CAPTCHA Recognition System

Author: Harsha. Contact: +91-7019065110. E-mail: harshar2025@gmail.com

The CAPTCHA Recognition System is a deep learning task hosted on GitHub. Its primary purpose is to automatically recognize and solve image-based CAPTCHAs. CAPTCHAs typically feature distorted text, numbers, or symbols that must be correctly identified to pass the verification. Given data has a fixed 200x50 dimension images with .csv file containing labels with only numerical values with a fixed size of 6, unlike generic CAPTCHAs. This report goes in details of the Approach taken, Challenges faced with how it was addressed, Performance Evaluation, Model Strengths and where it can be improved.

Approach

Understanding the dataset is the most important part of any Deep Learning task. It sets the training environment and output goals. Here, the given data and the goal brings up two main challenges.

1. Data Preprocessing

   Image Data - Images provided has heavy salt and pepper noise, numbers and noise with matching pixel values and a decent amount of distortion.
   Label Data - A simple .csv file with Image paths and number labels, which can be easily extracted with pandas.

   Main approach for Image Data:
   a. Remove salt and pepper noise.
   b. Distinguish pixel intensity values from numbers and noise.
   c. Converting Image to binary as in Black for number and White for background.
   d. Leave Image Distortion as it is because the numbers were readable and did not lose its shape.

2. Accurate sequential recognition with reasonable use of hardware resources

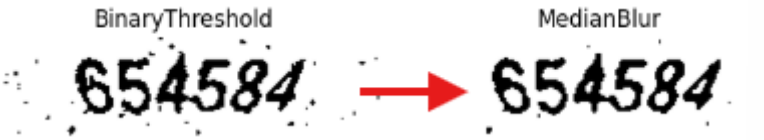
   a. The convolutional layers and fully connected layers are shared across all digits, meaning the model learns a general representation of the entire image before specializing for each digit.
   b. This assumes the digits are spatially correlated in the image (e.g., aligned horizontally).
   c. Each digit gets its own classifier, allowing the model to predict a fixed length sequence (6 digits). This is simpler than using a recurrent network (e.g., RNN) or CTC (Connectionist Temporal Classification) for variable length sequences, which reduces the need for high-end GPUs.
   d. In Output Head, six separate linear layers, each taking the 512-dimensional feature vector and outputting 10 logits (one for each digit class, 0–9)).

Challenges and Solutions

While the approach is technically sound, several limitations arise from the design and scope:

1. Choosing the right pre-processing step after applying MedianBlur which reduced sharpness
   Solution- To increase the image sharpness, the best method was to apply Threshold and convert 0 to 255 values to just 0 or 255. After comparing all the Threshold methods, Binary threshold was the most effective one as the pixel intensity range were consistent among all the images.

2. Dealing with thicker noise after applying Binary Threshold
   Solution- To reduce thick noise, the best method was to simply apply MedianBlur again to the Binary image and remove major noise. This method made sure that the image did not lose its sharpness as it was Binary.



3. Choosing the right architecture with the right balance on accuracy and efficiency
   Solution- Since the objective was to recognise 6 digits, which is a fixed number. It made no sense to apply higher level architecture or even do Transfer Learning as the pre-processed image had just two intensity values and a single channel making it a generic feature problem with low level features. So, building a classifier from scratch with 6 output nodes and making it spatially dependent was the right approach.

4. Dealing with low accuracy because of fewer convolutional layers
   Solution- To build a lightweight model, it is very important to choose just enough convolutional layers for the task but that resulted in extracting lesser features across the image. Since the model was heavily dependent on spatial features, an additional convolutional layer of 128 channel and dropout of around 0.4 had to be added to the architecture, to improve performance.

5. Choosing the right way to evaluate the model performance
   Solution- Classification task with no localization brings in a simple way to evaluate the model performance, that is to calculate the accuracy of whole CAPTCHA. But this does not properly guide us to visualize how the model is learning or performing better. To understand the effects of any change in the architecture or hyperparameter, evaluating the per digit accuracy and location of that digit accuracy helped in better tuning of the model resulting in significant performance leaps.

Performance

The model's overall performance is summarized by the following metrics:

- **Accuracy: 81.60%**
- **Precision: 81.67%**
- **Recall: 81.60%**
- **F1-Score: 81.62%**

The model correctly identifies 81.60% of all CAPTCHA sequences, meaning it gets the entire 6-digit sequence right in roughly 8 out of 10 cases. The close alignment of accuracy, precision, recall, and F1-score (all around 81.6%) indicate consistent performance with no significant imbalance between false positives or false negatives. However, the 18.4% error rate implies that nearly 1 in 5 CAPTCHAs is misrecognized, which leaves space for improvement.

Per-Digit Positional Accuracy

The accuracy for each digit position in the 6-digit CAPTCHA sequence is as follows:

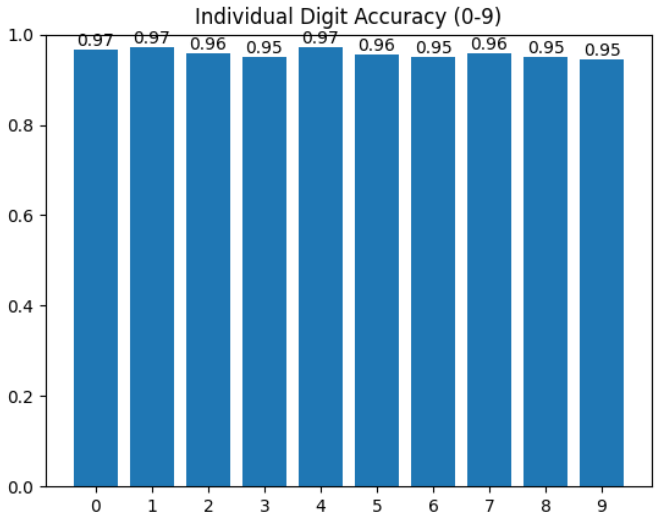
Digit 1: 97.85%, Digit 2: 96.45%, Digit 3: 95.35%

Digit 4: 94.55%, Digit 5: 93.80%, Digit 6: 96.80%

The model performs strongly on a per-digit basis, with accuracies ranging from 93.80% (Digit 5) to 97.85% (Digit 1). There is an observable pattern which shows that the digit position 2-5 have comparatively low performance. It might be the architecture of the model where it recognises the digit through spatial features of nearby digits, the start and end digits will have lesser noise for the model to recognise, thus better accuracy.

Individual Digit Accuracy

The accuracy for recognizing individual digit (0–9) across all positions is:



Digits 4 (97.31%) and 1 (97.13%) are the most accurately recognized, while Digit 9 (94.60%) and Digit 6 (94.92%) are the least accurate. These differences are minor but could reflect subtle challenges, such as visual similarity (e.g., 9 vs. 6 or 8) or distortions in the CAPTCHA images. Overall, the high individual digit accuracies (all above 94%) reinforce that the system excels at single-digit recognition, and the lower full-sequence accuracy (81.60%) likely results from cumulative errors across the six-digit sequence rather than poor digit-specific performance.

Limitations

1. Model cannot perform well outside of training environment challenges like with a unique noise, pixel intensities and distortion.
2. Model cannot be used with captcha of variable lengths.
3. Model cannot be used to localize the digits.
4. Since the features of a Binary Digits are not high level, model can be at overfitting risk at higher epochs.
5. Model struggles to get high accuracy with this architecture even with higher individual digit accuracy.

Potential Improvements

1. **Advanced Noise Reduction:** Experiment with adaptive filtering or morphological techniques to better eliminate noise.
2. **Better Architecture:** Incorporate additional convolutional layers or explore residual connections to capture complex features.
3. **More Regularization:** Implement advanced version of regularization such as weight decay((L2) to penalise and prevent similar mistakes like recognizing 6 for 8 or 9.
4. **Data Augmentation:** Apply varied augmentation strategies to increase robustness against unseen noise and distortions.
5. **Use Advanced Architecture:** Use architectures where it supports sequential classification to make it variable length recognising model.

Conclusion

In conclusion, the CAPTCHA Recognition System effectively demonstrates how targeted preprocessing, lightweight architecture, and positional evaluation can significantly enhance digit sequence recognition from noisy CAPTCHA images. Despite challenges such as salt-and-pepper noise, distortion, and limited pixel intensity variation, the model achieves a commendable overall accuracy of 81.60% for full 6-digit sequences, with individual digit accuracy consistently above 94%. The architecture’s fixed-output design ensured efficient training and inference on limited hardware. Although not suited for variable-length CAPTCHAs or localization tasks, this approach proves highly effective within its defined constraints. The evaluation strategy, especially per-digit and positional accuracy analysis, provided valuable insights into the model's behaviour, guiding meaningful improvements.