HANDWRITING RECOGNITION

TEAM MEMBERS

01 ADITHYA
CB.SC.U4AIE24005

02 ARCHITHA RAJASEKAR
CB.SC.U4AIE24009

03 BHUVANESWARAN S
CB.SC.U4AIE24012

04 HARSHITH REDDY
CB.SC.U4AIE24018

PROJECT OVERVIEW

Goal:

Develop a deep learning model that accurately recognizes handwritten text. The model combines Convolutional Neural Networks (CNN) for spatial feature extraction with Recurrent Neural Networks (RNN) for sequence modeling, and utilizes Connectionist Temporal Classification (CTC) loss to handle unsegmented and variable-length sequences.

Introduction

- ❑ Handwriting recognition is the technology that allows computers to interpret and convert handwritten text into digital, machine-readable form.

- ❑ It involves processing and analyzing the shapes, strokes, and patterns in handwritten input—whether from scanned documents, images, or real-time digital pen strokes—to accurately identify the characters and words written by a human.

- ❑ Neural networks, particularly when combined with Recurrent Neural Networks (RNNs) are well-suited to process sequential data and can effectively learn temporal dependencies.

- ❑ This capability makes them robust in recognizing characters in different handwriting styles and in various conditions (e.g., slanted, cursive, or disconnected strokes).

METHODOLOGY

1. Data Loading and Initial Visualization

- CSV Import:

The code begins by reading two CSV files containing labels:

A training labels file (train_labels.csv)

A validation labels file (written_name_validation_v2.csv)

Image Visualization:

It then uses matplotlib and OpenCV to display a few sample images along with their labels. This helps in quickly verifying that the data is loaded correctly.

- **Data Cleaning:**

It checks for any missing (NaN) values in both datasets and removes those rows.

Images labeled as "UNREADABLE" are filtered out from both training and validation sets.

All remaining identity labels are converted to uppercase for consistency.

METHODOLOGY

2. Preprocessing the Images

- Preprocess Function:

A custom function named preprocess is defined to standardize the images. The steps include:

Creating a blank white image of size 64×256

Cropping:

If the image width exceeds 256 pixels or the height exceeds 64 pixels, the image is cropped to fit the blank white sheet

- Padding:

If the image size does not exceeds 256 pixels or the height of 64 pixels,the image will fixed inside the white blank sheet

- Rotation:

The image is rotated 90 degrees clockwise to match the orientation expected by the model.

Normalization and Reshaping:

Each image is converted to grayscale, normalized by dividing pixel values by 255, and reshaped to have a shape of (256, 64, 1) (height, width, channels) before being added to the dataset.

METHODOLOGY

Converting Text to Numbers:
The handwritten names are converted into numbers based on a predefined alphabet. This helps the model work with the text as numerical data.

Building the Model:
The model is designed with two main parts:
A CNN (Convolutional Neural Network) extracts important features from the images.
A RNN (Recurrent Neural Network), using LSTM layers, understands the sequence of letters from these features. The model ends with a softmax layer that outputs probabilities for each character.

METHODOLOGY

Model Architecture:

1. Input Layer

Shape: (64,256, 1) → This represents grayscale images of size 64×256 pixels.

This layer receives the preprocessed images and passes them to the next layers.

2. Convolutional Layers (Feature Extraction)

Purpose: Extract essential features (edges, curves, and shapes) from the handwriting.

Process: A 3×3 filter slides over the image, detecting patterns.

Activation Function: ReLU (Rectified Linear Unit), which allows only positive values, making learning faster and avoiding negative outputs.

- $f(x)=\max(0,x)$

METHODOLOGY

Batch Normalization

- **Purpose:** Normalizes activations in the network to speed up training and improve stability.
- Helps in preventing vanishing or exploding gradients.

MaxPooling Layers

- **Purpose:** Reduces the image size while keeping the most important information.
- **Process:** A **2×2 filter** selects the most important features, reducing computations.

Dropout Layers

- **Purpose:** Prevents **overfitting** by randomly disabling some neurons during training, ensuring the model doesn’t memorize training data.

METHODOLOGY

3. Reshape Layer (CNN to RNN Conversion)

- The extracted features are converted into a sequence format for the RNN.
- Shape Change: Converts (Batch Size, Height, Width, Features) → (Batch Size, Time Steps, Features).
- This prepares the data for the LSTM layers.

4. Recurrent Layers (Text Sequence Prediction)

- Bidirectional LSTM (Long Short-Term Memory)
- Purpose: Understands sequential relationships between extracted features.
- Bidirectional: Reads the sequence from both directions (left to right and right to left) to capture more information.

METHODOLOGY

5. Dense (Fully Connected) Layer

Purpose: Converts LSTM outputs into probabilities for each possible character.

Activation: Softmax, which gives a probability distribution for each character.

6. Connectionist Temporal Classification (CTC) Loss

- Purpose: Handles variable-length predictions (since words may have different lengths).
- How it works: Instead of requiring fixed alignment between input and output, it optimizes based on the most likely character sequence.
- Helps in training the model without needing exact positions of each character.

7. Training and Prediction

- The model is trained using the Adam optimizer to minimize the CTC loss.
- Predictions are decoded using CTC decoding, which converts the network’s output into readable text.

METHODOLOGY

Evaluation metrics

- Character-Level Accuracy:

Measures the overall accuracy of each character in the predicted sequence compared to the ground truth.

- Word-Level Accuracy:

Evaluates the accuracy at the level of complete words or labels, requiring an exact match between prediction and ground truth.

Testing on New Data:

Finally, the trained model is tested on a new set of images to see how well it can recognize handwriting in unseen data.

FUTURE WORK

1. **Enhancing Model Accuracy**

• Adjust hyperparameters like learning rate, batch size, and number of LSTM layers to increase accuracy.

• Try a variety of optimizers like SGD with momentum or RMSprop.

2. Leveraging an Expanded Training Dataset

•Currently, at least 30,000 training images are used. The dataset can be expanded by introducing more handwritten samples to further improve performance.

3. Attention Mechanism Usage

- The addition of an attention mechanism to the LSTM architecture allows the model to focus on important areas of the input image, thus improving recognition accuracy.

4. Noise or Poor Image Processing

- Use image denoising techniques, such as Gaussian Blur, Median Filtering, or Autoencoders, to enhance the resolution of low-resolution handwriting images.

5. Data Augmentation

•Apply transformations such as rotation, scaling, contrast modification, and Gaussian noise to diversify the dataset and enhance model generalization.

Thanks!