Captcha Recognition using convolutional neural networks with low structural complexity

Haolin Yang
University High School, CA 92612, Orange County
Email: kevinhlyang@outlook.com, 23yanghaolin@iusd.org

Abstract: CAPTCHAs are automated tests designed to distinguish between humans and computers. They could be easily solved by humans, but they become challenging for machines to solve, therefore preventing programs from abusing online services and occupying internet resources. Using Convolutional Neural Networks, the CAPTCHA tests could be solved automatically at high efficiency. Current approaches of high accuracy CAPTCHA recognition can be structurally complicated. As a result, our team explored a different approach to solve CAPTCHAs using a Convolutional Neural Network that is more efficient in terms of structural complexity and run time, with image processing being a possibility to enhance the accuracy. We also tested our networks on CAPTCHA datasets with character adhesion and background noise.

1. INTRODUCTION
CAPTCHA (Completely Automated Public Turing Test to tell Computers and Humans Apart) is an automated test created to prevent websites from being repeatedly accessed by an automatic program in a short period of time and wasting network resources. Most service providers online have implemented CAPTCHA tests before the user is allowed to commit certain actions, such as submitting a form. Among all the CAPTCHAs, commonly used types contain low resolution, deformed characters with character adhesions and background noise, which the user must read and type correctly into an input box. This is a relatively simple task for humans, taking an average of 10 seconds to solve [1], but it presents a difficulty for computers, because such noise makes it difficult for a program to differentiate the characters from them. However, using Convolutional Neural Networks (CNN), these CAPTCHA tests could be solved efficiently and accurately by computers. Creating a simple, efficient and accurate method to recognize CAPTCHAs can assist with the verification of the security of existing forms of CAPTCHAs and the creation of new, more secure ones. The same approach for CAPTCHA recognition could also be applied in several other fields, including handwriting recognition, license plate recognition, and many more. The focus of our research is to optimize a CNN model for structural simplicity, accuracy, and amount of training data required. To further improve the accuracy of the models and decrease the training data needed, we also experimented with preprocessing the CAPTCHA images using methods such as Fourier Transform with the goal of eliminating background noise.

2. BACKGROUND
Conventionally, CAPTCHAs could be recognized by locating the numbers and characters within an image and identifying them individually after segmentation. For example, Yan and Ahmad [5] was able to segment the Microsoft CAPTCHAs with an average time of ~80ms and a success rate of higher than 90%. After segmentation, multiple classifiers were used for recognition, but the combined segmentation and recognition success rate was only 60%. Furthermore, the segmentation of CAPTCHAs become more
difficult to implement in newer samples, as characters are difficult to be segmented with skews, character overlaps, and character adhesions added into the CAPTCHAs. Because of this, many of the newer CAPTCHAs are more similar to handwritten text. In 1998, LeCun et al. [2] proposed that Convolutional Neural Networks (CNN) outperforms most other techniques when compared on a handwritten digit recognition task. These networks are multilayer and perform classification at the end. In 2013, Goodfellow et al. [3] proposed that the steps of localization, segmentation, and recognition which are performed separately in conventional approaches could be integrated together using a deep CNN network that operates directly on the image pixels and achieved 96% accuracy at recognizing street numbers when evaluated on the SVHN dataset. In 2015, Stark et al. [6] proposed a method in which Active Deep Learning was used for CAPTCHA recognition to solve the problem of insufficient training data for CNN networks. Recently, more complex deep CNN networks had been used for CAPTCHA recognition and achieved high accuracy. For example, Jing Wang et al. [7] proposed a new Dense Convolutional Network (DenseNet) model which they called DFCR for CAPTCHA recognition. The model reduced the number of convolutional blocks and built corresponding classifiers for different types of CAPTCHAs to reduce the memory consumption of the DenseNet model and reached recognition accuracy of above 99.9% on CAPTCHAs with background noises and character adhesion. Many other methods other than CNN networks had also been proposed for CAPTCHA recognition. For example, Liang Zhang et al. [4] proposed a recognition method for CAPTCHAs using Recurrent Neural Networks (RNN) composed of Long Short-term Memory (LSTM) blocks.

3. CNN NETWORK
After reading about current methods of high accuracy CAPTCHA recognition, such as using active learning or complicated deep CNNs such as DenseNets, our team set out to determine if CNN models which are more efficient on run time and structural complexity could be used for high accuracy recognition of CAPTCHA images with background noise, character adhesion and blurring. Based on existing methods of recognition, we built three CNN networks for testing. These networks do not necessarily have fewer parameters than other networks, but they have less runtime and complexity compared to other CNN networks such as DenseNet. Figure 1 shows the structure of our first network, Network 1. In the first network, the image initially undergoes three convolutions. Two transition layers with a structure constructed by MaxPool → Batch Normalization (BN) → ReLU are inserted between the three convolutions to implement down-sampling. The third convolution is followed by a BN, a MaxPool, after which the model is flattened. The second network, Network 2, is structurally similar to the first network, with the only difference being MaxPools replacing the transition layers. The third network, Network 3, is based on the second network, with the difference being that the third network only has two convolutions, with a single MaxPool sandwiched in between them. Each of the outputs of all three networks are then fed to the same classification layer. Figure 2 and 3 shows the structure of our classification layer. Because all three datasets used for testing are five character CAPTCHAs, the classification layer has five branches, each for one character. The structure of each branch of the classification layer is constructed by Dense(64) → Drop → Dense(36) for datasets with only lower case letters and numbers, and Dense(64) → Drop → Dense(62) for datasets with both upper case and lower case letters and numbers.

4. EXPERIMENTAL DATASETS, SETUP, AND RESULTS

4.1. DATASETS AND SETUP
For this paper, three different five character CAPTCHA datasets were used. Our first dataset, Dataset 1, is composed of all 26 English lower case letters and 10 digits randomly without slant. However, the first dataset does contain character adhesions, blurry parts, distraction lines, and differently shaded parts in the background. The first dataset has an input size of 50*200. It was downloaded from Kaggle. Our second dataset, Dataset 2, is composed of handwritten all 26 English lower case letters and upper case letters and 10 digits randomly without slant, character adhesion, or any other kind of noise. The second dataset has an input size of 28*140, and we created it ourselves through combining individual characters from the mnist and
emnist datasets. Our third dataset, Dataset 3, is composed of all 26 English upper and lower case letters and 10 digits randomly, and the third dataset contains slant, character adhesions, and dots and lines in the background for distractions. It has an input size of 80*190 and was downloaded from Kaggle. 1070 samples were selected from each Dataset for training. We used the Windows 10 operating system, Intel(R) Core(TM) i7-8700, GTX 1050 Ti, and all experiments were completed on Keras. The images were read as grayscale using OpenCV with their respective input sizes. All the networks are trained using RMSprop. Each model was trained 150 epochs for each dataset with batch size 1.

![Example from Dataset 1](2wc38) ![Example from Dataset 2](0A9Vt) ![Example from Dataset 3](0NemM)

4.2. RESULTS

Table 1 showed the accuracy of our models after 150 epochs of training. All three models were able to accurately recognize the CAPTCHAs from the first and second dataset, with a lowest accuracy of 88.6% and a highest accuracy of 98.8%. The third dataset was a challenge and all three models performed fairly poorly on it after 150 epochs of training, only achieving a highest accuracy of 73.99%, but the accuracy might increase with more training. We also observed that when trained on the second dataset, network 2 sees a sudden drop in accuracy between epochs 140 and 150. In this experiment, network 2’s accuracy on the second dataset dropped from 98.04% at epoch 143 to 0% at epoch 145. The reason that this occurs was because starting at epoch 144, the loss value given by the loss function was NaN. It is uncertain why this occurred. The loss function we used was CategoricalCrossentropy.

![Figure 1. The structure of Network 1.](image1)

![Figure 2. The structure of the classification layer for lower case letters and digits only.](image2)

![Figure 3. The structure of the classification layer for both lower case and upper case letters and digits.](image3)

| Output Shape: 36 |
|------------------|
| Output Shape: 62 |

Table 1. Accuracy and Output Shape for Each Network on Each Dataset.

*Output Shape is before flatten
*The Network 2 Accuracy Dataset 2 recorded is its accuracy at epoch 143
5. IMAGE PROCESSING

5.1. FOURIER TRANSFORM
To improve the accuracy and efficiency of our models, we also experimented with image processing. If preprocessed CAPTCHAs with all the noise removed were used, the identification process should be
simplified. The main method we experimented with was performing Fourier Transform on the CAPTCHA using Numpy. However, the result of Fourier Transform on CAPTCHAs was far from ideal. Most of the noise are similar to the actual characters, meaning Fourier Transform could not be used to actually remove them.

Result of Fourier Transform on an Example Image (XjGIB) from Dataset 3

5.2. BLURRING AND DEBLURRING
One other approach we experimented with was blurring and deblurring. We observed that normally, to make it easier for the human user to recognize them, the actual characters have higher resolution or are in heavier strokes than the noise in order for them to stand out. From this observation, we proposed that after blurring the image by a certain amount, only the characters remain recognizable. After that, deblurring the blurred image could produce an end result with only the characters. To test this idea out, we used the gaussian filter to perform the blurring part, after which the image will be deblurred using Richardson-Lucy Deconvolution with 30 iterations. Scipy ndimage was used to perform the gaussian filter, and Scikit-Image was used to perform the Richardson-Lucy Deconvolution. The processed image has slightly less noise than the original image, but the characters have lower resolution and are therefore less recognizable. Overall, this type of image processing did not have an impact on the accuracy of the models.

An Example Image from Dataset 3 (0G0WI) After the Blurring and Deblurring Process

6. CONCLUSION
Even though there are many kinds of CAPTCHAs, text-based CAPTCHAs are commonly used even if they are not the most secure option because they are cheap, convenient, and user friendly. Because text-based CAPTCHAs are more vulnerable to attacks and less secure than originally intended, improvements need to be made. Finding ways to solve text-based CAPTCHAs that are more efficient and more accurate is a great way to increase their security by finding their weaknesses. Overall, CNN is an efficient and accurate method of recognizing CAPTCHAs, and there could be more developments in using it to improve the security of text-based CAPTCHAs. To determine if CNN networks which have less structural complexity and shorter runtimes could be used to accurately recognize CAPTCHAs with character adhesions and background noise, our team constructed three CNN networks that are structurally more efficient than many of the current methods of high accuracy CAPTCHA recognition, and tested them on three different CAPTCHA datasets. The results show that even though these networks have less structural complexity, they are still capable of reaching high accuracy recognition, such as 94.67% accuracy for Network 1 on the first dataset, with only 1070 samples from each dataset selected for training. Even on instances where the accuracy is low, we believe the accuracy could be greatly improved with more training. These results suggested that the third dataset, which is composed of all 26 upper case and lower case letters and 10 digits randomly with character adhesions, slant, and dots and lines in the background as distractions, was considerably harder for the models to recognize compared to the other two datasets, and it required more training to reach a desired accuracy. The future security of text-based CAPTCHAs remains an open problem and is part of our ongoing work.
References

[1] Bursztein, E., Bethard, S., Mitchell, J.C., Jurafsky, D., Fabry, C. (2010) How good are humans at solving CAPTCHAs? a large scale evaluation. IEEE S&P ’10.
[2] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE.
[3] Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V. (2014) Multi-digit number recognition from street view imagery using deep convolutional neural networks. ICLR.
[4] Zhang, L., Huang, S.G., Shi, Z.X., Hu, R.G. (2011) CAPTCHA recognition method based on LSTM RNN. Pattern Recogn., 1: 40–47.
[5] Yan, J., Ahmad, A.S.E. (2008) A low-cost attack on a Microsoft CAPTCHA. Proceedings of the ACM Conference on Computer and Communications Security, 543–554.
[6] Stark, F., Hazirbas, C., Triebel, R., Cremers, D. (2015) Captcha recognition with active deep learning. GCPR Workshop on New Challenges in Neural Computation.
[7] Wang, J., Qin, J.H., Xiang, X.Y., Tan, Y., Pan, N. (2019) CAPTCHA recognition based on deep convolutional neural network. Math. Biosci. Eng., 16: 5851–5861.
[8] Kaggle. (2018) CAPTCHA Images. https://www.kaggle.com/fournierp/captcha-version-2-images.
[9] Kaggle. (2020) CaptchalImages1070. https://www.kaggle.com/datascientistsohail/captchalimages1070.