Enhancing CAPTCHA Accessibility for Dyslexic and Visually Impaired Users Using Neural Networks

...

Prepared by:- Group 127

Rahil Hardik Shukla (rshukla7)

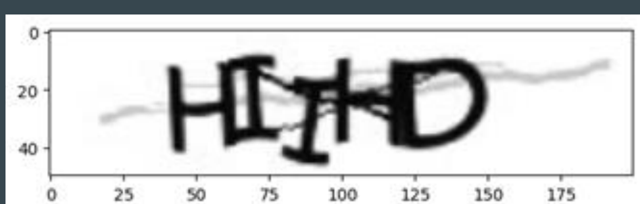
North Carolina State University

Motivation

- CAPTCHAs are essential for web security but hinder accessibility.
- Users with dyslexia struggle with distorted, overlapping or jumbled text.
- Colorblind users face challenges when CAPTCHAs rely on color differentiation.

Problem Statement

- Most CAPTCHA solvers focus on bypassing them for automation or security testing.
- This project aims to help users with accessibility issues solve CAPTCHAs without compromising security.



H I I H D

Input

Output

Dataset Overview

- Datasets sourced from Kaggle and synthetically generated using Python libraries.
- Includes 113,062 CAPTCHA images with various distortions, fonts and colors.
- Augmentation applied: rotation, skewing and contrast changes.
- Designed to simulate real-world CAPTCHA challenges for dyslexic and colorblind users.

Grayscale Conversion

- Converting RGB images consisting of three color channels — Red, Green, and Blue to grayscale images containing a single intensity channel representing light brightness.
- Why is it done

  1. CAPTCHA text relies on shape and not color

  2. Reduces input channels from 3 (RGB) to 1 (grayscale)

- This will not only speed up the computation reducing the training time but also simplifies learning by focusing on structure and not color variance

Denoising & Binarization

- Denoising removes unwanted visual artifacts (like background noise) from an image. Binarization converts an image into black and white to sharply distinguish the foreground from the background

  Why is it done

1. CAPTCHA images often contain visual noise such as background clutter, curves, and dots that make character recognition harder.
2. Gaussian blurring smooths out small-scale artifacts, while thresholding sharpens the separation between text and background.

Impact on Model Training: Applying Gaussian blur helps remove unwanted pixel-level noise that could confuse the model during learning. Following this with binarization creates clear boundaries between foreground characters and background, which strengthens the model's ability to detect character shapes. Together, these steps improve the clarity of the input, leading to more accurate feature extraction and better generalization during training.

Resize & Normalization

<u>Steps:</u>

-Resize images to fixed dimensions (100x50)

-Normalize pixel values to range [0, 1]

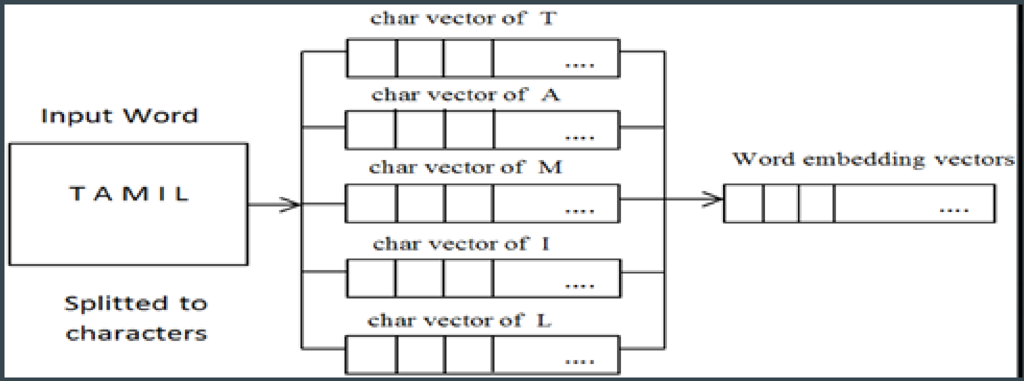
<u>Advantages</u>

1. Resize ensures that all inputs to the model are consistent in shape, allowing batch processing and simplifying model architecture
2. Normalization helps neural networks converge faster and prevents instability in gradient descent

- By resizing all images to a uniform size, the model can efficiently process input data without needing to handle variable shapes. Normalizing pixel values ensures that all features contribute proportionally during training, resulting in faster convergence and more reliable learning.

Result of Preprocessing

Label Encoding via Character Embeddings



- Converting each character to an integer and use an embedding layer
- Doing this is way more efficient that one-hot encoding. Moreover, embeddings allow model to learn character similarities.
- By utilizing character embeddings the model can generalize better to unseen words and capture subword information, leading to improved performance in tasks involving noisy or complex textual data. This enhances the model's ability to learn meaningful patterns from limited data.

Baseline Model Description

Model Architecture:

- Input layer: Grayscale image of shape (50, 100, 1)
- Two convolutional layers with ReLU activation
- Max pooling after each convolution layer
- Flatten layer to convert feature maps to vector
- Fully connected dense layer with dropout regularization
- Five parallel dense output layers with softmax activation (one per character)

This architecture is a balance of simplicity and effectiveness. Convolutional layers are well-suited for image data allowing the model to extract hierarchical spatial features. By using shared base layers and multiple output heads the model is capable of predicting all five characters in the CAPTCHA image efficiently. Moreover, Its lightweight structure ensures quick training even on limited data.

Methodology & Evaluation Metrics

Methodology:

The baseline CNN model was trained on labeled CAPTCHA images and evaluated on a separate set of test images. Each image contained a fixed-length 5-character alphanumeric code. The model predicts all five characters simultaneously using a multi-output softmax structure. The evaluation was conducted to assess both individual character accuracy and full CAPTCHA string accuracy.

Metrics Used:

- Per-character classification accuracy
- Final CAPTCHA accuracy (all 5 characters correct)
- Partial CAPTCHA accuracy (first 3 characters correct)

Evaluation Table & Results

Metric	Accuracy
Average Character-wise Accuracy	87%
Final 5-character CAPTCHA Accuracy	68%
First 3-character CAPTCHA Accuracy	79%

- The model performs well on character-level predictions achieving an average accuracy of around 87%. While exact 5-character predictions are more challenging the model still achieves a strong 68% success rate with even better performance (79%) when considering just the first three characters.
- The model shows high confidence and reliability in predicting the first three characters. However, the difficulty increases with the final two characters leading to reduced overall accuracy. This indicates the need for further refinement. In the next phase, we will fine-tune this model to improve its ability to accurately predict all five characters and raise the overall CAPTCHA success rate.

References

1. CAPTCHA Image Dataset - Kaggle: https://www.kaggle.com/datasets/parsasam/captcha-dataset

2. Python CAPTCHA Library: https://github.com/lepture/captcha