Breaking CAPTCHA: A Convolutional Neural Network Approach

Georgios Pnigouras

Abstract

This study explores CAPTCHA breaking using machine learning methods, focusing on Convolutional Neural Networks (CNNs). The topic is important as it demonstrates the capabilities of CNNs in solving complex image recognition tasks. The primary research question investigates whether, and to what extent, machine learning methods can accurately detect characters in distorted CAPTCHA images. The dataset used consists of 1,039 CAPTCHA images, each containing five alphanumeric characters. Initially, an OCR-based model was tested, but it performed poorly. Two CNN-based models were then employed. The first treated the image as a whole during training, while the second, more advanced model, divided training into five sub-processes, each responsible for predicting one character position. The results show that the simple OCR model achieved a low accuracy of 5.29% for full-text predictions, while the CNN trained on the entire image improved to slightly over 43%. The best performance came from the advanced CNN model, achieving over 75% accuracy by focusing on individual character positions. These findings highlight the potential of tailored CNN architectures in solving CAPTCHA recognition challenges effectively.

1. Introduction

The detection of CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) presents a valuable opportunity to explore the capabilities and limitations of Convolutional Neural Networks (CNNs) in addressing real-world challenges. CAPTCHAs are security mechanisms designed to distinguish between human users and automated bots by presenting distorted text that is intentionally difficult for machines to interpret.

In this project, the input variable (X) consists of a dataset of 1,039 images, each containing five distorted alphanumeric characters (see Appendix 1). The primary objective is to train advanced machine learning models, specifically CNNs, to accurately predict the characters present in these images (y, the target variable). By leveraging deep learning, the model aims to enhance the recognition of complex patterns and deformations inherent in CAPTCHA images.

The features (X) in this dataset are pixel-level representations of grayscale images, while the target variable (y) consists of the individual characters embedded within each CAPTCHA image. These features and targets are particularly well-suited for CNNs due to their ability to learn hierarchical spatial representations in image data. The accurate detection of CAPTCHA characters not only poses a significant technical challenge but also serves as an excellent benchmark for evaluating the robustness, efficiency, and generalization capabilities of CNN architectures.

Although this specific form of CAPTCHA is now considered outdated and is no longer widely used by institutions or companies for spam and bot protection, this analysis remains relevant. The ethical implications of this research are mitigated by the fact that the findings cannot be directly applied to bypass modern security measures. Instead, this project provides a safe and practical environment to demonstrate the power of CNNs while maintaining ethical integrity. Furthermore, this work highlights broader machine learning applications in optical character recognition (OCR) and automated pattern detection, emphasizing its relevance in fields like document processing, accessibility technologies, and automated data entry. (1)

2. Data and Methods

2.1. Data

The dataset used for this analysis comes from the University of Monterrey and consists of 1,039 CAPTCHA images, each 50x200 pixels in size, containing 5 alphanumeric characters. The filenames of the images provide the corresponding labels. An example of an image is shown in Appendix 1. We can see that the true label, `"3rg2w6"`, is easily readable by humans, but the introduction of obstructions such as the line going through the image, the blurring and distortion of characters, alongside what seems to be a lowercase `"r"` before the `"3"` is bound to confuse an algorithm attempting to read this text.

However, these are not exactly the images fed into the model, i.e., the images are slightly preprocessed by converting them

to grayscale and standardizing their dimensions to 50x200 pixels. This is to ensure that the model knows to handle one color channel instead of the three that RGB pictures use, and also that all the images are the same size. This procedure does not affect at all the readability by human standards. The dataset is then split into training, validation, and test sets to ensure proper evaluation of the models. Each image serves as the input (X), while the label (y) corresponds to the five-character sequence in the CAPTCHA. (2)

2.2. Methods

2.2.1. Optical Character Recognition

The first and simplest method we will use in this project is Optical Character Recognition (OCR). OCR is a process that converts text from a digital image into text. This method involves three main steps: pre-processing, segmentation, and recognition. During pre-processing, the image is enhanced to improve recognition accuracy through techniques such as noise reduction, binarization, skew correction, and contrast enhancement. In the segmentation step, the identified text regions are divided into smaller units like lines, words, or individual characters for more precise recognition. The final step involves recognizing the text, evaluating its accuracy, and enhancing validation over time. The OCR method was implemented in R using the `imager` and `tesseract` packages, which offer various functions for image visualization and processing. The `ocr()` function serves as the final step to extract and recognize text from images.

2.2.2. Convolutional Neural Network

Recognizing the expected poor performance of traditional Optical Character Recognition (OCR) techniques on distorted CAPTCHA images, we emphasized the need for more advanced deep learning-based methods. To address this challenge, two Convolutional Neural Network (CNN) models were developed. While both models share a similar convolutional architecture, they differ greatly in their approach to training and prediction. The first model is designed to optimize predictions for all five CAPTCHA characters simultaneously, whereas the second model decomposes the task into five independent sub-processes, each responsible for predicting a single character in a specific position.

The first stage of both approaches involved comprehensive image preprocessing. All images were converted to grayscale to reduce computational complexity, as color information does not contribute meaningfully to character recognition. To ensure consistency across the dataset, each image was resized to a fixed dimension of 50 pixels in height and 200 pixels in width. Furthermore, pixel values were normalized to a range between 0 (black) and 1 (white) to enhance model stability and convergence. Additionally, the expected vocabulary was limited to 36 unique characters, consisting of 26 lowercase Latin letters (a–z) and 10 digits (0–9). To facilitate efficient representation during training, each character was mapped to a unique numerical identifier, ensuring uniform encoding.

For both approaches, the dataset was partitioned into training, validation, and test subsets, following a ratio of 80%, 10%, and 10%, respectively. In the second method, which utilizes five independent sub-models (one per character position), one-hot encoding was applied to effectively represent character labels.

The first model follows a unified prediction approach, attempting to classify all five CAPTCHA characters in a single forward pass. It consists of three convolutional layers, each utilizing a 3×3 kernel to extract hierarchical spatial features from the images. Following each convolutional layer, a pooling layer is applied to reduce dimensionality while retaining essential features. To mitigate overfitting, L2 regularization is employed, discouraging excessively large weights, alongside a dropout layer for additional regularization. The extracted feature maps are then flattened into a one-dimensional vector before being processed by two fully connected dense layers. The first dense layer captures high-level patterns, while the final dense layer generates predictions for all five CAPTCHA characters simultaneously. The architecture of this model is detailed in Appendix 2.

The second model shares a similar convolutional backbone but diverges in its dense layer design. Instead of predicting all characters in one step, it is structured into five separate output branches, with each branch dedicated to classifying one specific character position. Each branch contains two sets of dense layers—one for learning advanced position-specific features and another for generating a probability distribution over all possible character labels. This independent classification for each character position allows for a more specialized learning process. The architecture of this model is detailed in Appendix 3.

2.3. Evaluation

To assess the performance of both models, evaluation was conducted using the training, validation, and test sets. Throughout training, we closely monitored both accuracy and loss for the training and validation sets at the end of each epoch. By tracking validation accuracy and loss, we assessed how well the models generalize to unseen data. The evaluation process relied on two key components: the loss function and the optimizer.

For the first model, the loss function used was Sparse Categorical Cross-Entropy, as the labels were provided as integer indices rather than one-hot encoded vectors. This loss function effectively measures the discrepancy between true labels and predicted probability distributions when dealing

with categorical outputs. In contrast, the second model, which produces five separate predictions and uses one-hot encoding, required a different approach. Here, Categorical Cross-Entropy was used instead, as it is specifically designed for multi-class classification with one-hot encoded labels, ensuring that each output branch learns the correct probability distribution for its respective character position.

The optimizer, Adam, was applied in both models to dynamically adjust the model weights based on loss values, ensuring stable and efficient convergence. Model performance was primarily measured using accuracy, defined as the fraction of correctly classified CAPTCHA characters over the total number of characters in the dataset. Additionally, a separate test set—an independent dataset that was not used during training or validation—was employed for final model evaluation.

For the second model, the evaluation followed the same fundamental process but with minor differences. Since this model generates five independent predictions (one for each character position), accuracy was initially measured at the individual character level. Each output branch's performance was assessed separately by counting the fraction of correctly predicted characters. However, this metric alone was insufficient, as the ultimate goal was to predict the entire CAPTCHA correctly, not just individual characters.

To accurately reflect this goal, an additional metric was introduced: full CAPTCHA accuracy. A CAPTCHA prediction was only considered correct if all five characters were predicted correctly in the correct order, matching the true labels exactly. This ensured that overall model performance was aligned with real-world CAPTCHA recognition requirements.

3. Results

3.1. Optical Character Recognition

The "OCR Accuracy" results highlight the need for advanced methods to handle distorted CAPTCHA images. The Tesseract model accurately predicts only 5.29%.
From Appendix 1 we can see how the model attempts to predict CAPTCHAs, and we can see that it does indeed have many problems in getting even the number of digits correct, for example in label `"244e2"`, the predicted label is `"24"`. This makes sense, especially given that this model intends to predict actual text, not distorted images of random characters.
Furthermore, we can look at Appendix 4, which shows that the average prediction accuracy for each separate character does not exceed 50%, with the last character, the fifth, having the highest accuracy. For the confusion matrix of character prediction, seen in Appendix 5, we see a much higher misclassification performance compared to other confusion matrices for other models, which will be discussed in later sections, underlying that this technology is not at all effective at reading CAPTCHAs.
In conclusion, the results clearly demonstrate that the simple OCR method is insufficient for recognizing text that has been distorted or modified. Thus, in the later sections, we will shift our focus to deep learning methods to overcome the limitations of the simple approach, which tends to perform better.

3.2. Simple CNN - Full CAPTCHA Prediction

This first, simple attempt at a Convolutional Neural Network achieved good results in training and validation, with a 93.12% training accuracy and 80.38% validation accuracy. The gap between the two accuracies, visualized in Appendix 6 indicates a slight tendency of the model to overfit the training data. However, the testing accuracy is most striking, at a meager 43.27% (see Appendix 2). The model appears to be predicting individual letters rather well, but due to the conjunction of them in every label, it is often the case that the predictions are slightly wrong. The actual results of the prediction of each individual letter can be seen in Appendix 7 and Appendices 8,9,10,11, and 12,. Appendix 7, shows a hidden, but intuitive in cause, pattern for the accuracy of the individual letters, where the outer letters in the CAPTCHA (1st and 5th) are much better identified than central letters (specifically 3rd and 4th). This is most likely due to the nature of the CAPTCHAS, where letters are often in part superimposed unto one another, and the outer letters are clearer, with more defined edges from the blank space. Appendix 3, shows some examples of these misclassifications.

3.3. CNN - Individual Letter Prediction

The more refined Convolutional Neural Network performs rather well in predicting CAPTCHAS. The largest difference between this and the previous model, as previously described, is the ability to predict individual letters in the CAPTCHA. Breaking the problem down is intuitive, as it allows differentiation between how a letter or number appears at the edges versus in the center of the CAPTCHA, which was a major issue in the simple model. With this more developed CNN, we achieve, as can be seen in Appendix 4, an averaged (in between individual letters) testing accuracy of 98.2%, and an averaged validation accuracy of 90.31%. On the test set, we achieve a 76.9% prediction accuracy, which is extremely satisfactory, and a large increase from the simple model. We can see an example of the results for the test set in Appendix 5. When looking at individual character recognition visualized in Appendix 13, we see once again a similar pattern, where the outer letters are more accurately classified, whereas the model lightly struggles in the middle characters.

4. Conclusion

This paper investigated CAPTCHA recognition, firstly using a traditional OCR method, but primarily CNN approaches. The results gathered underline significant limitations in primitive OCR for handling distorted CAPTCHA images, evidenced by the low accuracy achieved, at 5.29%.

The CNN-based methods demonstrated the potential for improved accuracy and recognition in classifying the CAPTCHAS, with the refined model achieving very good prediction abilities. The division of the CAPTCHAs into individual characters greatly boosted the model's overall accuracy, addressing challenges faced by the simpler model in discerning overlapping and distorted characters. The advanced CNN model, focusing on predicting each character position separately, achieved an impressive 76.9% accuracy on the test set, substantially improving over the simple CNN model's 43.27% accuracy. This improvement highlights the importance of training Neural Networks to break down complex problems into smaller, more manageable tasks, a strategy that proved highly effective in this context.

This is however not to say that the model is perfect, and limitations, mostly dependent on our dataset are very clear. The dataset is rather small and does not account for varying types of distortions and noise, but most importantly, it does not include all sets of both capitalized and non-alphanumeric characters. The model is also not length-adaptable, meaning that it is not able to identify CAPTCHAs with other lengths than 5 characters, as the model is only told to break the text into 5 characters, and the output is limited to a 5-character vector. This would clearly pose a further challenge, but due to time constraints, the analysis was limited. A further model, which was tested, but would achieve extremely low scores in accuracy perhaps precisely due to the low amount of training samples, was a multi-head transformer, which would also be rather suited for the task of separate character recognition, with its attention heads.

We want to express that, from an ethical perspective, the ability to decode CAPTCHAs raises concerns about the misuse of this technology. The intent of this research is purely academic and aimed at understanding the weaknesses of these systems. Fortunately, this form of CAPTCHAs has been discontinued en masse in favor of much more complex methods using pictures and pointer-movement recognition, with little worry that our models would perform well on these advanced tests.

In conclusion, this study has demonstrated the effectiveness of CNNs in breaking CAPTCHAs. The results highlight that deep learning techniques are able to solve complex image recognition challenges and provide valuable insights into the design and evaluation of machine learning models for similar tasks. While there are indeed strong limitations to the approach taken here, the findings may inspire future research in this area, to identify potential applications in text-reading and perhaps handwriting identification.

5. Appendix

5.1. Appendix 1

Figure 1. Example Captcha

5.2. Appendix 2

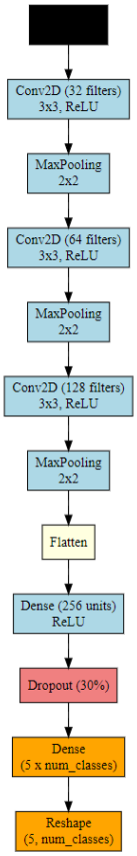

Figure 2. Simple CNN Architecture

5.3. Appendix 3

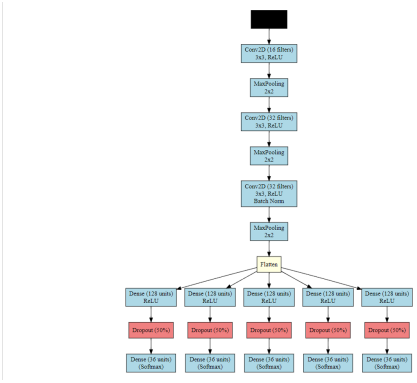

Figure 3. Advanced CNN Architecture

5.4. Appendix 4

Predicted	Actual	Correct
27read	226md	FALSE
722d5n	22d5n	FALSE
2356q	2356g	FALSE
2andig	23mdg	FALSE
723n83	23n88	FALSE
242yyn	243mm	FALSE
24	244e2	FALSE
2ays	245y5	FALSE
24fan	24f6w	FALSE
24nen	24pew	FALSE

Table 1. Example OCR Model Predictions

5.5. Appendix 5

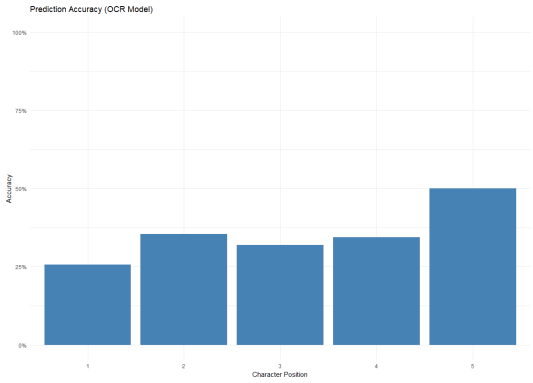

Figure 4. Bar Plot for OCR Model of Per-Character Accuracy

5.6. Appendix 6

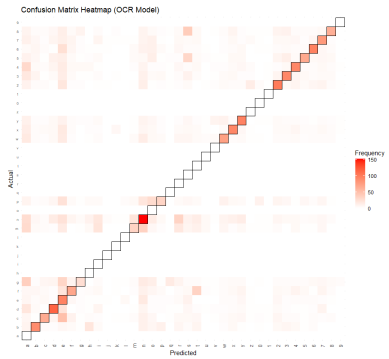

Figure 5. Confusion Matrix for OCR Model

5.7. Appendix 7

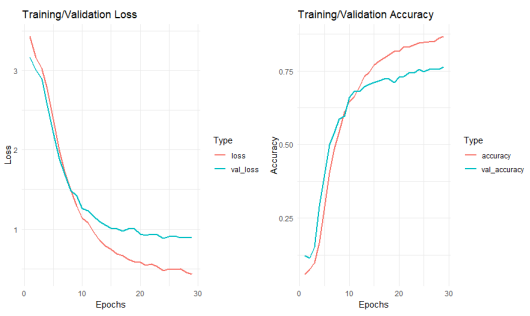

Figure 6. Simple Model Loss/Accuracy Plots during Training

5.8. Appendix 8

Accuracy Type	Accuracy
Training All Char	93.12%
Validation All Char	80.38%
Testing All Char	43.27%
Testing Char 1	91.35%
Testing Char 2	86.54%
Testing Char 3	80.77%
Testing Char 4	68.27%
Testing Char 5	82.69%

Table 2. Accuracy Scores for Training, Validation, and Testing for Advanced CNN Model

5.9. Appendix 9

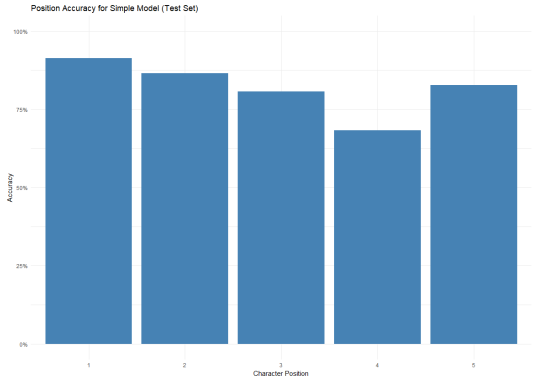

Figure 7. Bar Plot for Simple CNN of Per-Character Accuracy on Test Set

5.10. Appendix 10

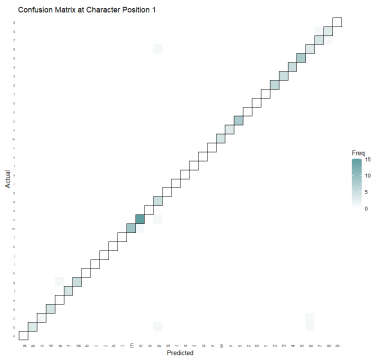

Figure 8. Confusion Matrix for Simple CNN for Character 1

5.11. Appendix 11

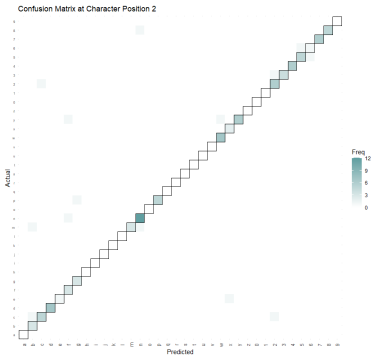

Figure 9. Confusion Matrix for Simple CNN for Character 2

5.12. Appendix 12

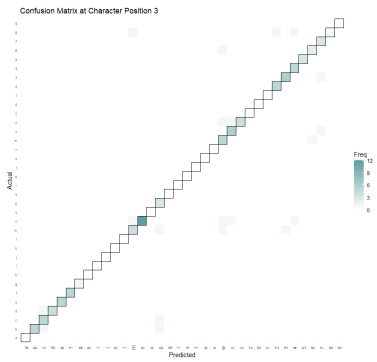

Figure 10. Confusion Matrix for Simple CNN for Character 3

5.13. Appendix 13

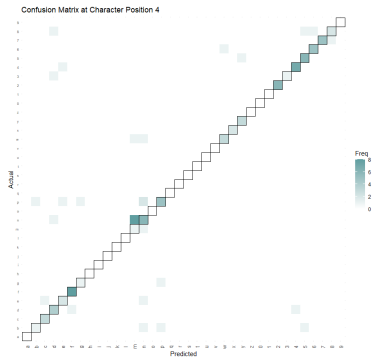

Figure 11. Confusion Matrix for Simple CNN for Character 4

5.14. Appendix 14

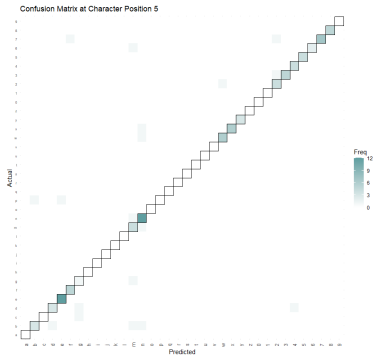

Figure 12. Confusion Matrix for Simple CNN for Character 5

5.15. Appendix 15

Predicted	Actual	Correct
23g88	23n88	FALSE
25w53	25w53	TRUE
268g2	268g2	TRUE
2y7bm	2x7bm	FALSE
2ycn8	2ycn8	TRUE
2ypgp	2yggg	FALSE

Table 3. Example Simple CNN Test Set Predictions

5.16. Appendix 16

Accuracy Type	Accuracy
Training Char 1	98.56%
Training Char 2	98.80%
Training Char 3	97.35%
Training Char 4	98.19%
Training Char 5	98.32%
Validation Char 1	99.04%
Validation Char 2	93.27%
Validation Char 3	85.58%
Validation Char 4	83.65%
Validation Char 5	89.42%
Testing Char 1	100.00%
Testing Char 2	97.12%
Testing Char 3	93.27%
Testing Char 4	86.54%
Testing Char 5	94.23%

Table 4. Accuracy Scores for Training, Validation, and Testing for Advanced CNN Model

5.17. Appendix 17

Predicted	Actual	Correct
23n88	23n88	TRUE
25c53	25w53	FALSE
268g2	268g2	TRUE
2x7bm	2x7bm	TRUE
2ycn8	2ycn8	TRUE
2yggg	2yggg	TRUE

Table 5. Example Advanced CNN Test Set Predictions

5.18. Appendix 18

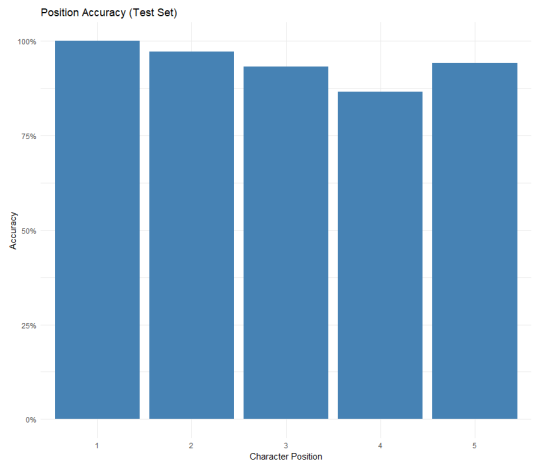

Figure 13. Bar Plot for Advanced CNN of Per-Character Accuracy on Test Set

References

[1] A. Searles, Y. Nakatsuka, E. Ozturk, A. Paverd, G. Tsudik, and A. Enkoji, *An Empirical Study Evaluation of Modern CAPTCHAs*, 2023.

[2] R. Wilhelmy, "Captcha dataset," Jul 2013. [Online]. Available: https://www.researchgate.net/publication/248380891_captcha_dataset