Handwritten Text Identification and Recognition from an Image

Swagat Kumar Tripathy

🧠**Abstract.** This project presents a deep learning pipeline that automates the identification and recognition of handwritten text from scanned images using a Convolutional Recurrent Neural Network (CRNN). The model combines CNN-based spatial feature extraction with RNN-based sequence modeling and Connectionist Temporal Classification (CTC) loss for robust recognition without requiring character-level segmentation. We trained and evaluated the system on a subset of the IAM Handwriting Dataset. Results demonstrate a character-level accuracy of 95.86% and a word-level accuracy of 92.40%, with potential for further improvement using beam decoding, attention mechanisms, and stronger CNN backbones.

📖 Introduction

Text recognition from handwritten inputs is a classical problem in computer vision, OCR (Optical Character Recognition), and document digitization. Unlike printed text, handwriting varies significantly in shape, alignment, spacing, and orientation. This makes rule-based OCR methods unreliable in real-world applications.

The proposed system addresses this challenge by integrating deep learning components: Convolutional Neural Networks (CNNs) to extract spatial features and Bidirectional LSTMs (BiLSTM) to model temporal sequences. To align image features with variable-length label sequences, we use CTC (Connectionist Temporal Classification), a loss function that enables training without requiring character-level alignment.

The goal of this project is twofold:

- **Identification:** Determine whether the handwritten content is an alphabet, number, word, sentence, or special character.
- **Recognition:** Accurately predict the handwritten content in text format.

This work enables downstream applications in digitizing old manuscripts, automating form reading, and real-time handwriting interpretation.

📂 Dataset

Source: IAM Handwriting Dataset (subset)

Image Format: 32px height, 256px width, grayscale .png files

Label Format: Located in words.txt, each line contains the filename and transcription

Example label: a01-000u-00-00 ok 154 408 768 27 51 AT A

Ground-truth label: "AT A"

Preprocessing Steps:

- Convert images to grayscale.
- Resize each image to (256 × 32).
- Normalize pixel values to range [0, 1].
- Expand image dimensions to (32, 256, 1) for CNN input.
- Encode character labels using a defined char_list.
- Filter out samples where label length > number of CNN time steps (~63).

⚙️ Model Architecture: CRNN with CTC

1. 📷 CNN: Convolutional Feature Extractor

- 7 Convolutional layers (most with 3×3 kernels)
- Increasing filters: 64 → 512
- MaxPooling to reduce spatial dimensions
- BatchNormalization for faster convergence

2. 🔁 RNN: Recurrent Sequence Modeler

- Two Bidirectional LSTM layers with 256 units each
- Output sequence represents character probabilities at each time step

3. 🔤 Output Layer + CTC Loss

- Dense layer with softmax activation
- Output shape: (batch_size, time_steps, num_classes)
- Loss computed via keras.backend.ctc_batch_cost

🧮 Loss Function (CTC)

CTC enables mapping variable-length input sequences to variable-length outputs without explicit alignment. It solves the problem of repeated characters and unknown alignment between time steps and labels.

The CTC loss takes:

- y_pred (logits after softmax),
- y_true (encoded labels),
- input_length (fixed),
- label_length (variable)

This allows flexible training even when the number of time steps ≠ number of characters.

🏋️ Training Strategy

Parameter	Value
Optimizer	Adam (lr = 0.0001)
Epochs	50
Batch Size	16
Input Size	32×256×1
Validation Split	10%

⚙️ Data Augmentation

- Rotation (±2°)
- Zoom (±2%)
- Width/Height shift (±2%)

🔁 Callbacks

- **ModelCheckpoint:** Save best weights based on validation loss
- **EarlyStopping:** Stop if no improvement after 5 epochs

📈 Evaluation Metrics

Two evaluation metrics are used:

- **Character-Level Accuracy:** Based on normalized edit distance (Levenshtein distance)
- **Word-Level Accuracy:** Percentage of exact matches between predicted and true strings

Metric	Result
Character Accuracy	95.86%
Word Accuracy	92.40%

Evaluation used **beam search decoding** with beam_width = 10

This improves over simple greedy decoding.

⚠️ Limitations

- Labels must not exceed the model's output time steps.
- Model is trained only for English handwritten words.
- Performance drops for highly cursive or noisy inputs.
- Requires well-aligned, horizontal text lines.

🚀 Enhancement Scope

- **Stronger CNN Backbone:** Replace manual CNN with MobileNetV2 or EfficientNet.
- **Attention-based Decoder:** Add Transformer-style decoders for more accurate alignment.
- **Language Model Integration:** Post-process decoded text using a character-level language model.
- **Support for Multi-Line and Paragraph Text:** Extend current line-level recognition to paragraphs.
- **Cross-Language Support:** Train on multilingual datasets (e.g., Devanagari, Arabic).
- **Real-Time Application:** Convert this model to serve in real-time handwriting input via stylus/tablet.

✅ Conclusion

This project successfully implements a CRNN-based approach for recognizing handwritten text from raw image input. The system combines CNN for spatial feature extraction, BiLSTM for sequence modeling, and CTC loss for training on unsegmented data.

The architecture shows strong potential and flexibility due to high accuracy. With attention-based decoding, stronger feature backbones, and language-aware post-processing, the performance can be significantly improved, making it suitable for industrial-scale document automation, historical document digitization, and assistive technologies for the visually impaired.

📚 References

[1] IAM Handwriting Dataset: http://www.fki.inf.unibe.ch/databases/iam-handwriting-database

[2] CRNN Paper: https://arxiv.org/abs/1507.05717

[3] TensorFlow OCR Tutorials: https://www.tensorflow.org/tutorials/images

[4] Editdistance Library: https://github.com/aflc/editdistance

[5] TrOCR (Transformer OCR): https://arxiv.org/abs/2109.10282