Breaking CAPTCHA: A Deep Learning Approach

[Adithya Addepalli Casichetty]

February 10, 2025

1 Introduction

This report presents our approach to developing a deep learning system capable of breaking CAPTCHA images. I tackled this challenge through two main tasks: a classification task with a fixed set of words, and a more complex generation task capable of handling arbitrary text.

2 Task 1: Classification

2.1 Problem Definition

The first task involved classifying CAPTCHA images into one of 100 predefined classes.

2.2 Model Architecture

I implemented a Convolutional Neural Network (CNN) architecture consisting of:

- Three convolutional blocks, each containing:
  - 2D Convolutional layer
  - ReLU activation
  - MaxPooling layer
  - Batch Normalization
- Fully connected layers for classification

2.3 Training Process

The model was trained using:

- Cross-Entropy Loss

- Adam optimizer with learning rate 0.001
- Batch size of 32
- 50 epochs

2.4 Results

After 50 epochs of training, the model achieved a training accuracy of 86.97% and a validation accuracy of 93.57%. The training accuracy is lower than the validation accuracy due to the strong augmentation applied for the training dataset, which is absent in the validation dataset.

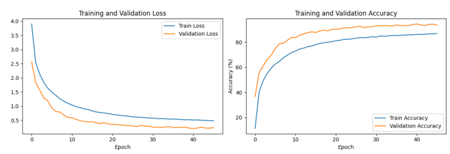

Figure 1: Training and Validation Loss and Accuracy over Epochs (Task 1)

3 Task 2: Generation

3.1 Problem Definition

The second task involved a more challenging scenario: generating arbitrary text from CAPTCHA images without being restricted to a predefined set of words. This required a more sophisticated architecture,

3.2 Model Architecture

I developed a hybrid CNN-RNN architecture:

3.2.1 CNN Component

Similar to Task 1, but optimized for feature extraction:

- Three convolutional blocks with increasing channel depth ($32 \rightarrow 64 \rightarrow 128$)
- Batch normalization after each block
- Feature dimensionality reduction through linear projection

3.2.2 RNN Component

Added sequence generation capabilities:

- LSTM layer for sequence modeling
- Character embedding layer
- Output projection layer for character prediction

3.3 Observations

Key Observations:

- I had initially used extensive dropouts and enhanced data augmentation(as I had done in Task 1)to the training dataset in the hopes of avoiding overfitting.This however lead to abysmal results, with the character accuracy stagnating at around 20% after the 15th epoch.
- After simplifying the architecture by applying the same basic augmentation for both training and validation dataset and removing dropouts, the accuracy shot up.
- Another observation I made is that the model for task 2, unlike task 1, reaches a high character accuracy in lesser number of epoch cycles compared to the model from task 1. I decided to reduce number of epoch cycles from 50 to 30 due to this

3.4 Results

After 30 epochs of training, the model achieved a character level accuracy of 86.47% and a validation accuracy of 65.58%.

3.5 Error Analysis

Common errors observed:

- The model is able to analyze the letters in Captcha, however, it is unable to predict the length as a result of which extra characters are getting padded

```python
def test_single_image(model, image_path, transform, device):
    model.eval()

    image = Image.open(image_path).convert('RGB')
    image = transform(image).unsqueeze(0).to(device)

    with torch.no_grad():
        output = model(image)
        _, predicted = output.max(2)

        pred_word = ''.join([train_dataset.idx_to_char[idx.item()]
                            for idx in predicted[0]])

        print(f"Predicted word: {pred_word}")


test_single_image(model, "0_CHaNCe.png", transform, device)
```

```
Predicted word: CHaNCeeeee
```

Figure 2: Incorrect length prediction

4 Discussion

4.1 Comparative Analysis

The generation model (Task 2) demonstrated superior flexibility compared to the classification model (Task 1), albeit with increased complexity in both architecture and training. While Task 1 achieved higher accuracy within its limited scope, Task 2's ability to handle arbitrary text makes it more practical for real-world applications.

4.2 Challenges and Solutions

Key challenges encountered:

- Variable-length output handling