Report on CRNN-Based CAPTCHA Recognition System

1. Approach to Solving the Problem

The problem was approached using the following steps:

Dataset Preparation & Preprocessing:

A dataset of CAPTCHA images and their corresponding labels was used.
Labels were ensured to be 6-digit numerical values, padded with leading zeros if necessary.

Generate more data with augmentation

Images were converted to grayscale, normalized, and resized to a fixed size (200x50 pixels).

Data augmentation techniques, including rotation, Gaussian blur, and random affine transformations, were applied to improve generalization.

Model Architecture:

A CRNN (Convolutional Recurrent Neural Network) was designed:

CNN layers for feature extraction.
LSTM layers for sequence learning.
Fully Connected layer for classification.
CTC Loss (Connectionist Temporal Classification) was used to handle variable-length sequences without requiring character-level alignment.

Training Process:

The model was trained using Focal CTC Loss, an improved variation of CTC loss that helps stabilize training by reducing the impact of easy-to-classify samples.

The optimizer used was AdamW with a learning rate of 0.00005, including weight decay for better generalization.

Evaluation:

The model was evaluated on a separate validation dataset.
Performance was measured using validation loss and accuracy (correctly predicted CAPTCHA texts).
Images were saved with ground truth and predicted labels for error analysis.

2. Key Challenges Faced and Solutions

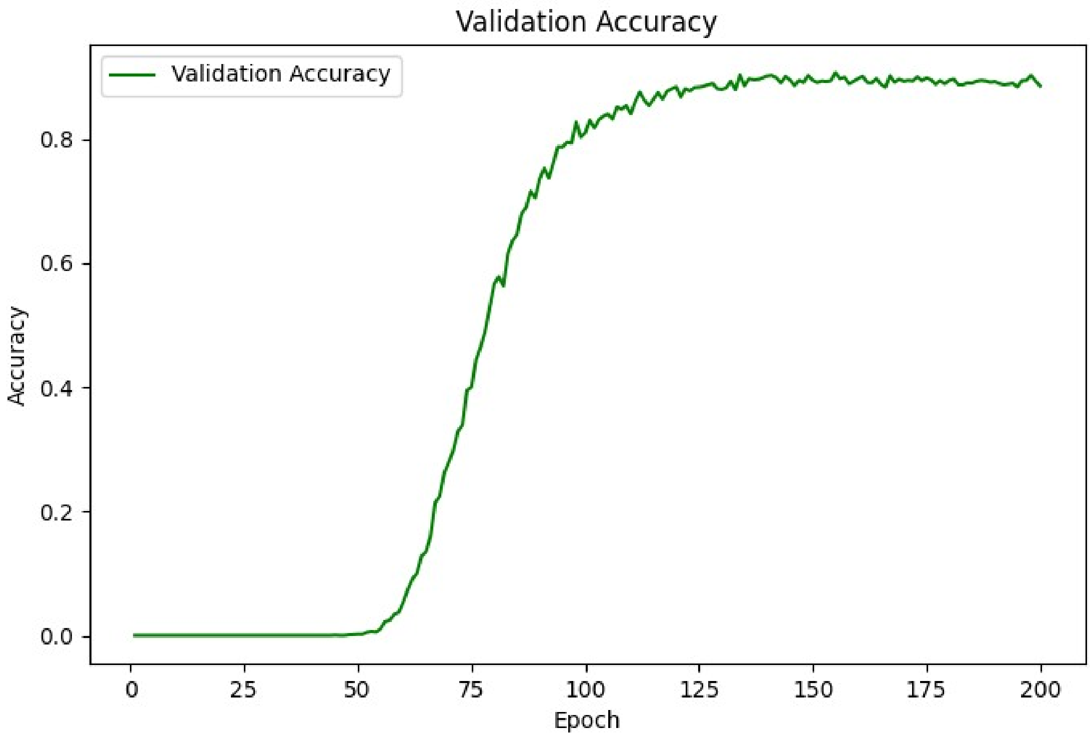
1. Overfitting on Training Data
Challenge: The model initially performed well on training data but poorly on validation data, indicating overfitting.
Solution:
Applied data augmentation (rotation, blur, and random affine transformations).
Used dropout layers in the CNN and LSTM to regularize the model.
Implemented weight decay (1e-3) in the AdamW optimizer to improve generalization.

2. Training Instability (Exploding or Vanishing Gradients)
Challenge: During training, gradients occasionally became unstable, leading to NaN loss values.
Solution:
Used gradient clipping (max_norm=5.0) to prevent exploding gradients.
Applied Focal CTC Loss, which helps in stable loss computation.
Ensured input lengths and target lengths were correctly computed to avoid CTC misalignment issues.

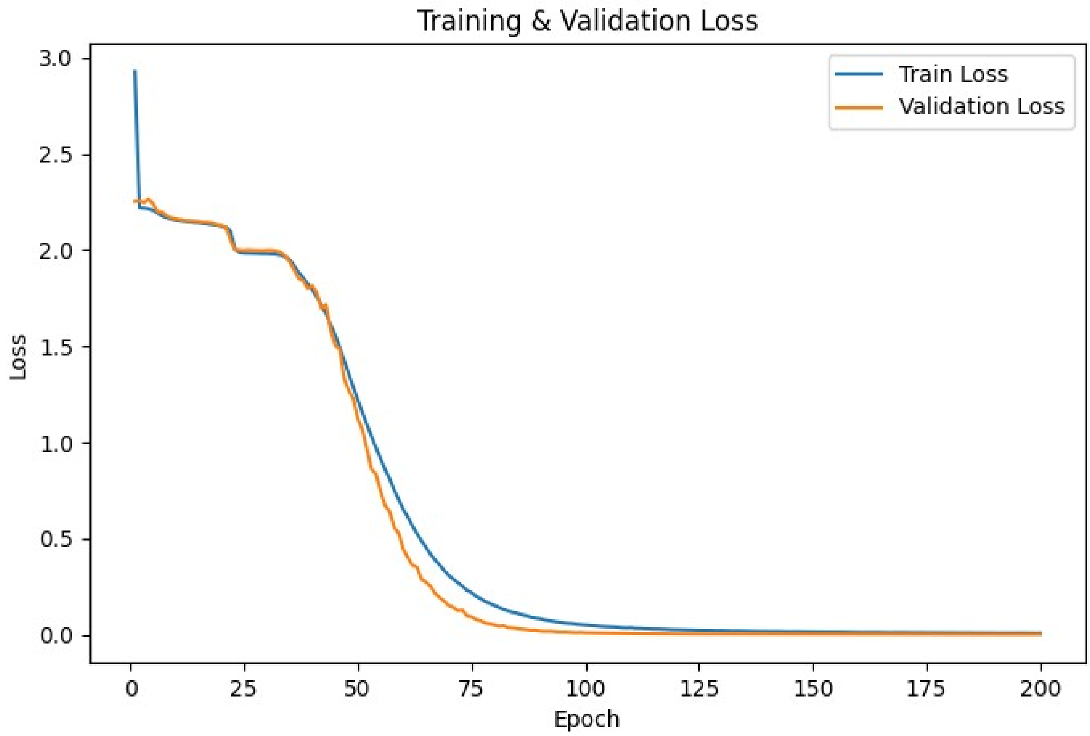
3. Handling Captchas with Different Noise Levels
Challenge: Some captchas had additional distortions, making recognition difficult.
Solution:
Introduced Gaussian blur augmentation to make the model robust to blurry images.
Used random affine transformations to simulate different distortions.

3. Performance Metrics and Graphical Representation

Validation Accuracy
The model achieved an accuracy of 89.35% on the validation dataset, meaning it correctly predicted the full 6-digit CAPTCHA for 89.35% of images.



Training & Validation Loss Graph
The following graph shows the loss trend over training epochs:



Training Loss: Starts high and gradually decreases.
Validation Loss: Shows a stable decline, confirming generalization.

Future Improvements
Implement transformer-based OCR models for better accuracy.
Fine-tune hyperparameters using automated tuning strategies.
Introduce synthetic CAPTCHA generation for more robust training.

Search