Preprocessing of CAPTCHA images before recognition

Lean Jeng Wen Joshua, Aurick Daniel F S
(連正文 111000178), (許木羽 111000177)
joshualeanjw@gmail.com, aurickd@gmail.com

Abstract. This report investigates various preprocessing techniques for CAPTCHA image recognition, enhancing neural network performance. We explore methods like noise reduction, color segmentation, and morphological transformations on CAPTCHAs of different complexities. Using neural network architectures, we benchmark these techniques, observing notable improvements in recognition accuracy, especially for complex CAPTCHAs. Our findings highlight the effectiveness of specific preprocessing methods in CAPTCHA recognition, offering insights with broader implications in image processing and machine learning. This work contributes to developing more efficient automated systems for CAPTCHA solving, with potential security and accessibility impacts.

1. Introduction

The role of CAPTCHA [1] (Completely Automated Public Turing test to tell Computers and Humans Apart) in digital security is crucial, yet its recognition poses unique challenges due to its design complexity aimed at deterring automated solving. This report focuses on preprocessing CAPTCHA images to enhance their recognition by neural networks.

Advancements in machine learning, especially neural networks, have significantly impacted image recognition. However, CAPTCHA images, known for their noise, distortion, and complex backgrounds, require specialized preprocessing methods for effective recognition. This study examines various preprocessing techniques, such as noise reduction, color segmentation, and morphological operations, assessing their impact on the performance of neural network architectures.

Our research aims to benchmark these preprocessing methods, enhancing CAPTCHA recognition accuracy and contributing to the broader field of image processing and machine learning. This report not only addresses the gap between CAPTCHA complexity and neural network capabilities but also aims to advance the development of more robust CAPTCHA recognition systems. The following sections detail our methodologies, experiments, and the insights gained from our comprehensive analysis.

2. Methods

The inherent design of CAPTCHAs presents a formidable challenge for standard Optical Character Recognition (OCR) systems due to their intentional complexity [2]. To overcome this, our methodology integrates image preprocessing techniques [3] with advanced neural network models. The general process shown in Figure 1.

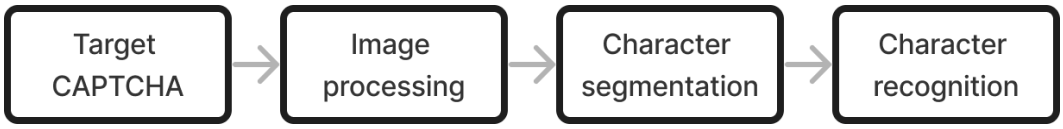

Figure 1. General process of CAPTCHA recognition

In the initial stage of our CAPTCHA recognition framework, we focus on employing a range of image processing techniques to simplify the inherently complex nature of CAPTCHA images. This step is critical as it transforms the CAPTCHA from a state that is challenging for standard recognition algorithms to one that is more conducive to analysis and interpretation. CAPTCHAs are typically designed with various obfuscating features such as background noise, distorted text, overlapping characters, and color gradients, all of which are intended to prevent automated systems from accurately interpreting the content. The general process for image processing is shown in Figure 2. A sample of a CAPTCHA image is shown in Figure 3, 5, and the ideal output image of the image processing stage is shown in Figure 4, 6.

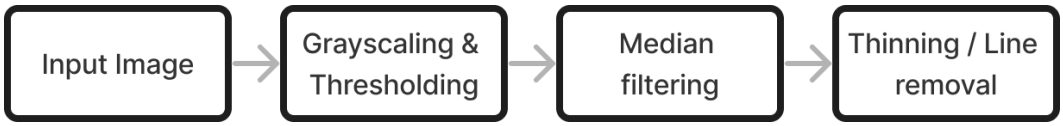

Figure 2. General methods of image processing part of CAPTCHA recognition

Figure 3, 5. Original image of a low difficulty CAPTCHA and its state after processing.

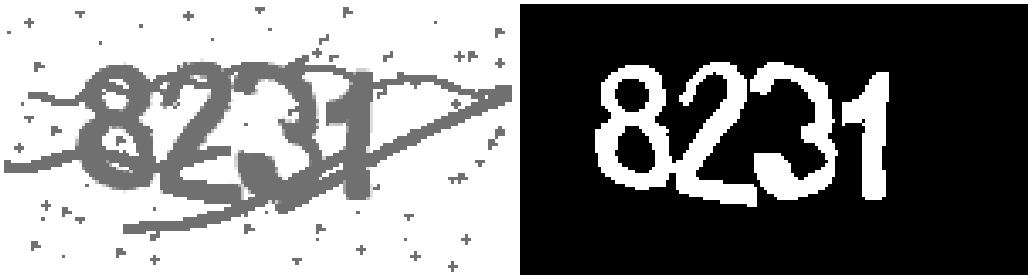

Figure 4, 6. Original image of a high difficulty CAPTCHA and its state after processing.

The initial step in our preprocessing pipeline involves converting the CAPTCHA image from a color image to grayscale. This simplification is essential as it reduces the complexity by eliminating the color information, focusing solely on the intensity of the pixels [4]. Following grayscaling, we apply a thresholding technique. Thresholding is a form of image segmentation that involves converting the grayscale image into a binary image [4]. It does this by setting all pixels above a certain intensity to white and all others to black, which helps in highlighting the text against the background. This binary distinction makes the subsequent processing steps more effective as it provides a clearer distinction between the text (foreground) and the background. Figure 7, 8 shows the originals and Figure 9, 10 shows the images after grayscale and thresholding.

Figure 7, 9. Low difficulty original image before and after grayscale and thresholding

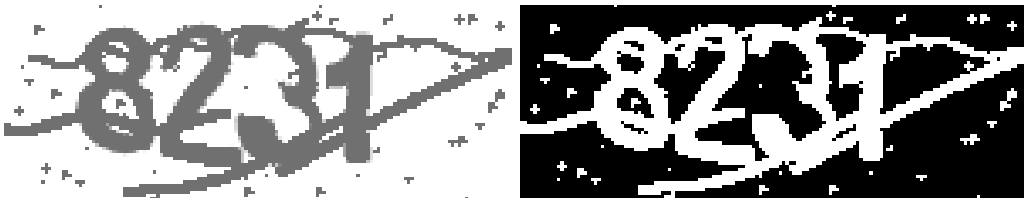

Figure 8, 10. High difficulty original image before and after grayscale and thresholding

As the above result, CAPTCHAs where added noises have low intensity, it is easy to clean the image of noise. For difficult CAPTCHAs where color and intensity do not pose a threat, the next method tested is median filtering, a noise reduction technique that is particularly effective in removing small-scale noise from an image while preserving the edges of the characters. In median filtering, each pixel is replaced with the median value of the neighboring pixels within a specified window. This process smooths out the image, reducing the impact of any artifacts or irregularities that might confuse the recognition algorithms. Figure 11 shows the image that ran through a 3x3 filter size median filter, Figure 12 shows the same method done twice.

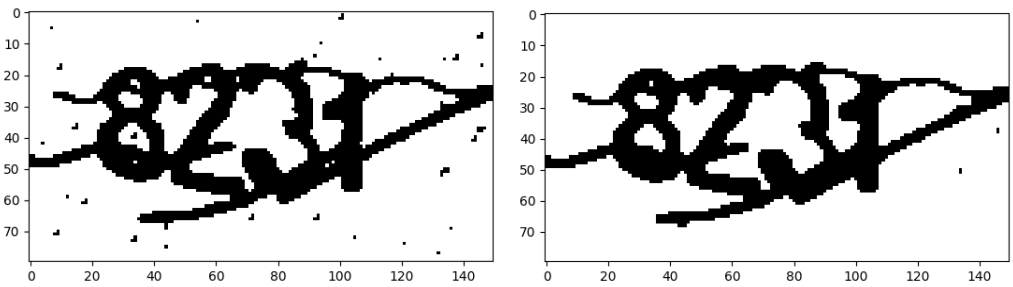

Figure 11, 12. Image after median filtering once and twice

We perform median filtering multiple times instead of using a larger filter size to retain the most information. Lines of noises retain from the median filtering step, an observation is that the line noises are thinner relative to the features to extract. Thinning is a morphological operation that is used to reduce the thickness of the characters to a single pixel width [5, 6, 7, 8, 9]. This process is beneficial for standardizing the appearance of the text and making it more uniform, which is particularly useful for neural network-based OCR systems. Line removal is an additional step aimed at eliminating any straight lines that are not part of the characters but may be present in the CAPTCHA as a form of noise. This is crucial because such lines can be misinterpreted as part of the characters, leading to errors in recognition.

It is better to retain information rather than remove as much noise as possible, as noises can be ignored in machine learning steps as long as majority trained data is clean of noise. We found two methods to have relatively better results.

A modified median filtering [10] function where both axes of the blurred version of the image are checked for a certain height, preventing the thicker features to be removed, and only removing lines below a certain threshold. Figure 13 and 14 shows the before and after the application of this modified median filtering function. Figure 15 is the pseudocode of the modified median filtering function. This algorithm works since the number is always grouped together in a certain part and the sum value (horizontally or vertically) should be higher than the other part. Some sum value (e.g. vertical) in a certain part of the number might be lower than threshold, that's why we use blurred to check the sum, so the value of vertical in the middle of the number is distributed evenly.

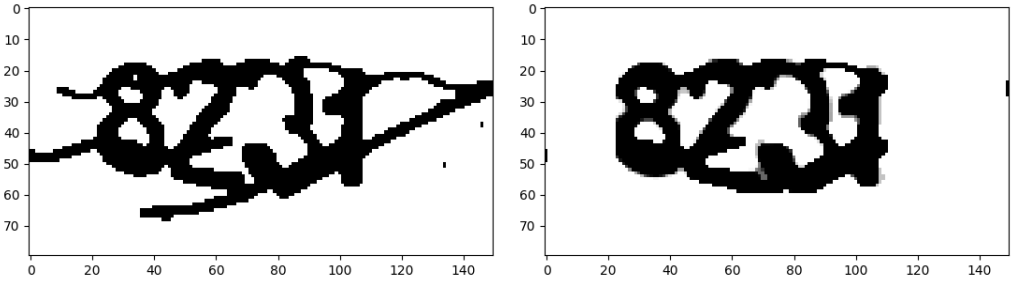

Figure 13, 14. Before and after applying the modified median filtering function

```
Function modified_median_filter(input_image, blur_image)
    Initialize output as a copy of input_image
    Calculate vertical_sum by summing blur_image along the vertical axis
    Calculate horizontal_sum by summing blur_image along the horizontal axis
    Set margin (marg) to 1

    For each pixel in input_image (excluding the margin):
        If vertical_sum at current column > n * 255 // n as vertical threshold
       OR horizontal_sum at current row > m * 255: // m as horizontal threshold
            Set the corresponding pixel in output to 255

    Return output
```

Figure 15. Pseudocode of the modified median filtering function

Another method of removing lines, using a custom morphological function [11]. It examines each pixel in an image and counts the number of black pixels in its immediate neighborhood, defined by a specified size. If the count of black pixels is below a given threshold, the pixel is turned white. This operation effectively modifies pixels based on their local context (neighboring pixels), and is used to alter the image structure, such as removing small black regions or thinning lines. Figure 16 and 17 shows the before and after the application of the custom morphological function. Figure 18 uses median filtering to remove the rest of the thin lines. Figure 19 is the pseudocode of the custom morphological function.

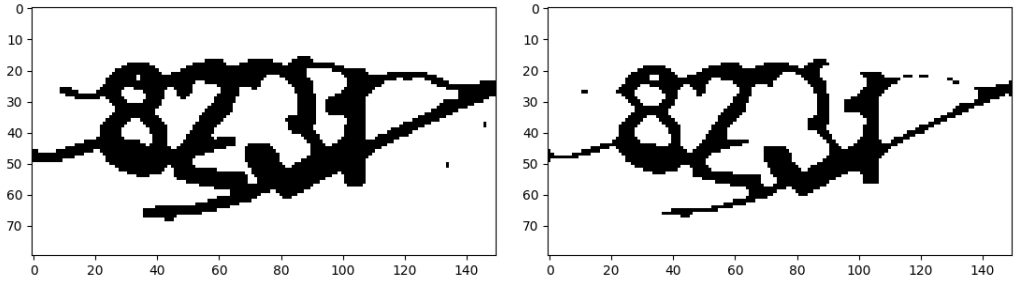

Figure 16, 17. Before and after applying the custom morphological function

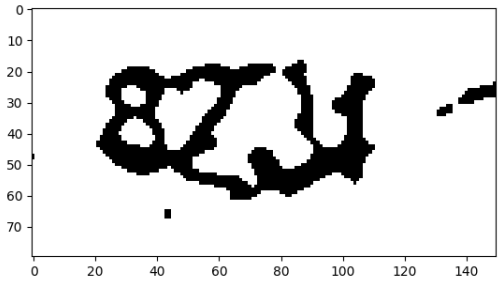

Figure 18. After applying median filter to remove thin lines

```
Function custom_morphological_operation(image, neighborhood_size, threshold)
    Initialize output as a copy of image
    Calculate offset as half of neighborhood_size

    For each pixel in image (excluding a border of size offset):
        Initialize black_pixel_count to 0

        For each neighbor pixel in the neighborhood of the current pixel:
            If the neighbor pixel is black (value 0):
                Increment black_pixel_count by 1

        If black_pixel_count is less than threshold:
            Set the current pixel in output to white (value 255)

    Return output
```

Figure 19. Pseudocode of the modified median filtering function

The second step in our CAPTCHA recognition process is the crucial task of character separation, which involves segmenting each individual digit or character from the CAPTCHA image. This step is essential because, in the machine learning phase of our methodology, the neural network models are trained to recognize individual digits (0 to 9) or characters, rather than interpreting an entire string of text at once. Efficient separation of these characters is pivotal for the success of the recognition process. The process for separation is shown in Figure 20.

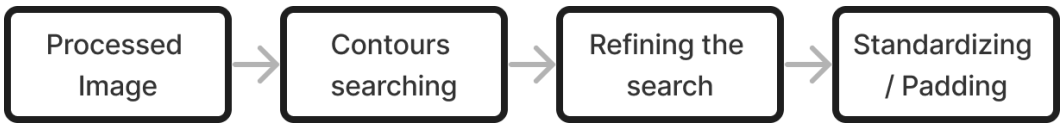

Figure 20. Process of character separation part of CAPTCHA recognition

The first task is to identify the contours of each character in the CAPTCHA image. Contour searching is an effective method for this, as it involves detecting the edges of characters based on their shapes. This is typically done using algorithms that can identify continuous points along the boundary of characters, differentiating them from the background [12]. The result of this process is a series of contours, each ideally corresponding to an individual character. Figure 21 shows a sample highlighting of the contours found.

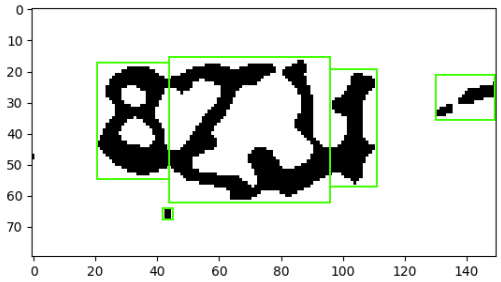

Figure 21. An example search for contours, highlighting the bounding box of results.

CAPTCHAs often feature characters that are either overlapping or very close to each other, posing a challenge for straightforward contour detection. To address this, we analyze the dimensions of each detected contour. If the width of a contour is significantly greater than its height (exceeding a predefined constant ratio), it's likely that the contour contains multiple characters that are stuck together. In such cases, the contour is further analyzed to determine the best splitting point, effectively separating the merged characters. Additionally, we implement a pixel threshold to eliminate contours that are too small to be valid characters, as these are often artifacts or noise.

Once the characters are isolated, the next crucial step is to standardize these images through padding. Padding is added to maintain the original aspect ratio without distorting the character. This process involves surrounding the resized character with additional pixels, This standardization ensures that each character maintains its structural

integrity, making it suitable for accurate recognition by the neural network. Two original, Figure 22, 23, and two sample separation is shown in Figure 24, 25.

Figure 22, 24. Difficult CAPTCHA post processed image and separation result

Figure 23, 25. Simple CAPTCHA post processed image and separation result

3. Results

In the final step of our study, while our primary focus remains on image processing techniques for CAPTCHA recognition, it is essential to briefly touch upon the neural network model employed for training and to present the results of this training. This gives context to the effectiveness of the preprocessing methods and their impact on the overall recognition system.

For the recognition of individual characters from CAPTCHA images, we utilized a Convolutional Neural Network (CNN) [13] model. CNNs are particularly well-suited for image recognition tasks due to their ability to capture spatial hierarchies in image data. Our chosen model architecture consists of few convolutional layers that extract features from the images, followed by pooling layers that reduce dimensionality and increase computational efficiency. The network also includes fully connected layers towards the end to classify each image into one of the possible characters (0-9). Figure 26 shows the topology of the neural network used.

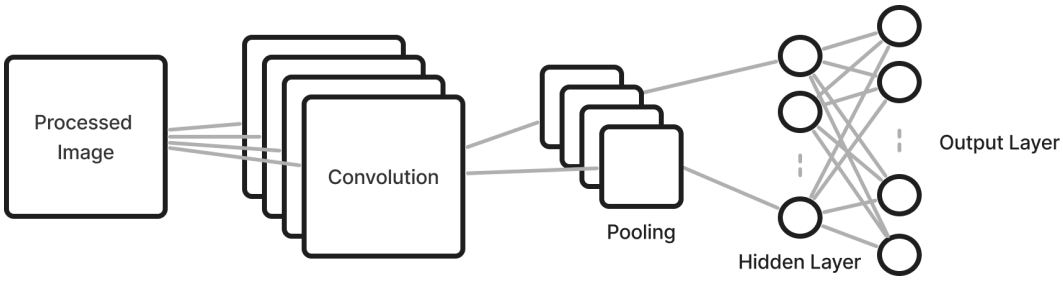

Figure 26. Convolutional neural network topology

The training process involved feeding the standardized and preprocessed images of CAPTCHA characters into the CNN model. For each type of CAPTCHA (simple and difficult), we used roughly 100 datasets for each character. And roughly 100 testing datasets to evaluate the results. Results comparing with and without preprocessing are shown below.

Comparing a simple CAPTCHA, the former being the dataset with preprocessing, and the latter being the dataset without preprocessing.

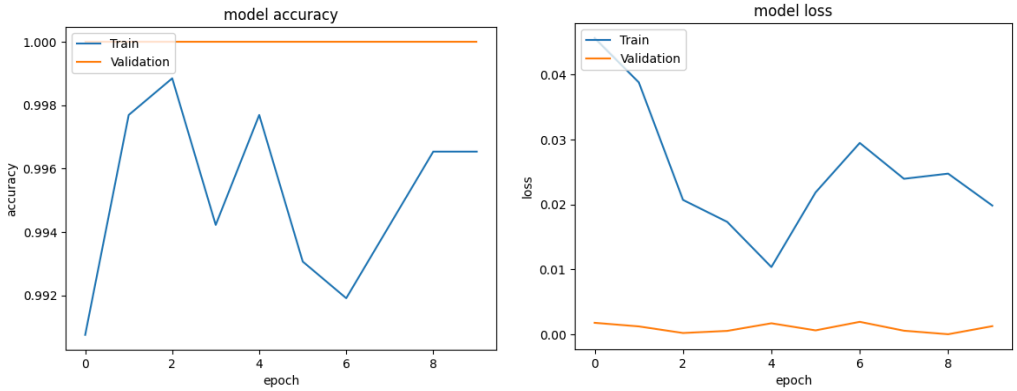

Figure 27, 28. Accuracy and loss while training (simple, with preprocess)

	precision	recall	f1-score	support
0	1.00	1.00	1.00	99
1	1.00	1.00	1.00	97
2	0.99	1.00	0.99	98
3	1.00	1.00	1.00	100
4	1.00	1.00	1.00	100
5	1.00	1.00	1.00	100
6	1.00	1.00	1.00	100
7	1.00	1.00	1.00	100
8	1.00	1.00	1.00	99
9	1.00	0.99	0.99	99
acc			1.00	992
m_avg	1.00	1.00	1.00	992
w_avg	1.00	1.00	1.00	992

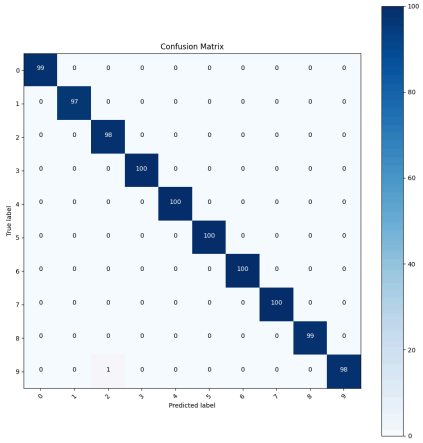

Figure 29, 30. Classification report, confusion matrix (simple, with preprocess)

Figure 31, 32. Accuracy and loss while training (simple, NO preprocess)

	precision	recall	f1-score	support
0	0.96	0.89	0.92	99
1	0.97	0.93	0.95	97
2	0.88	0.93	0.90	98
3	0.92	0.89	0.90	100
4	0.97	0.96	0.96	100
5	0.91	0.94	0.93	100
6	0.96	0.88	0.92	100
7	0.92	0.93	0.93	100
8	0.85	0.92	0.88	99
9	0.90	0.95	0.93	99
acc			0.92	992
m_avg	0.92	0.92	0.92	992
w_avg	0.92	0.92	0.92	992

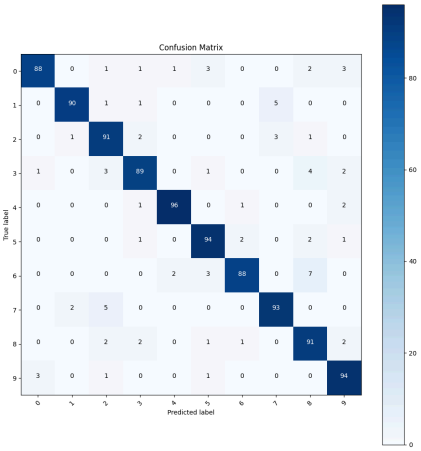

Figure 33, 34. Classification report, confusion matrix (simple, NO preprocess)

Comparing a difficult CAPTCHA, the former being the dataset with preprocessing, and the latter being the dataset without preprocessing.

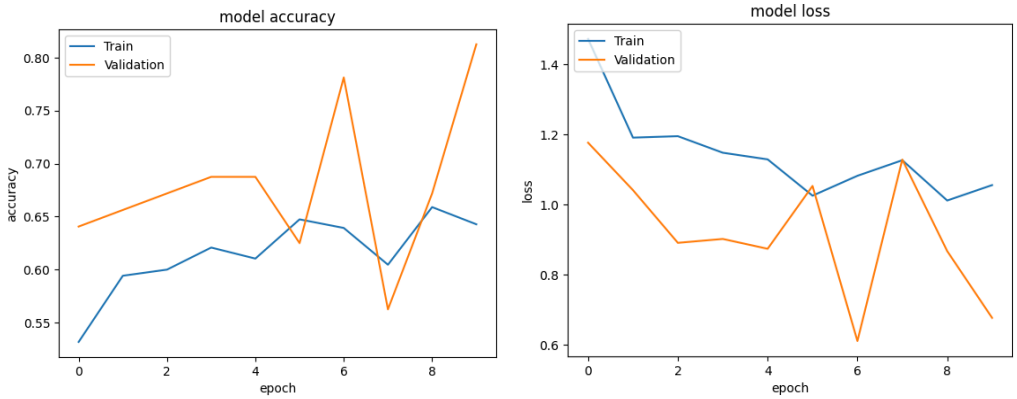

Figure 35, 36. Accuracy and loss while training (difficult, with preprocess)

	precision	recall	f1-score	support
0	0.82	0.76	0.79	99
1	0.75	0.68	0.71	97
2	0.67	0.73	0.70	98
3	0.76	0.72	0.74	100
4	0.75	0.84	0.79	100
5	0.88	0.78	0.83	100
6	0.86	0.73	0.79	100
7	0.77	0.60	0.67	100
8	0.60	0.79	0.68	99
9	0.70	0.82	0.76	99
acc			0.74	992
m_avg	0.76	0.74	0.75	992
w_avg	0.76	0.74	0.75	992

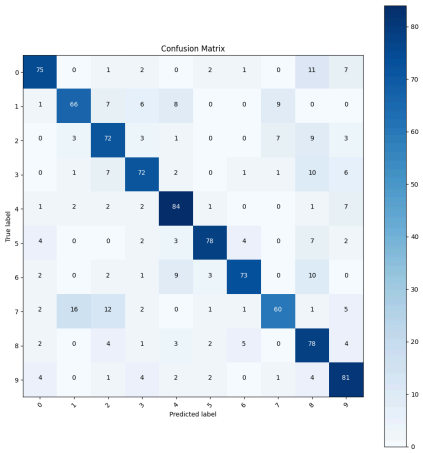

Figure 37, 38. Classification report, confusion matrix (difficult, with preprocess)

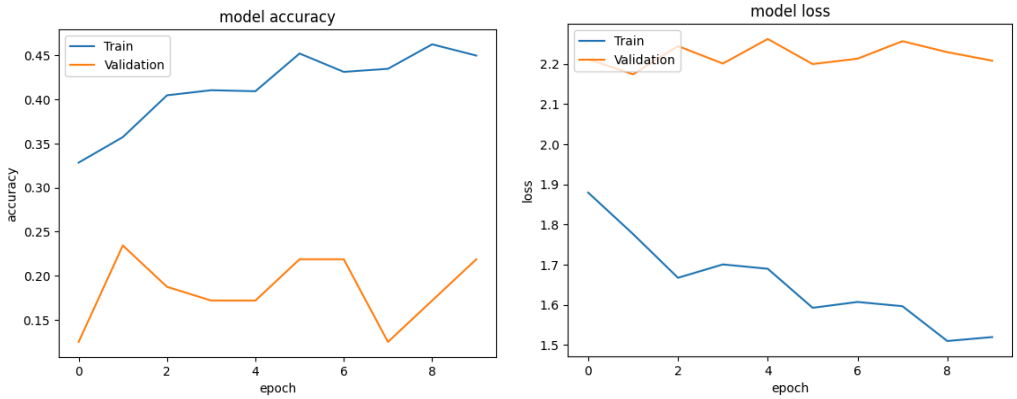

Figure 39, 40. Accuracy and loss while training (difficult, NO preprocess)

	precision	recall	f1-score	support
0	0.52	0.69	0.59	99
1	0.49	0.51	0.50	97
2	0.62	0.59	0.61	98
3	0.47	0.46	0.46	100
4	0.66	0.49	0.56	100
5	0.60	0.66	0.63	100
6	0.57	0.61	0.59	100
7	0.64	0.54	0.58	100
8	0.54	0.46	0.50	99
9	0.52	0.57	0.54	99
acc			0.56	992
m_avg	0.56	0.56	0.56	992
w_avg	0.56	0.56	0.56	992

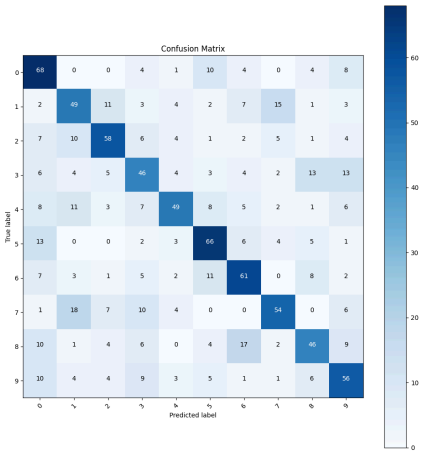

Figure 41, 42. Classification report, confusion matrix (difficult, NO preprocess)

4. Conclusion

In this study, we have presented a comprehensive approach to enhancing the recognition accuracy of CAPTCHA images using advanced image preprocessing techniques, followed by Convolutional Neural Network (CNN) based recognition. Our investigation focused on the implementation of a sequential preprocessing pipeline with image processing methods.

The effectiveness of these preprocessing techniques was evident in the substantial improvement in recognition accuracy achieved by the CNN model. Our results showed an increase in accuracy roughly ranging from 5% to 25% post-processing. This improvement was particularly marked in CAPTCHAs characterized by high levels of noise, distortion, and character overlap, elements traditionally challenging for standard recognition systems.

This work contributes to the ongoing development of more efficient and effective automated systems for CAPTCHA solving. It has implications for both the security domain, in understanding the robustness of CAPTCHAs against sophisticated attacks, and the accessibility field, in improving automated assistance for users who struggle with CAPTCHA verification. In conclusion, our study provides valuable insights and a solid foundation for future research aimed at further advancing the state-of-the-art in CAPTCHA recognition technology.

Reference

[1] R. Gossweiler, M. Kamvar, and S. Baluja, "What's Up CAPTCHA? A CAPTCHA Based on Image Orientation," in Proc. 18th Int. Conf. World Wide Web (WWW '09), Apr. 2009, pp. 841–850. Available: https://dl.acm.org/doi/pdf/10.1145/1526709.1526822

[2] E. Bursztein, M. Martin, and J. C. Mitchell, "Text-based CAPTCHA Strengths and Weaknesses," in Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS '11), Oct. 2011, pp. 125–138. doi: 10.1145/2046707.2046724. Available: http://www.decom.ufop.br/menotti/rp142/sem/sem1-dp1-artigo.pdf

[3] N. Şengöz, T. Yiğit, Ö. Özmen, and A. H. Işı, "Importance of Preprocessing in Histopathology Image Classification Using Deep Convolutional Neural Network," in Journal of Medical Imaging and Health Informatics, vol. 2, no. 1, pp. 1-6, Feb. 16, 2022. ISSN 2757-7422, doi: 10.54569/aair.1016544 Published online: Feb 16, 2022. Available: https://dergipark.org.tr/en/download/article-file/2053595

[4] M. Salvi, U. R. Acharya, F. Molinari, and K. M. Meiburger, "The impact of pre- and post-image processing techniques on deep learning frameworks: A comprehensive review for digital pathology image analysis," Computers in Biology and Medicine, vol. 128, Jan. 2021. DOI: 10.1016/j.compbiomed.2020.104129. PMID: 33254082. Available online: https://www.sciencedirect.com/science/article/pii/S0010482520304601

[5] R. Gonzalez and R. Woods *Digital Image Processing*, Addison-Wesley Publishing Company, 1992, pp 518 - 548.

[6] E. Davies *Machine Vision: Theory, Algorithms and Practicalities*, Academic Press, 1990, pp 149 - 161.

[7] R. Haralick and L. Shapiro *Computer and Robot Vision*, Vol 1, Addison-Wesley Publishing Company, 1992, Chap. 5, pp 168 - 173.

[8] A. Jain *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989, Chap. 9.

[9] D. Vernon *Machine Vision*, Prentice-Hall, 1991, Chap. 4.

[10] Y. Zhu and C. Huang, "An Improved Median Filtering Algorithm for Image Noise Reduction," Physics Procedia, vol. 25, pp. 609-616, 2012. DOI: 10.1016/j.phpro.2012.03.133. Available online: April 12, 2012. License: CC BY-NC-ND 3.0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1875389212005494

[11] P. Maragos, "Morphological Filtering," in The Essential Guide to Image Processing, 2nd ed., 2009, pp. 293-321. DOI: 10.1016/B978-0-12-374457-9.00013-5. Available online: July 27, 2009. National Technical University of Athens. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123744579000135

[12] D. J. Williams and M. Shah, "A Fast Algorithm for Active Contours and Curvature Estimation," CVGIP: Image Understanding, vol. 55, no. 1, pp. 14-26, Jan. 1992. Received: Mar. 8, 1991; Accepted: May 15, 1991. Available online: Nov. 29, 2004. Department of Computer Science, University of Central Florida, Orlando, Florida 32816, USA. [Online]. Available: https://www.sciencedirect.com/science/article/pii/104996609290003L

[13] R. Chauhan, K. K. Ghanshala, and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," in 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), 2018, pp. 1-6. DOI: 10.1109/ICSCCC.2018.8703316. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8703316&tag=1