CAPTCHA Recognition with Active Deep Learning

Caner Hazırbaş
Apple Inc.
15 PUBLICATIONS **2,183** CITATIONS

Daniel Cremers
Technische Universität München
478 PUBLICATIONS **26,427** CITATIONS

CAPTCHA Recognition with Active Deep Learning

Fabian Stark, Caner Hazırbaş, Rudolph Triebel, and Daniel Cremers

Technical University of Munich, Germany
{fabian.stark,c.hazirbas,rudolph.triebel,cremers}@in.tum.de

Abstract. CAPTCHAs are automated tests to tell computers and humans apart. They are designed to be easily solvable by humans, but unsolvable by machines. With Convolutional Neural Networks these tests can also be solved automatically. However, the strength of CNNs relies on the training data that the classifier is learnt on and especially on the size of the training set. Hence, it is intractable to solve the problem with CNNs in case of insufficient training data. We propose an Active Deep Learning strategy that makes use of the ability to gain new training data for free without any human intervention which is possible in the special case of CAPTCHAs. We discuss how to choose the new samples to re-train the network and present results on an auto-generated CAPTCHA dataset. Our approach dramatically improves the performance of the network if we initially have only few labeled training data.

1 Introduction

A CAPTCHA [1] (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part) is an automated test to identify whether the user is a human or not. CAPTCHAs are often used on the internet to prevent automated programs from abusing online services. Nowadays, most service providers such as email or online shopping sites require users to solve CAPTCHAs, which most often consist of some distorted text that must be read and typed in correctly. For humans this is a comparably simple task, but computers still have difficulties here. Useful CAPTCHAs should be solvable by humans at least 80% of the times while programs using reasonable resources should succeed in less than 0.01% of the cases [4]. An example of a CAPTCHA is shown in Fig. 1.

Recently, researchers have started investigating automated methods to solve CAPTCHAs. Many of these existing solutions first perform character segmentation and then recognition. However, they can not solve newer, more challenging CAPTCHAs, where the letters are skewed so that they can not be separated by vertical lines. Thus, rectangular windows can not be used for segmentation, and more powerful classification methods are needed. One successful approach proposed recently by Goodfellow *et al.* [9] uses a deep Convolutional Neural Network (CNN), a framework that is also used in many other tasks such as object classification [5, 6], automatic speech recognition [8, 10] or natural language processing [3, 7]. However, a major requirement for training a deep CNN is a very large training data set (for example the ImageNet [15] data for image classification),

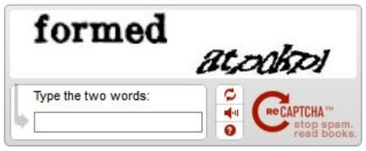


Fig. 1. Google's reCAPTCHAs [2] consist of a distorted word and a word scanned from a text book. The user must type both words, thereby helping to digitize printed books.

and for `CAPTCHA` recognition, there is usually only a small annotated training data set available. Furthermore, the appearance of `CAPTCHA`s can, in contrast to objects in natural images, often change significantly from those in the training data, e.g. due to major changes in the distortion applied to the text.

To address these problems, we propose in this paper an approach that is based on Active Learning. The idea here is to start learning with a comparably small training set and to add new training samples in every subsequent learning round. The decision whether to add a sample to the training data is based on the uncertainty that the classifier associates with a given prediction. Under the assumption that this uncertainty estimation is well calibrated, the algorithm selects the most informative samples to learn from, resulting in less training samples required than in standard passive learning. As a further advantage, the algorithm can adapt to new input data that differs in appearance from the current training data. We note however that our problem is different to other Active Learning settings in that we do not need a human supervisor to acquire the ground truth labels for training. Instead, we use the return value that we obtain automatically when solving a `CAPTCHA`. Thus, if the classifier is able to solve a `CAPTCHA` correctly we can use that information for re-training, because then the ground truth label is known. Of course, if the `CAPTCHA` is not solved we don't have that information, but we will show how learning can be done from the correctly predicted samples only. In summary, we present three novel contributions: First, we show how to compute uncertainty from a deep CNN and how this relates to correct classification. Second, we peform Active Learning with a deep CNN. And third, we show that already the correct, but uncertain classified samples are enough for efficient learning, with the effect that we need only little training data, and this is obtained without any human intervention.

2 Related Work

Conventional methods aim at detecting the text within natural images in two disjoint steps [11]: localizing the regions of words or single characters within the image, segmenting [17] and then recognizing them [19]. In addition, a dictionary can be used to dismiss unlikely words. For example, Mori and Malik [18] proposed a method to solve `CAPTCHA`s using a dictionary with all 411 words that the considered `CAPTCHA`s contain. Chellapilla and Simard [4] also solve `CAPTCHA`s

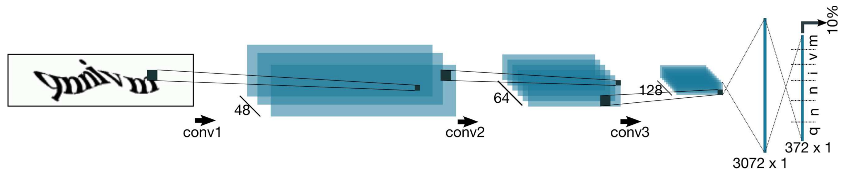


Fig. 2. Convolutional Neural Network for CAPTCHA Recognition. Our CNN is composed of three convolution, three pooling and two fully-connected layers. The last layer outputs the probability distributions for all digits for which we can compute the prediction and uncertainty of the prediction.

by segmenting single characters and recognizing them, but without a dictionary. However, in modern CAPTCHAs, single characters can not be segmented easily with rectangular windows, as the characters can overlap each other (see Fig. 1). These CAPTCHAs are more similar to hand-written text, and LeCun *et al.* [16] proposed to use Convolutional Neural Networks (CNN) for recognition of hand-written digits. These CNNs are designed to construct the hierarchy of the objects layer by layer and perform classification. In 2014, Goodfellow *et al.* [9] proposed to combine localization, segmentation and recognition of multi-character text using deep CNNs. Training is done on millions of images using a cluster of several computers. Jaderberg *et al.* [12] proposed a CNN for text recognition on natural scene images. However, for training they artificially create a very large set of text images. In contrast, we use a much smaller training set. By exploiting Active Learning, we fine-tune the network during runtime, and our network is fed with correctly classified but highly uncertain test samples.

3 A Deep CNN for CAPTCHA Recognition

We propose a deep CNN to solve the whole sequence of a CAPTCHA. Our purpose is to recognize the full sequence without pre-segmentation. We use the network structure shown in Fig. 2. We focus on CAPTCHAs with 6 digits. Each digit is represented by 62 neurons in the output layer. We define a bijection $\Theta(x)$ that maps a character $x \in \{$‘0’,...‘9’, ‘A’,..., ‘Z’, ‘a’,..., ‘z’$\}$ to an integer $l \in \{0, ...61\}$:

$$\Theta(x) = \begin{cases} 0 \dots 9, & \text{if x = ‘0’} \dots \text{’9’} \\ 10 \dots 35, & \text{if x = ‘A’} \dots \text{’Z’} \\ 36 \dots 61, & \text{if x = ‘a’} \dots \text{’z’} \end{cases} . \qquad (1)$$

We assign the first 62 output neurons to the first digit of the sequence, the second 62 neurons to the second digit and so on. Thus, for a digit x_i the neuron index n is computed as $n = i \cdot 62 + \Theta(x_i)$, where $i \in \{0, ..., 5\}$ is the index of the digit, i.e. the output layer has $6 \cdot 62 = 372$ neurons. To predict a digit, we consider the corresponding 62 neurons and normalize their sum to 1. Fig. 4 shows an example of a network output. Here, the predicted character index for the first digit is $c_0 = 52$ and the predicted label is $x = \Theta^{-1}(c_0) =$ ‘q’.

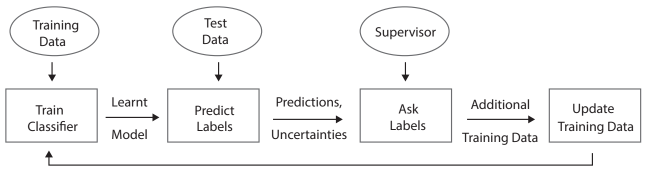


Fig. 3. Flow chart of Active Learning. We start with training on a small data set. Then, the classifier is applied to some new data, resulting in a label prediction and an associated uncertainty. From this uncertainty, the classifier decides whether to ask for a ground truth label or not. In our case, this query is done by solving the given CAPTCHA with the prediction and using it if it was correct. Then, the training data is increased and learning is performed again. In our method, we use a deep CNN, which can be efficiently re-trained using the newly added training samples.

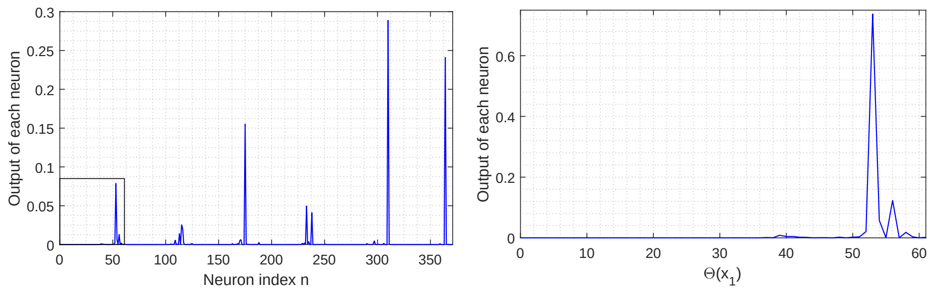


Fig. 4. Example output of the network for the CAPTCHA "qnnivm" in Fig. 2. **Left:** There are 62 outputs for each digit. The black box shows the output for the first digit. **Right:** Probability distribution for the first digit. The sum is normalized to 1.

4 Active Learning to Reduce the Required Training Data

For a good classification accuracy, CNNs usually require a very large training set. However, collecting millions of hand-labeled CAPTCHAs is infeasible. Therefore, we propose to use Active Learning (see Fig. 3). The main idea of this is to add new training data only if necessary, i.e. if the sample is informative enough for re-learning. This is decided based on the uncertainty of the prediction, which we compute using the best-versus-second-best strategy [14], as described next.

4.1 Obtaining the uncertainty

As mentioned above, we estimate the predictive distribution of each digit by normalizing the sum of the corresponding network outputs to 1. From this we compute the overall uncertainty η using "best-vs-second-best" as

$$\eta = \frac{1}{d} \cdot \sum_{i=1}^{d} \frac{\arg\max \{\mathcal{P}(x_i) \setminus \arg\max \mathcal{P}(x_i)\}}{\arg\max \mathcal{P}(x_i)}, \tag{2}$$

where $\mathcal{P}(x_i)$ is the set of all network outputs for digit d_i. Thus we divide the second best by the best prediction for every digit.

4.2 Querying Ground Truth Information

Our CAPTCHA recognition problem is unique in the sense that we can perform learning without human intervention. We achieve this by only using those data samples for re-training, for which the classifier already provided a correct label. For these, the CAPTCHA can be solved and we know what the correct text is. However, simply using all these correctly classified samples for re-training would be very inefficient. In fact, training would be done more and more often, because the classifier will be better over time and therefore classify more samples correctly. Thus, with every new correctly classified sample a retraining would be necessary. To avoid this, we use the uncertainty values presented above: We sort the correctly classified test samples in each learning round by prediction uncertainty and use only the most uncertain ones for re-training. This results in a lower number of required training samples, but as we will show in the experiments, the most uncertain samples are also the most informative for learning.

5 Experimental Evaluation

We present the results of our approach on auto-generated CAPTCHAs. All experiments have been executed using the Caffe [13] deep learning framework on an NVIDIA GeForce® GTC™ 750 Ti GPU.

5.1 Dataset Generation

As there is no hand-labeled CAPTCHA dataset, we use our own scripts to generate CAPTCHAs. During the auto-generation, we ensure that there is no duplication in the dataset.

We use the **Cool PHP CAPTCHA** framework to generate CAPTCHAs. They are composed of distorted text with a fixed length of 6 similar to Google's reCAPTCHA. They have a size of 180×50. We have modified the framework to generate black and white images. Furthermore we have disabled shadows and the line through the text. We also do not use dictionary words, but random characters. Therefore we have removed the rule that every second character has to be a vowel. We fix the font to "AntykwaBold". Fig. 5 shows some examples of our auto-generated CAPTCHAs.

5.2 Network Design

We use the network illustrated in Fig. 2. The convolutional layers have a size of 48, 64 and 128. They all have a kernel size of 5×5 and a padding size of 2. The pooling layers have a window size of 2×2. The first and third pooling layers and also the first convolutional layer have a stride of 2. Then the network has

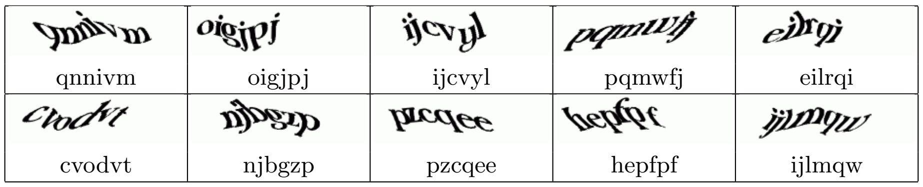


Fig. 5. Example CAPTCHAs used in the experiments.

one fully connected layer with a size of 3072 and a second fully connected layer (classifier) that has an output size of 372. We also add rectified linear units and Dropout after every convolutional and the first fully connected layer. The batch size for every iteration is 64.

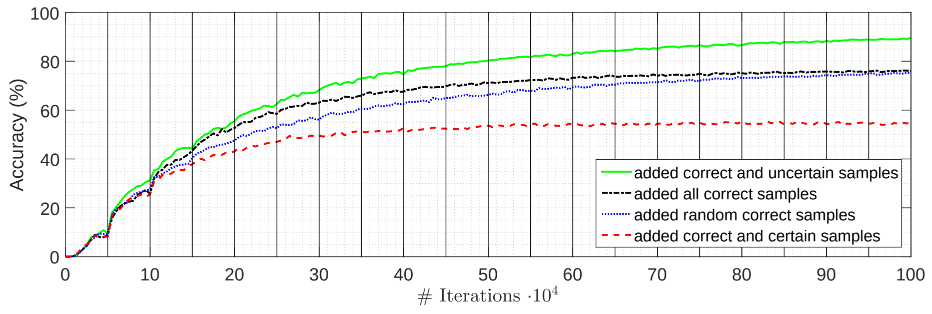
5.3 Qualitative Evaluation

We train the network with the SGD algorithm. However, in contrast to other methods we train the network for all digits independently. The learning rate changes by the rule $\alpha = \alpha_0 \cdot (1+\gamma \cdot t)^{-\beta}$ where the base learning rate $\alpha_0 = 10^{-2}$, $\beta = 0.75$, $\gamma = 10^{-4}$ and t is the current number of iteration. We set *momentum* $\mu = 0.9$ and *regularization parameter* λ is $5 \cdot 10^{-4}$.

As the most expensive part is to get the training samples, our approach aims at decreasing the required size of the initial training set. So we first of all train our network with a very small initial training set of 10^4 images for $5 \cdot 10^4$ iterations. We only achieve an accuracy of 9.6% which even decreases with more iterations. Because of that, we want to make use of Active Learning.

First of all, we again train our network with 10^4 training images for $5 \cdot 10^4$ iterations. Afterwards, we classify $5 \cdot 10^4$ test images. Then, we pick new training samples from the correctly classified ones. We can take all of them, or we only pick $5 \cdot 10^3$ samples based on their uncertainty: Either with the highest uncertainty, the lowest uncertainty, or randomly. Uncertainty is computed as described in Section 4.1. Once the new selected samples are added to the training set, we re-train the network for $5 \cdot 10^4$ iterations. Subsequently we follow the same procedure. We apply this algorithm for in total 20 Active Learning rounds (*epochs*). The accuracy is computed after every $5 \cdot 10^3$ iterations on a fixed validation set. We get the best performance with the correct but uncertain predictions (see top plot in Fig. 6). All results are the average out of two runs.

However increasing the number of samples in the training set requires more storage. Moreover, one should increase the number of iterations to benefit more from the cumulated set which will cause longer training time. For all these reasons, we suggest to use only the selected samples at each iteration to re-train the network. Therefore we again train with 10^4 initial training images for $5 \cdot 10^4$ iterations. Then we classify 10^5 test images and replace the training set with 10^4 of the correct classified ones and train for $2.5 \cdot 10^5$ iterations again. Subsequently we follow the same procedure and decrease the number of iterations after each epoch according to the following rule: $2.5 \cdot 10^4$ iterations until epoch 6, $2 \cdot 10^4$ until



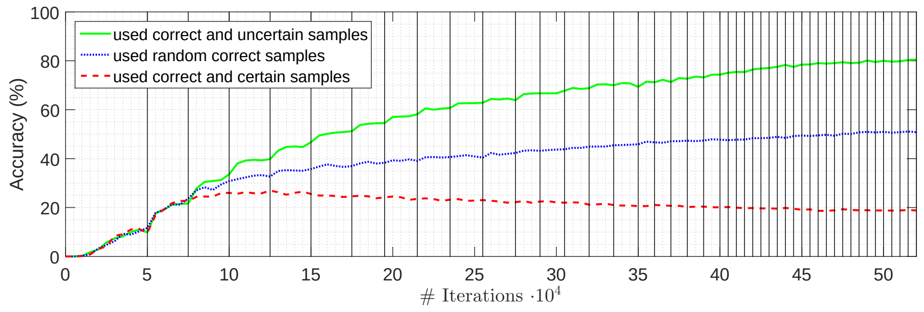


Fig. 6. Learning curves for Active Deep Learning. **Top:** The training set is increased with the selected samples after each iteration. When using all the correct ones (black curve), we stop adding new images to the training set after $50 \cdot 10^4$ iterations, because the size of the training set already exceeds $3 \cdot 10^6$. **Bottom:** Network is retrained only on the new samples. Vertical black lines denote the end of every Active Learning epoch.

epoch 11, $1.5 \cdot 10^4$ until epoch 16, $1 \cdot 10^4$ until epoch 21 and $5 \cdot 10^3$ until epoch 40. We again get the best performance with the correct but uncertain predictions (see bottom plot in Fig. 6). This is reasonable as the network in fact classifies the images correctly, but still is very uncertain about the prediction. Hence it can learn from the fact that it was indeed right with its classification. One can argue that learning with misclassified samples should yield better results. This is indeed the case, however not possible in practice.

6 Conclusion

We propose a CAPTCHA solving technique that uses initially a very small set of images to train a deep CNN and then improves the classifier by exploiting the test samples. New training samples are chosen from the test set based on their uncertainty. Our results show that the performance of the network can be significantly improved with the correctly classified but uncertain test samples.

Acknowlegments The work in this paper was partly funded by the EU project SPENCER (ICT-2011-600877).

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security. In: EUROCRYPT (2003)
2. von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: recaptcha: Human-based character recognition via web security measures. Science (2008)
3. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. The Journal of Machine Learning Research 3, 1137–1155 (2003)
4. Chellapilla, K., Simard, P.Y.: Using machine learning to break visual human interaction proofs (hips). In: NIPS (2004)
5. Claudiu Ciresan, D., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep big simple neural nets excel on handwritten digit recognition. arXiv preprint arXiv:1003.0358 (2010)
6. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Int. Conf. on Artificial Intell. and Statistics. pp. 215–223 (2011)
7. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning. pp. 160–167. ACM (2008)
8. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. Audio, Speech, and Language Processing, IEEE Transactions on 20(1), 30–42 (2012)
9. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. ICLR (2014)
10. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. Signal Processing Magazine, IEEE 29(6), 82–97 (2012)
11. Jaderberg, M., Vedaldi, A., Zisserman, A.: Deep features for text spotting. In: CVPR (2014)
12. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. IJCV (2015)
13. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
14. Joshi, A., Porikli, F., Papanikolopoulos, N.: Multi-class active learning for image classification (2009)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
16. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998)
17. Lu, Y.: Machine printed character segmentation; an overview. Pattern Recognition 28(1), 67–80 (1995)
18. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: breaking a visual captcha (2003)
19. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: 2013 12th International Conference on Document Analysis and Recognition. vol. 2, pp. 958–958. IEEE Computer Society (2003)