data preprocessing methods worked quite well to reduce noise and aid in classification. There were some situations where the line removal method and the character segmentation method produced faulty results, which affected training accuracy. After training and testing on a validation set, we computed a test accuracy of 74%.

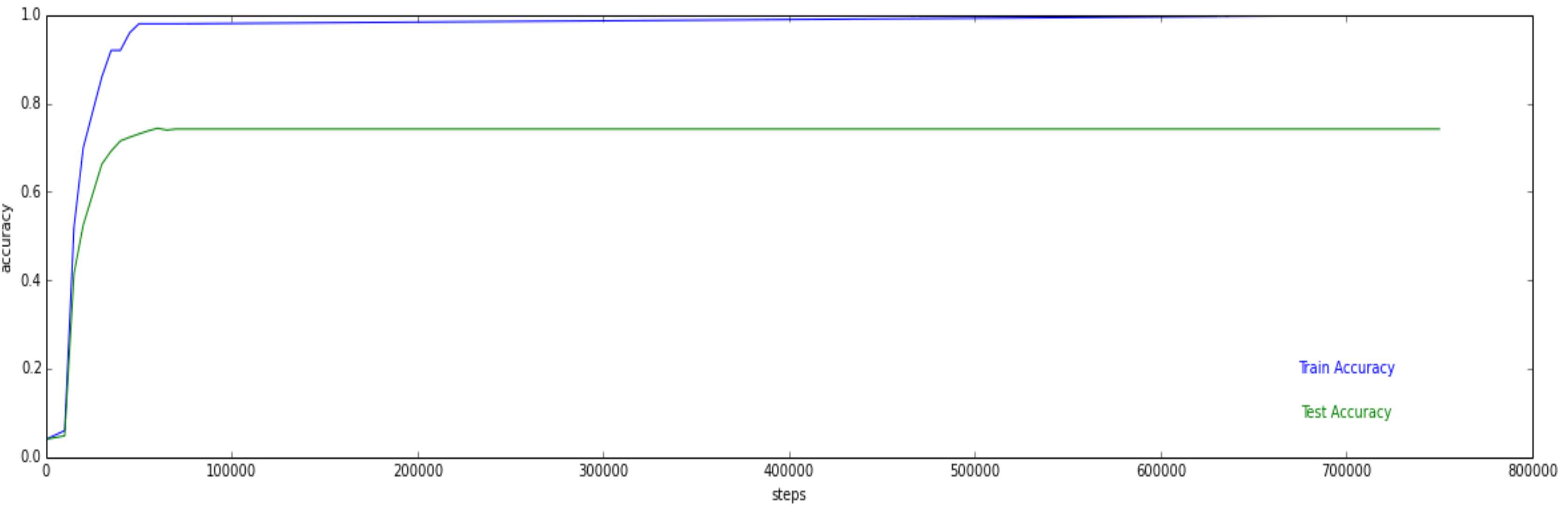


Figure 7: Classification results on validation set after training.

Conclusions and Future work:

Overall this project was a successful demonstration of how to use convolutional neural networks to read text in an adversarial setting. However, there are more ways in which the performance of text recognition can be improved.

One of the flaws in our system is how the segmentation of the image affects the later part of the classification. One way to get around this problem would be to write an end-to-end convolutional network that would encapsulate the tasks of segmenting the letters and identifying them. Another benefit of this method would be it is that neural networks are much faster than our current segmentation code.

References

1. Mori G, Malik J. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In: Computer Vision and Pattern Recognition, 2003 Proceedings 2003 IEEE Computer Society Conference on [Internet]. IEEE; 2003 [cited 2016 Mar 31]. p. I – 134. Available from: ttp://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1211347
2. Shrivastava V. Artificial Neural Network Based Optical Character Recognition. Signal Image Process Int J. 2012 Oct 31;3(5):73–80.
3. Jønsson M, Bothe H-H. OCR-Algorithm for Detection of Subtitles in Television and Cinema. In: CVHI [Internet]. Citeseer; 2007 [cited 2016 Apr 1]. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.143.370&rep=rep1&type=pdf