Developing a Neural Network for Image Classification to Break Image Selection reCAPTCHA Program

John Funk
Eastern Connecticut State University
funkjo@my.easternct.edu

ABSTRACT

A common feature on many websites is a way to catch bots and automated scripts and prevent them from logging into accounts or making new accounts. These features are called CAPTCHAs. Prior research has shown that machine learning models are capable of learning how to classify images from a dataset used by image selection CAPTCHAs. In this research, I propose a convolutional neural network that can classify images from a dataset of traffic themed CAPTCHA images with an accuracy of 83%. The results from this research show that images with obstructions, small or distant target classes, or multiple target classes are difficult for the model to classify, proving my hypothesis. These types of images should represent the majority of images in a CAPTCHAs dataset in order to improve its security and robustness.

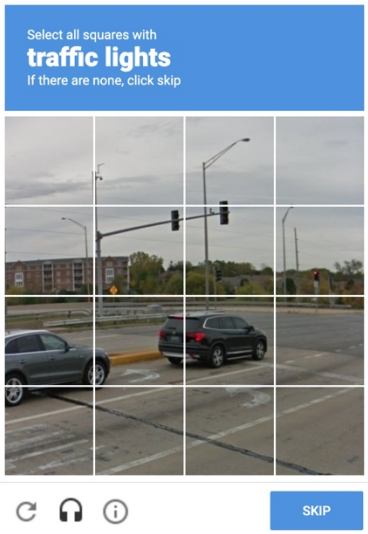

Fig. 1: An example of a modern image selection CAPTCHA program, called a reCAPTCHA. In this example, the user has to select the two images in the middle that contain traffic lights in order to proceed.

1. INTRODUCTION

As an added measure of security on many modern websites, a user is required to pass a test that proves they aren't a "robot." This security measure is intended to "prevent exploitation by bots and automated scripts" [1] and is what is called a CAPTCHA ("Completely Automated Public Turing test to tell Computers and Humans

Apart"). In 2007, research on CAPTCHAs was conducted and the result was a security measure that could filter out bots and automated scripts from using the websites service by requiring anybody attempting to access the site to correctly identify images of cats.[1] This task is very easy for humans to complete and at the time, was very difficult for computers. However, in 2008, using the power of machine learning and image classification, the CAPTCHA using the Asirra data set of cat images was exploited and a model was created that could correctly classify these images with accuracy up to 82%.[5] More modern CAPTCHAs are far more robust, but it is still important to push the limits of these security measures and see if they can be broken so that they can be improved.

Modern CAPTCHA programs are called reCAPTCHAs and come in many different forms including prompting the user to simply check off a box claiming they are not a robot, typing a series of letters in a distorted image, and selecting the correct image from a sequence of images, which usually are images of everyday road traffic, as depicted in **Figure 1**. These images are low quality and often times the target class is far in the background, not fully pictured, or are partially obstructed.

Fig. 2: Example images from the dataset that the neural network will train and test on. Among the dataset of images are images of palm trees, fire hydrants, cars, traffic lights, and crosswalks.

Attempting to break this kind of CAPTCHA requires the creation of a machine learning model for image classification. Image classification refers to the ability of a computer to recognize an image as a particular target class, such as a car. This can be achieved by acquiring a large data set of images of cars and training a Convolutional Neural Network (CNN) to identify the patterns in these images that indicate a car is present. A CNN is a type of Neural Network that is particularly good at the task of image recognition.[6] In a CNN, images are given to the network as an input in the form of a numerical matrix.[3] This data passes through each layer of the network and depending on the value in each node, only certain nodes will fire and pass data to the next layer. At

the end of the network, the model makes a prediction about the image.

The objective of this project is to attempt to create a machine learning model that can accurately classify images that are used in more modern examples of CAPTCHA programs. These images are low quality, have a high degree of noise, and the target object in the images may be small or only partially in view. The goal is to understand what sorts of images the model has difficulty classifying so that these images can be used more often in CAPTCHA programs in order to increase their security from bots and automated scripts that attempt to bypass them. My hypothesis is that the model will have difficulty classifying images where there might be multiple class objects in the image and where the class object is very small or in the background of the image.

2. METHODOLOGY

The data that will be used to train and test the model on is a dataset of over 11,000 images that are typically seen in image selection CAPTCHA programs. The dataset was downloaded from a GitHub repository here: https://github.com/deathlyface/recaptcha-dataset.

Table 1: The class distribution of each class of image in the dataset. This is not a balanced dataset because there is a slight bias towards cars and crosswalks.

Class Distribution of Dataset					
Data Set	Cars	Crosswalks	Hydrants	Traffic Lights	Palm Trees
Train	1,135	1,198	924	774	885
Test	65	62	48	37	47

The dataset contains 11 different categories of images including cars, bridges, traffic lights, crosswalks, etc. Each image is a color image and is either 120x120 pixels or 100x100. For the purposes of this project, only a subset of the data including five of the categories of images will be used for the model to train and test on. The categories of images are palm trees, fire hydrants, cars, crosswalks, and traffic lights. Examples of these images are depicted in **Figure 2.** In the dataset that will be used for this project there are 1,200 images of cars, 1,260 images of cross walks, 972 images of fire hydrants, 932 images of palm trees, and 811 images of traffic lights. Before the model can be created, each image must be resized to be the same size and converted into

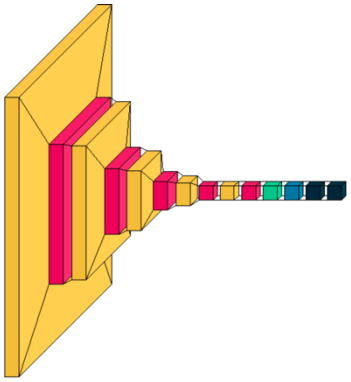

Fig 3. A visualization of the structure of the model. Each yellow layer is a convolutional layer, each red layer is a max-pooling layer, the green layer is the dropout layer, the light blue layer is the flatten layer, and the two black layers are dense fully connected layers.

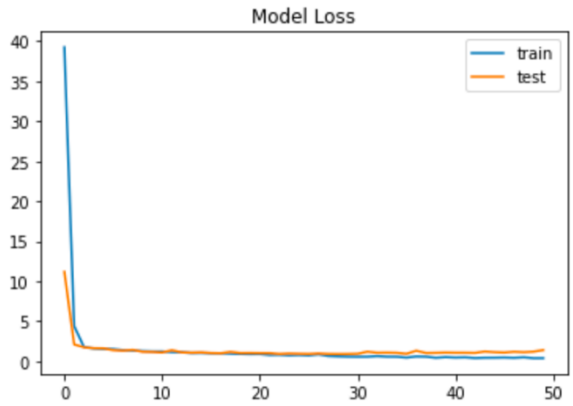

Fig 4. A visualization of the training loss and test loss per epoch. Towards the end of the training, the test loss is slightly higher than the training loss, indicating that the model is a good fit.

NumPy matrices. Every image in the dataset is converted to 100x100 pixels. Images must be converted to matrices because the neural network is created using TensorFlow, which can only take an input of matrices where each value represents each pixel in the image. Initial experimentation with the model showed that converting images to grayscale was not an effective method to improve accuracy of classification. The model's accuracy at predicting grayscale images was below 50%. As a result, converting to grayscale was not a viable option for this research.

The dataset of images is split into a training set and a test set. 95% of the images are in the training set. The neural network model's structure was created using the TensorFlow and Keras Python packages. The model's structure follows a Conv-Pool-Conv-Pool pattern. There are five convolutional layers, each of which is followed by a pooling layer. The convolutional layer is the building block of the convolutional neural network as it is where most of the learning is done because it has weights that need to be trained.[7] Each convolutional layer takes a two-dimensional input. The pooling layer that follows each convolutional layer reduces the dimensionality of the input data to the next

convolutional layer.[7] Each pooling layer in this model is a maximum pooling layer.

Following the Conv-Pool-Conv-Pool layer pattern is a single dropout layer. The dropout layer prevents nodes in the hidden layers of the model from firing. The purpose of the dropout layer is to decrease overfitting in the model. "At each training stage, individual nodes are either dropped out of the net with probability *1-p* or kept with probability *p*." [8] Finally, after the dropout layer, the resulting data is flattened and sent through a fully connected dense layer to the output layer which contains five nodes, one for each image class. The resulting output is a list of five probabilities representing the model's confidence that the input image is that class. This model structure is depicted in **Figure 3**.

The model trains on the training data over 50 epochs and validates the training on a validation set of 10% of the training data after each epoch. During the training phase, the model has callbacks to an early stopping function and a model checkpoint function. The early stopping function monitors the loss on the validation set and ends the training before the 50th epoch if the loss is not improving. The comparison between training loss and validation loss can be seen in **Figure 4**. The model checkpoint monitors the accuracy on the validation set and saves the best model, because sometimes the model at the end of the 50 epochs is not actually the most accurate. The full code to preprocess the data, create and train the model, and evaluate the model's results can be found in the following GitHub repository: [https://github.com/funkjo/senior_research]

3. RESULTS

After training the model on 4,425 images of cars, crosswalks, fire hydrants, palm trees, and traffic lights, the model was 83.78% accurate at classifying unseen images. The accuracy on the test set is an improvement from the accuracy on the validation set during training. The validation set achieved a maximum accuracy of 81.1% accuracy. The training accuracy compared to the validation accuracy during training can be seen in **Figure 5**.

The model was most effective at classifying images of hydrants (88.37%), crosswalks (87.93%), and palm trees (86.49%). The model did not perform well on images of cars (75%) and images of traffic lights (65.31%). This is an

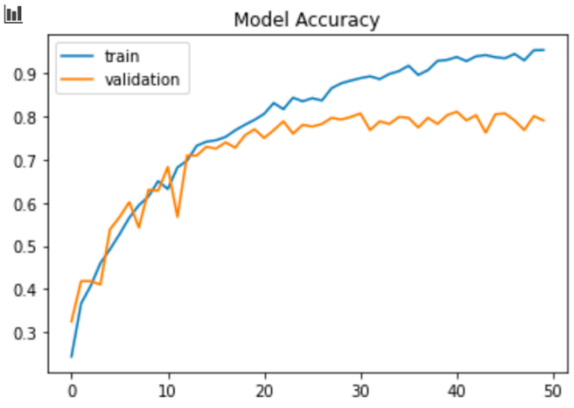

Fig 5. A visualization of the models accuracy on training data compared to validation data. The gap in the accuracy between the training set and validation set could be an indication of overfitting in the model.

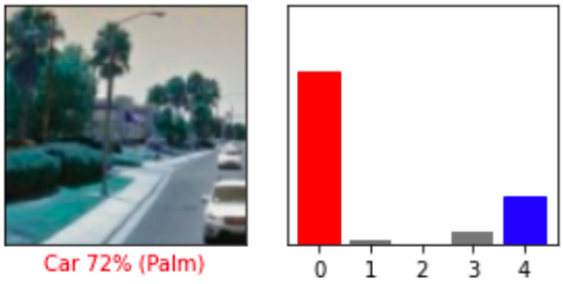

Fig 6. The model classified this image as a car and was technically correct because there are two cars in the picture. However, the correct classification was palm tree. The model was mostly confident that this image was a car and partially confident that it was a palm tree.

interesting result because the second largest image class had one of the lowest accuracies. Despite there being a large bias in the dataset towards images of cars, the training was relatively ineffective compared to the accuracies of the other images.

The model performed significantly better on images of hydrants, crosswalks, and palm trees because of the nature of those images. For the most part, in each of those image classes the subject of the image is in the center of the image, relatively large compared to its surroundings, and unobstructed and fully in view. In comparison, the model struggled on cars and traffic lights because in many cases, the car or traffic light was far in the background of the image or was obstructed or cut off.

Another interesting aspect of the results was that in some cases, even when the model was correct, it was wrong. This was because the image that it was classifying had multiple target classes. For example, in some instances the model would classify an image as a crosswalk, but the correct classification was a car. This is because both the car and the crosswalk were featured in the image. **Figure 6** shows how multiple target classes in an image could confuse the model.

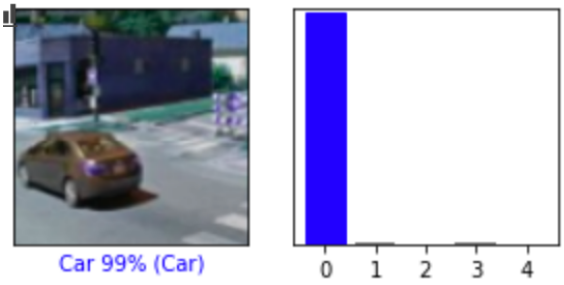

Fig 7. Although the model was not was not as effective at classifying images of cars, when the car is in full view, centered, large, and unobstructed the model is 99% confident in its classification.

Some of the other images in the dataset proved to be difficult for the model because of the high degree of noise. Some of the images in the dataset were completely incoherent and understandably difficult to classify. In some instances, the images could have been too difficult to classify even for a human. As a result, some of the misclassifications were caused by intense noise in images.

4. DISCUSSION

The results show that the hypothesis was correct. The images that lacked multiple class targets, noise, distortion, and were closer to the foreground and more centralized were easier for the model to predict. For the most part, images that followed this pattern were images of fire hydrants, crosswalks, and palm trees. On the other hand, the model had difficulty with images of cars and traffic lights for these reasons.

Another reason why the model had difficulty that was not predicted in the hypothesis were images with obstructions or images where the target was cut off.

When a machine learning model was more than 82% accurate at classifying images from the Asirra dataset of images, CAPTCHAs that utilized that dataset were considered to be compromised.[5] The model created in this research was 83.78% accurate on unseen data, surpassing the benchmark set in previous work. As a result, any CAPTCHA program utilizing the dataset in this work could potentially be compromised.

In order to improve the safety of websites from bots and automated scripts and to increase the robustness of modern image selection CAPTCHA programs, the research conducted in this work would suggest that images where the target class is in the background, obstructed or cut off, or surrounded by other target classes are harder to classify. As a result, more of these types

of images should be included in the programs dataset.

Future work would include improving the dataset. For example, training on a larger and more balanced dataset would help to improve accuracy. In addition, removing images with extreme amounts of noise in order to clean the dataset would be beneficial for this research.

Other future work would include training the model on specific color channels of images. The model might put varying degrees of importance on different color channels (red, blue, green). Experimenting with different color channels could increase the accuracy of the model. In addition to training on different color channels, the model could also be trained on different types of images. For example, different target classes could be introduced. Or the model could be trained on different types of images of the same classes such as images where the target class is extremely close in the foreground and broken up into multiple images.

Neural networks in other works, such as in the article, “Bangla Handwritten Digit Recognition Using Deep Convolutional Neural Network,” [2] have different model structures. Other future work would include experimenting with different model structures such as a Conv-Conv-Pool structure or a VGG like model structure.[9]

REFERENCES

[1] J. Elson, J. (JD) Douceur, J. Howell, and J. Saul, "Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization," Oct. 2007, Accessed: Oct. 02, 2020. [Online].

[2] R. Basri, M. R. Haque, M. Akter, and M. S. Uddin, "Bangla Handwritten Digit Recognition Using Deep Convolutional Neural Network," in *Proceedings of the International Conference on Computing Advancements*, New York, NY, USA, Jan. 2020, pp. 1–7, doi: 10.1145/3377049.3377077.

[3] Y. Zhang, J. Gao, and H. Zhou, "Breeds Classification with Deep Convolutional Neural Network," in *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*, New York, NY, USA, Feb. 2020, pp. 145–151, doi: 10.1145/3383972.3383975.

[4] X. Yang, Z. Zeng, S. G. Teo, L. Wang, V. Chandrasekhar, and S. Hoi, "Deep Learning for Practical Image Recognition: Case Study on Kaggle Competitions," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, New York, NY, USA, Jul. 2018, pp. 923–931, doi: 10.1145/3219819.3219907.

[5] P. Golle, "Machine learning attacks against the Asirra CAPTCHA," in *Proceedings of the 15th ACM conference on Computer and communications security*, New York, NY, USA, Oct. 2008, pp. 535–542, doi: 10.1145/1455770.1455838.

[6] R. Zhong and T. Tezuka, "Parametric Learning of Deep Convolutional Neural Network," in *Proceedings of the 19th International Database Engineering & Applications Symposium*, New York, NY, USA, Jul. 2015, pp. 226–227, doi: 10.1145/2790755.2790791.

[7] G. Erus, M. Habes, and C. Davatzikos, "Chapter 16 - Machine learning based imaging biomarkers in large scale population studies: A neuroimaging perspective," in *Handbook of Medical Image Computing and Computer Assisted Intervention*, S. K. Zhou, D. Rueckert, and G. Fichtinger, Eds. Academic Press, 2020, pp. 379–399.

[8] A. Budhiraja, "Learning Less to Learn Better — Dropout in (Deep) Machine learning," *Medium*, Mar. 06, 2018. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 (accessed Dec. 09, 2020).

[9] S. Ramesh, "A guide to an efficient way to build neural network architectures- Part II: Hyper-parameter…," *Medium*, Oct. 07, 2020. https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7 (accessed Dec. 09, 2020).