2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS) | 978-1-6654-0521-8/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICACCS51430.2021.9441737

Contour Based Deep Learning Engine to Solve CAPTCHA

Ashwani Kumar
Netaji Subhas Institute of Technology, University of Delhi
Delhi, India
ashwanisinghnet@gmail.com
0000-0002-5997-6732

Aditya Pratap Singh
Netaji Subhas Institute of Technology, University of Delhi
Delhi, India
pratapaditya1997@gmail.com
0000-0003-4105-7241

***Abstract*—A 'Completely Automated Public Turing test to tell Computers and Humans Apart' or better known as CAPTCHA is a image based test used to determine the authenticity of a user (ie. whether the user is human or not). In today's world, almost all the web services, such as online shopping sites, require users to solve CAPTCHAs that must be read and typed correctly. The challenge is that recognizing the CAPTCHAs is a relatively easy task for humans, but it is still hard to solve for computers. Ideally, a well-designed CAPTCHA should be solvable by humans at least 90% of the time, while programs using appropriate resources should succeed in less than 0.01% of the cases. In this paper, a deep neural network architecture is presented to extract text from CAPTCHA images on various platforms. The central theme of the paper is to develop an efficient & intelligent model that converts image-based CAPTCHA to text. We used convolutional neural network based architecture design instead of the traditional methods of CAPTCHA detection using image processing segmentation modules. The model consists of seven layers to efficiently correlate image features to the output character sequence. We tried a wide variety of configurations, including various loss and activation functions. We generated our own images database and the efficacy of our model was proven by the accuracy levels of 99.7%.**

***Index Terms*—CAPTCHA, Text Image Recognition, Convolutional Neural Networks, Image Contour, Deep Learning Engine**

I. Introduction

A CAPTCHA is an automated test to identify the authenticity of the user. Captchas are used on the internet to prevent bot programs from exploiting online services. Today, most online services mandate users to solve them, mostly consisting of some distorted text that must be read and mistyped. The motivation is to develop a system that provides much better accuracy. This project aims to build a complete Captcha pipeline application with the latest algorithms, libraries, and frameworks and use Convolutional Neural Networks for smart recognition. Text segmentation from a document image is a challenging task. Choosing correct parameters for efficient results in various steps of pipeline development requires graph plotting for minimizing errors. Furthermore, it sometimes requires rigorous simulation of algorithms and the model.

The paper is organized in the following order: we review the literature for CAPTCHA, Convolutional Neural Networks, and discuss some of the related works in Section 2. Section 3 discusses our design and the architecture of our proposed solution. Sample output of the model coupled with the relevant model statistics are presented in Section 4, followed by the Future Scope and Conclusions in Section 5 and section 6 respectively.

II. Literature Review

A. CAPTCHA

The acronym captcha was coined by Ahn, J et al. [1]. Primarily, hard and unsolved AI problems are used as CAPTCHAs for the simple reason that either the problem will be solved or we will be able to differentiate between humans and bots. For this reason, many of the researchers in the field of Computer Science are interested in solving this problem. Any ideal CAPTCHA program should generate a code that is easily solvable by humans but not with the automated computer programs. There are various schemes of CAPTCHAs, such as Text-based, Sound-based, and Image-based schemes. Recent work has been done to prevent bots by using anti-recognition and anti-segmentation patterns like a varied set of different alphabet, multiple fonts, varying string lengths, distorted or rotated image, slant, slope, or scaling in the image.

B. CNN(Convolutional Neural Network)

A Convolutional Neural Network (CNN) comprises one or more convolutional layers with the corresponding pooling layers. The architecture of a CNN is designed in such a way so as to take full advantage of the bi-dimensional structure of images. An important benefit of CNNs is that they are comparatively easier to train and have lesser parameters. To train CNNs, a technique called backpropagation is majorly used. Here, a set of input and output values are constructed and this input data is now provided to the first layer (ie. input layer) of the neural network. The output of n^{th} layer is provided as the input to the $n + 1^{th}$ layer. The output of the network is then compared with the actual output which is used to further fine tune the internal parameters.

Training a CNN is essentially similar to solving an minimization problem. This minimization function in the context of a CNN is called a loss function measuring the deviation between the actual and the predicted values. There are various types of loss functions such as cross-entropy, categorical cross-entropy,

978-1-6654-0521-8/21/$31.00 ©2021 IEEE

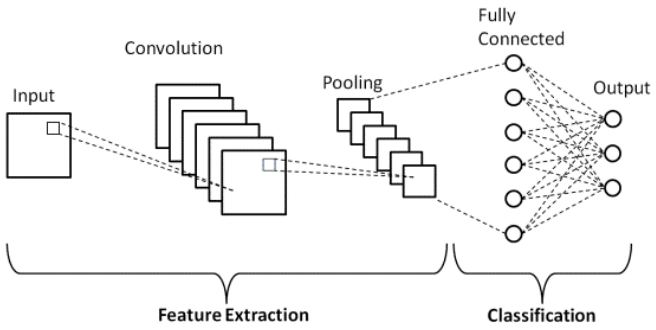

Fig. 1. Basic model of a convolutional neural network.

Fig. 2. Sample training data

hinge loss, and mean squared error. At each convolution layer, inputs are fed into a non-linear function known as the activation function. Various activation functions are used such as Sigmoid, Softmax, and ReLU.

C. Related Works

Øivind Due Trier et al. [14] presents feature extraction methods for recognition of segmented characters. Y. Le Cun et al. [13] present an application of backpropagation networks to handwritten digit recognition. The paper goes on to show that the large backpropagation networks can be applied to real image recognition problems. Hasan, W [8] explores recent work that has been done on different kind of CAPTCHA methods and their categories and discusses their strengths and weakness.

S. Sivakorn et al. [11] created a web-browser based system to solve image CAPTCHAs using the Google Reverse Image Search to label the images and then finds similar images based on image annotation, achieving an 83% success rate on similar images data

Kwon, Hyun et al. [9] have used style transfer based CNN model to achieve a better result. However, there are many other approaches which are independent of convolutional neural networks, such as Wang, Ye et al. [12], who are relying on classical image processing methods. Also, Elie Bursztein et al. [6] uses a sliding window approach to segment the characters and recognize them individually.

III. Proposed Solution

A. Data Synthesis and Labelling

We needed good quality training data, that looks like Figure 2, to train our CNN model. For this, we reverse-engineered the 'Really Simple CAPTCHA' library from the WordPress Plugin

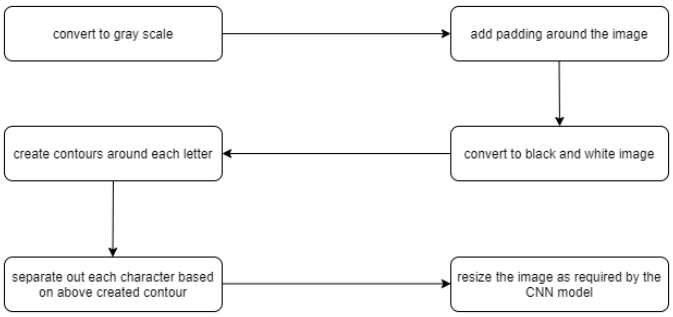

Fig. 3. Image pre-processing pipeline

directory [4]. We modified it to save out 10,000 CAPTCHA images along with the expected answer for each image. Each image underwent a rigorous transformation (Figure 3) - they are converted to grayscale, and padding is added so that there is sufficient space for the contour to be created correctly and the characters are not entangled amongst themselves. Letter contours are created around each of the characters in the image, and these characters are segmented based on these contours. We used a linear time component-labeling algorithm for contour tracing [7]. Finally, we resize the image to a fixed size as desired by our neural network model.

B. Our Approach

1) CNN Architecture: We have created a multi-layered graph network of convolutional layers with a ReLU activation function succeeded by the corresponding max-pooling layers. In the first layer, we apply a convolutional 2D layer on a 20×20 image pixel set. After the applications of all the convolutional layers, we pass the multi-dimensional intermediate output to the flatten layer, which flattens it to a linear array of values. This array is enormous in size, so it undergoes the two dense layers where it is finally converted to a 32 sized output array. Each convolutional layer has its loss function & activation function to optimize the corresponding layer parameters in each backpropagation phase, which are not included in the below architecture diagram for brevity. These are explained later in the section alongside Adam Optimizer. Refer to Figure 4 for the architecture

2) Loss Function: We implemented categorical cross-entropy as the loss function, which is majorly used for single label categorization. Categorical cross-entropy compares the distribution of the model predictions with the actual distribution. It is applicable when only one category is applicable to a single data point. In essence, an example can belong to one class only. Mathematically, it can be expressed as -

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N} y_{ij} * \log \hat{y}_{ij} \quad (1)$$

where $\hat{y}$ is the predicted value.

3) Optimizer: We are using Adam (Adaptive Moment Estimation) Optimizer, a replacement optimization algorithm for efficient stochastic gradient descent with little memory

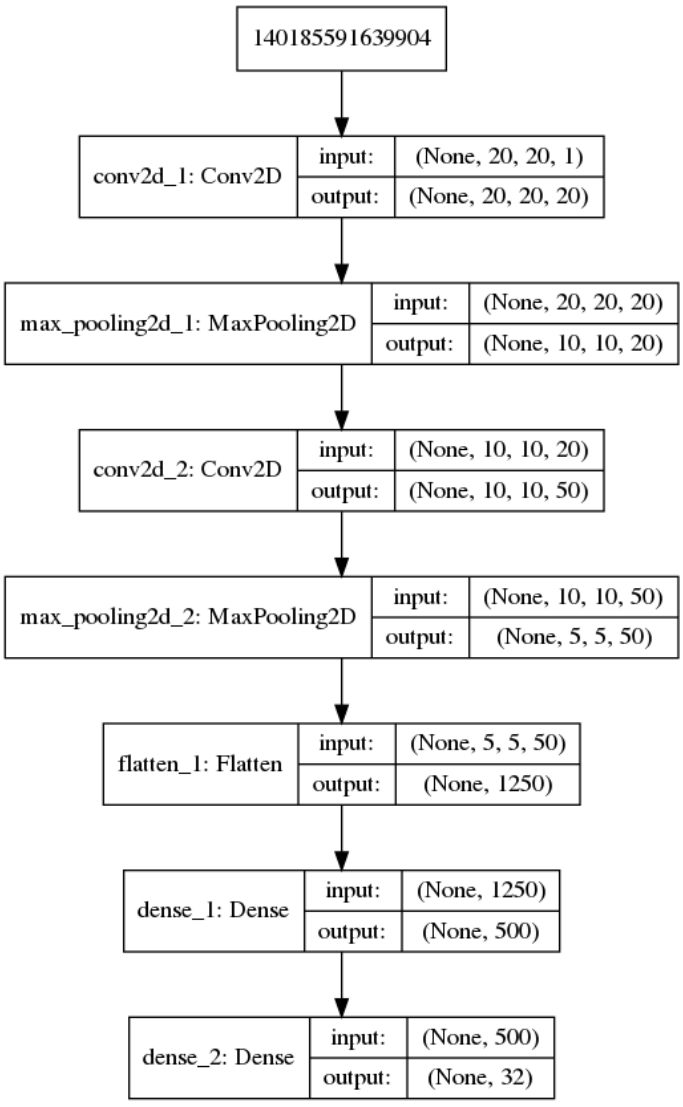

Fig. 4. Proposed convolutional neural network architecture

requirements. Adam combines the best properties of AdaGrad and RMSProp algorithms. The step-size hyper-parameter approximately bounds its step-sizes, and it does not require a stationary objective.

4) Activation Function: We compared various activation functions in our model to gauge their performance. As seen in Table I, ReLU performed much better than Sigmoid and Softmax activation functions. The ReLU (rectified linear activation function) is a piece-wise linear function whose output is the maximum of zero and the input value. Mathematically expressed

$$R(z) = max(0, z) \tag{2}$$

C. Workflow

The general workflow (see Figure 5) starts with uploading the image of a given CAPTCHA code via some interface, be it a website or an API, which is then sent to the server

Metrics	Sigmoid	Softmax	ReLU
Loss	0.162	0.091	0.011
Accuracy	98.7%	99.1%	99.7%

TABLE I
COMPARISON OF VARIOUS ACTIVATION FUNCTIONS

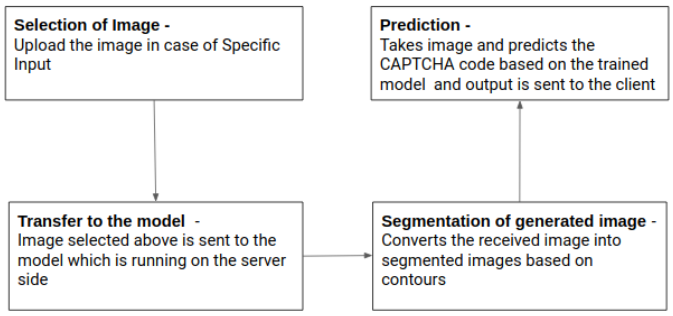

Fig. 5. Client-Server workflow design

hosting the model's pickle file. The given image is processed as described in section 3.1 and the model predicts the output, which is finally sent back to the client-side.

IV. Results

A. Specifications

The source code is available under the open-source MIT license for Linux, Windows, and BSD-like platforms. Figure IV-C shows the model's performance as seen on a Linux based machine with a 2.4Ghz dual-core 64-bit Intel processor with 8GB of DDR4 RAM and enterprise-class Western Digital SCSI hard disks, running 64-bit Linux Ubuntu. The implementation is Python 3.6 based, using Tensorflow & Keras for the creation of convolutional neural networks. Python-based framework - Flask is used to host the model's pickle file.

B. Result Image

The sample result image is shown in Figure 6

C. Model Statistics

The accuracy graph (refer Figure 7) shows a constant increase in accuracy as the iterations are increasing, and we can see the best results at the 6th iteration with an accuracy of 99.7% and minimum loss. On further iterations, we notice that the model begins to overfit on the training data, which is not desired.

V. Future Works

The speed of the pipeline is slow. It does heavy image processing that uses many iterations over images for edge detection, contour detection, and letter detection. We can overcome speed issues by implementing the image processing modules in cython. This model's scaling is not viable because Keras Deep learning CNN takes time and a lot of computation, and it is not suited for real-time applications. Our model does

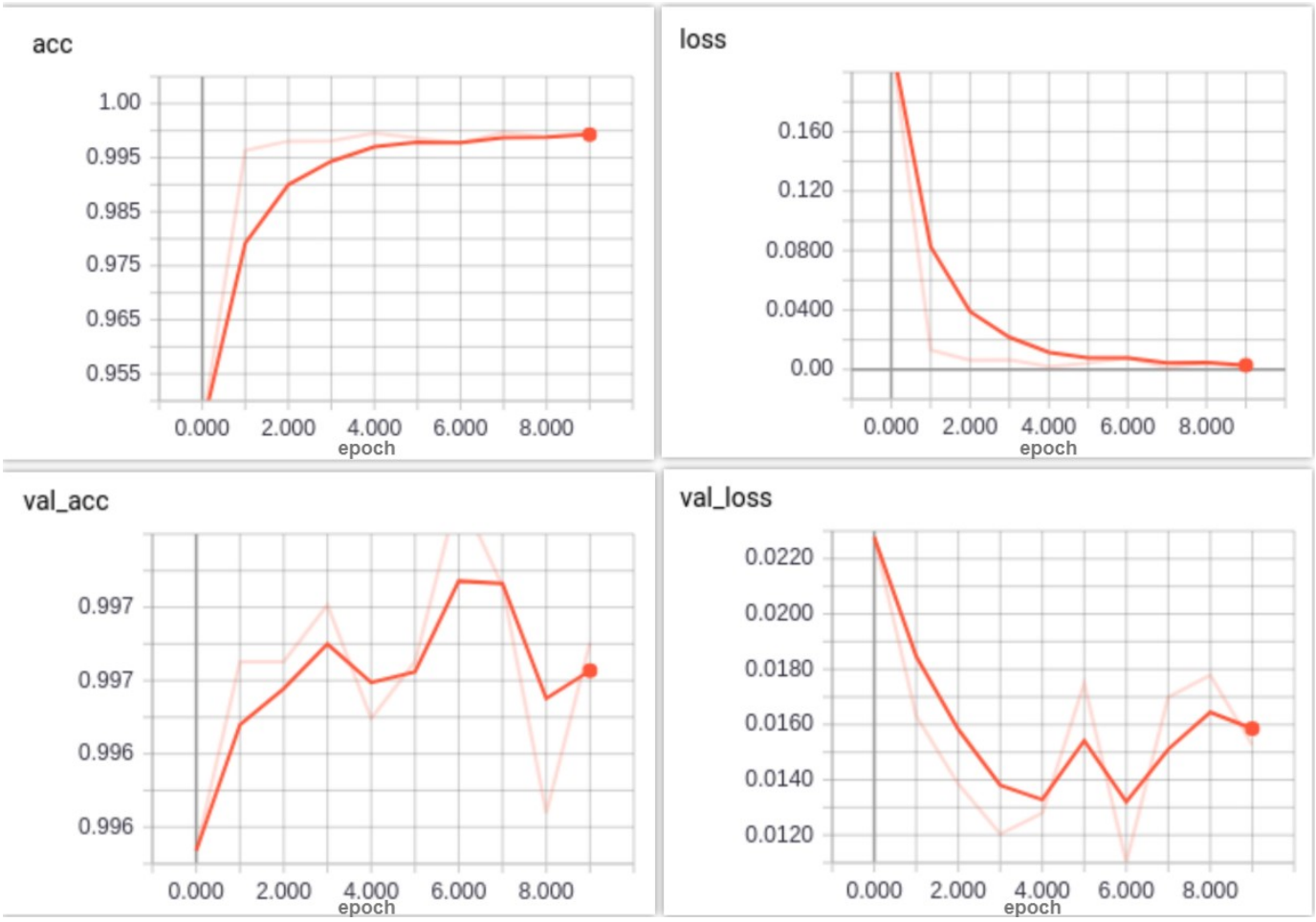

Fig. 7. Experimental metrics - for accuracy and loss

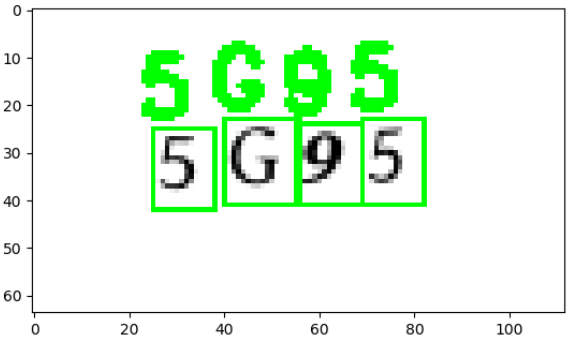

Fig. 6. Final result from the model

not handle synchronous queries. While scaling, image preprocessing steps should be done on a single machine (considering all machines have a copy of the model) because we can't split images before word segmentation. After the word segmentation, we can split queries over different machines for word prediction because our model is trained on words, but the forecast is on alphabets. If we split images before word segmentation, we may not be able to extract words accurately. Scaling of the model can be done in various ways; one of the ways is to use a distributed message queue such as Kafka in which synchronous queries are stacked and processed one by one in the order of their appearance. Data is scarce for model training. It requires a lot of person-hours to generate this kind of data. We don't have data for all varieties of CAPTCHA, and most of the dataset is paid. More computing resources like GPUs and more training data can be used further to improve the model's accuracy and robustness. Our dissertation's ideas can be extended for automating the CAPTCHA decoding from live websites where we can send the images from the source website to our server where we can host the optimized model.

VI. Conclusion

In this paper, we present a novel architecture for identifying the CAPTCHA code on various platforms. Our experimental results on different captcha codes indicate that our architecture design produces considerably good results given various limitations

References

[1] Ahn, J. (2003). CAPTCHA: Using Hard AI Problems for Security. In Advances in Cryptology — EUROCRYPT 2003 (pp. 294–311). Springer Berlin Heidelberg.
[2] Buscema, Massimo. (1998). Back Propagation Neural Networks. Substance use & misuse. 33. 233-70. 10.3109/10826089809115863.
[3] Chollet, F., & others. (2015). Keras. https://github.com/fchollet/keras.
[4] Data points, https://wordpress.org/plugins/really-simple-captcha/
[5] Diederik P. Kingma, & Jimmy Ba. (2017). Adam: A Method for Stochastic Optimization.
[6] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, & John C. Mitchell (2014). The End is Nigh: Generic Solving of Text-based CAPTCHAs. In 8th USENIX Workshop on Offensive Technologies (WOOT 14). USENIX Association.
[7] Fu Chang, Chun-Jen Chen, & Chi-Jen Lu (2004). A linear-time component-labeling algorithm using contour tracing techniqueComputer Vision and Image Understanding, 93(2), 206 - 220.
[8] Hasan, W. (2016). A Survey of Current Research on CAPTCHAInternational Journal of Computer Science & Engineering Survey, 7, 1-21.
[9] Kwon, Hyun; Yoon, Hyunsoo; Park, Ki-Woong. 2020. "CAPTCHA Image Generation: Two-Step Style-Transfer Learning in Deep Neural Networks" Sensors 20, no. 5: 1495.
[10] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, & Xiaoqiang Zheng (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265–283).
[11] S. Sivakorn, I. Polakis and A. D. Keromytis, "I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs," 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken, 2016, pp. 388-403, doi: 10.1109/EuroSP.2016.37.
[12] Wang, Ye & Lu, Mi. (2018). An Optimized System to Solve Text-Based Captcha. International Journal of Artificial Intelligence & Applications. 9. 19-36. 10.5121/ijaia.2018.9302.
[13] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. 1990. Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 396–404.
[14] Øivind Due Trier, Anil K. Jain, & Torfinn Taxt (1996). Feature extraction methods for character recognition-A surveyPattern Recognition, 29(4), 641 - 662.