

Handwritten Text Recognition Using CNN-RNN Hybrid Model

Naman Shah¹, Lalit Purohit², Mukesh Sakle³

¹M.Tech, Research Scholar, Information Technology, SGSITS, Indore

²Professor, Information Technology, SGSITS, Indore

³Assistant Professor, Information Technology, SGSITS, Indore

Abstract

Handwritten Text Recognition (HTR) has long stood as a challenging domain in computer vision and pattern recognition, particularly due to the variability in individual handwriting styles, noise in scanned documents, and differences in word structure. In this study, we present a CNN-RNN hybrid model enhanced with a Connectionist Temporal Classification (CTC) loss function, aimed at improving the accuracy and reliability of offline handwritten text recognition. By leveraging convolutional layers for spatial feature extraction and bidirectional LSTM layers for sequential modeling, our approach balances spatial and temporal understanding effectively. We trained our model using the IAM dataset, incorporating careful preprocessing and character filtering techniques. Experimental results demonstrate that our model achieves an accuracy of **87.5%** on a 200-sample validation set, a notable improvement over our baseline of **43.33%**. Compared to traditional techniques and several recent deep learning-based models, our pipeline maintains a strong performance while retaining architectural simplicity. We also analyze word-level precision, recall, and F1-score, and present a detailed comparison against recent advancements in the field. This work contributes a reliable and efficient HTR pipeline with room for improvement via ensemble learning and language model integration.

Keywords: Handwritten Text Recognition, CNN-RNN Hybrid, CTC Loss, IAM Dataset, Beam Search Decoding, Deep Learning, Sequence Modeling, Optical Character Recognition, LSTM

1. Introduction

Despite decades of research, accurate and robust recognition of handwritten text continues to present significant challenges. Unlike printed text, handwriting is inherently variable—differing across individuals, writing instruments, styles, and even emotional states. This inconsistency introduces significant ambiguity, particularly in cursive scripts, slanted characters, overlapping strokes, and inconsistently spaced words. Furthermore, noise in scanned images, variation in document resolution, and the lack of standardized input formats exacerbate the difficulty of recognition tasks. Traditionally, approaches to HTR relied heavily on handcrafted features, rule-based segmentation, and shallow classifiers such as Hidden Markov Models (HMMs) or Support Vector Machines (SVMs). However, these methods often fell short when applied to large-scale or diverse datasets, as they lacked the capacity to generalize well to unseen styles or contextual variations. With the advent of deep learning, a paradigm shift occurred. Convolutional Neural Networks (CNNs) demonstrated exceptional performance in feature extraction from visual inputs, while Recurrent Neural Networks (RNNs), particularly Long Short-Term

Memory (LSTM) units, showed strong potential in sequence modeling and temporal context understanding. In our work, we explore a hybrid CNN-RNN architecture equipped with Connectionist Temporal Classification (CTC) loss. The convolutional layers are responsible for extracting rich spatial features from handwritten word images, while the bidirectional LSTM layers handle sequential dependencies and order-aware representations. CTC loss, which aligns input and output sequences without requiring manual segmentation, enables the model to learn mappings between image sequences and variable-length text labels. We trained and evaluated our model using the IAM offline handwritten dataset. The model was developed iteratively with particular attention to preprocessing, data filtering, and decoding techniques. Beam search decoding, as opposed to naive greedy methods, was adopted for output inference to further improve recognition quality. This paper presents our complete pipeline—from data handling and model architecture to training methodology and evaluation. We also compare our work with several state-of-the-art approaches published in recent years and identify areas where our method outperforms or complements existing techniques. In doing so, this research aims to contribute a strong yet accessible HTR solution that balances performance with practical deployment readiness.

2. Literature Review

In 2020, Neto et al. [10] proposed **HTR-Flor**, a modular deep learning system that successfully combined Convolutional Neural Networks (CNNs) with sequential modeling via RNNs for offline handwritten text recognition. Using the IAM dataset, their system emphasized flexibility and scalability. Our implementation draws inspiration from this modularity, allowing flexibility in training, evaluation, and model saving strategies, while using a similar CNN-RNN backbone for processing image-based input.

With the surge of deep learning in the early 2020s, de Sousa Neto et al. (2024) [1] conducted a **systematic literature review** focusing on **data augmentation strategies** in offline HTR. They compiled techniques such as affine transformations, elastic distortions, and noise injection and demonstrated their effectiveness in improving generalization and reducing overfitting. While our current system does not fully utilize synthetic augmentation techniques, the insights from their review strongly support our intention to integrate such strategies in future iterations to further enhance robustness.

Rakesh et al. (2024) [4] presented a **survey of deep learning approaches** in HTR, identifying CNN-BLSTM-CTC hybrids as the most promising structure. Their work confirmed that these models consistently outperform traditional segmentation-based systems, particularly in end-to-end pipelines. Our approach aligns directly with their conclusions, using a CNN-BiLSTM model trained with CTC loss for effective, segmentation-free recognition.

Alaei et al. (2023) [5] introduced a **noise-resilient hybrid CNN-RNN model**, demonstrating significant robustness in degraded inputs by injecting structured noise and applying denoising layers. Although our current model does not include explicit denoising components, we address input variability through preprocessing and conservative dropout regularization. Their findings indicate a strong case for adding robustness techniques in future versions of our model.

Fischer (2023) [7] focused on a CNN-LSTM hybrid specifically applied to the **Washington Database**. His insights on sequence reshaping and the stabilizing role of CTC loss influenced our architecture's reshaping logic and training methodology. Despite using different datasets, the performance dynamics observed in Fischer's work paralleled those we encountered, further validating our CNN-RNN choice.

Liu et al. (2023) [6] worked on **multilingual HTR using BLSTM**, focusing on the NIST and IAM datasets. Their architecture achieved promising accuracy ($\approx 89.5\%$) despite the challenge of multilingual

variability. While we focus solely on English text from the IAM dataset, their results affirm the scalability of BiLSTM-based systems across language boundaries, offering a potential extension to our work.

Zhang et al. (2023) [8] conducted an in-depth analysis of **BiLSTM depth and dropout regularization**, illustrating that stacked layers with optimal dropout improve generalization and reduce overfitting. Our system incorporates these suggestions by including two BiLSTM layers along with dropout, balancing complexity with learning stability.

Ghosh et al. (2023) [9] highlighted the effectiveness of **synthetic data augmentation** for CNN-RNN architectures, especially under limited real annotated data. Their use of the IAM and RIMES datasets revealed significant gains in accuracy ($\approx 88.9\%$) using augmented data. Although our training set is relatively modest, our future roadmap includes such data expansion to support broader handwriting variability. In the domain of explainable AI, **Azimi et al. (2023)** [2] developed a **CNN-RNN model with attention** to enhance transparency in online handwritten character recognition. Their model, trained on IAM and CVL datasets, showed high accuracy (94.2%) but targeted stylus-driven input. Our offline approach, while not equipped with attention mechanisms yet, benefits from simpler deployment without real-time constraints.

Jungo et al. (2023) [3] explored the capabilities of **Transformer-based segmentation** for online character queries, achieving high recognition rates (accuracy $\approx 95.6\%$). However, Transformer models are resource-intensive and require larger datasets for training. Given our limited compute resources, we adopted a more conservative CNN-BiLSTM structure while still achieving competitive accuracy and low error rates, making our model efficient and accessible.

Overall, this literature indicates that CNN-RNN hybrids, especially with BiLSTM and CTC loss, remain the most balanced choice for offline HTR systems — particularly when computational efficiency is essential. Our work positions itself in this space by delivering a modular, stable, and high-performing pipeline, achieving a test accuracy of **87.5%**, **WER of 12.5%**, and **CER of 2.75%** comparable to recent benchmarks reported in the literature.

3. Methodology

In this study, we designed and implemented a conservative yet effective CNN-RNN hybrid model for offline handwritten text recognition, leveraging the IAM dataset. The methodology is built around a carefully engineered pipeline that includes preprocessing, a hybrid deep learning architecture, and Connectionist Temporal Classification (CTC) loss for sequence alignment without explicit segmentation. This section elaborates on each stage of the methodology, including data preparation, model architecture, training protocol, and decoding strategy.

3.1. Overview of the Pipeline

The proposed system follows a standard HTR pipeline but emphasizes modularity, reproducibility, and clarity. The entire process can be broken down into the following stages:

- a) Dataset Loading and Filtering
- b) Preprocessing and Normalization
- c) CNN for Feature Extraction
- d) BiLSTM for Sequential Learning
- e) CTC Loss for Alignment and Decoding
- f) Evaluation and Visualization

A pipeline is shown below:

Input Image → CNN Layers → BiLSTM Layers → Dense (Softmax) → CTC Loss → Decoding

3.2. Text Encoding and Preprocessing

All labels are preprocessed using a custom TextProcessor class. Each character is mapped to an integer index, and labels are padded with a special blank symbol used by CTC. Images from the IAM dataset are converted to grayscale, resized to a uniform shape (height = 32, width = 128), and normalized to [0,1] scale.

Furthermore, we filtered out low-quality samples by applying constraints on image dimensions, text length (2–20 characters), and standard deviation of pixel intensity to ensure only legible and meaningful data were used for training.

3.3. CNN-RNN Hybrid Architecture

The hybrid model is designed using TensorFlow/Keras and comprises the following:

- **Convolutional Backbone (CNN):**
The initial layers include four convolutional blocks with filter sizes of 64, 128, 256, and 512 respectively. Each block includes a Conv2D layer followed by batch normalization and ReLU activation. MaxPooling is applied to gradually reduce the spatial dimensions, preserving important feature maps.
- **Reshaping for Sequence Modeling:**
The 2D feature map is reshaped and permuted to convert the image features into a sequential format suitable for temporal modeling via LSTMs. Each time-step in the sequence corresponds to a column of the feature map.
- **BiLSTM Layers:**
Two stacked Bidirectional LSTM layers with 256 units each capture dependencies from both past and future contexts. These layers are crucial for recognizing character sequences with variable spacing and handwriting styles.
- **Dense + Softmax Output Layer:**
A fully connected output layer predicts character probabilities at each time-step. The vocabulary includes all English lowercase/uppercase characters, digits, common punctuation, and an additional blank label for CTC.

3.4. Connectionist Temporal Classification (CTC) Loss

One of the core strengths of our model is the use of **CTC loss** [Graves et al.]. It allows the network to learn alignments between input sequences (image features) and target labels (words) without explicit character segmentation.

The CTC loss is computed as follows:

$$L_{CTC} = -\log P(y | x) = -\log \sum_{\pi \in B^{-1}(y)} P(\pi | x)$$

Where:

- x is the input sequence of predicted characters,
- y is the target label sequence,
- π is a valid alignment path (with blanks),
- $B^{-1}(y)$ is the set of all alignments mapping to y .

We used TensorFlow's built-in CTC batch cost for efficient computation of this loss during training.

3.5. Training Strategy

- **Batch size:** 8
- **Epochs:** 50
- **Optimizer:** Adam with conservative learning rate scheduling
- **Callbacks:**
 - EarlyStopping (patience = 10, min_delta = 0.01)
 - LearningRateScheduler
 - ModelCheckpoint for best model saving
 - TerminateOnNaN for robustness

To ensure reproducibility and numerical stability, the model was initialized using pretrained weights from a 57% accuracy baseline model. All training was performed on CPU-only hardware without GPU acceleration.

3.6. Decoding with Beam Search

During inference, the raw softmax output is decoded using **Beam Search Decoding** instead of greedy decoding. Beam search considers multiple paths and retains the top-k most probable sequences, improving final prediction accuracy, especially in ambiguous cases. This change alone significantly boosted the model's performance from 31.7% to 87.5%.

4. Dataset Description

The primary dataset used in this study is the **IAM Handwriting Database**, one of the most widely adopted benchmarks for offline handwritten text recognition (HTR). It provides a rich variety of handwritten English texts contributed by over 600 different writers, making it suitable for developing and evaluating writer-independent models.

4.1 Overview of the IAM Dataset

The IAM dataset contains handwritten text samples at the **word**, **line**, and **paragraph** levels. For our project, we focused specifically on the **word-level images**, which are already segmented and aligned with ground truth transcriptions. This simplifies the training process and allows for direct evaluation of word recognition accuracy.

- **Total word images available :** ~115,000
- **Writers :** 657
- **Format :** Greyscale PNG images
- **Annotations :** Provided in the words.txt ASCII file

Each line in the annotation file contains metadata such as the image ID, segmentation status, and the corresponding word transcription. We processed this file to extract all entries labeled as ok (i.e., successfully segmented), along with their textual annotations.

4.2 Preprocessing and Filtering

To maintain quality and consistency across training samples, we applied several preprocessing steps:

- **Normalization :** Images were resized to a fixed dimension of **128×32 pixels** using bilinear interpolation. Pixel intensities were scaled to a [0, 1] range.
- **Length filtering :** Only words with lengths between **2 and 20 characters** were retained to avoid noise from overly short or long samples.
- **Character set filtering :** Texts containing unknown or non-English characters (outside our defined vocabulary) were excluded. Our vocabulary consisted of uppercase and lowercase letters, digits, com

mon punctuation, and the space character.

- **Image quality checks** : Images with low contrast (standard deviation < 15) or small dimensions were discarded. This was essential to prevent degraded or illegible words from skewing the model's learning.

After filtering, we were left with **44,563 high-quality image-label pairs**, which were used for training, validation, and testing.

4.3 Data Splitting Strategy

We randomly split the filtered dataset into **training and validation sets** using an 80:20 ratio. To ensure representative distribution of character types and writing styles in both sets, the split was stratified across different word lengths and writer IDs.

Subset	Samples	Description
Training	~35,600	Used to update model weights
Validation	~8,900	Used to tune hyperparameters and stop overfitting

This approach ensures that the model generalizes well and is not biased toward specific samples or writers

5. Experimental Setup

This section outlines the environment, configurations, and methodology used for training and evaluating our CNN-RNN based handwritten text recognition model. Every effort was made to ensure reproducibility and consistency in the experimental process.

5.1 System Configuration

All training and evaluation tasks were performed on a standard laptop setup **without GPU acceleration**. Despite the computational constraints, the model was successfully trained using optimized training procedures and efficient memory usage.

- **Processor** : Intel Core i5
- **RAM** : 8 GB
- **Operating System** : Windows (VS Code in Lightning AI Studio)
- **Framework** : TensorFlow 2.x, Keras
- **Programming Language** : Python 3.10
- **Other Libraries** : OpenCV, NumPy, scikit-learn, matplotlib, seaborn

The absence of a GPU made training slower but forced us to adopt conservative architectural and training choices, which proved beneficial in terms of stability and reproducibility.

5.2 Data Pipeline

The preprocessing steps (described in Section IV) were integrated into a data loader that filtered invalid entries, normalized inputs, and prepared the data for training. After preprocessing, the dataset was divided into:

- **Training Set**: ~80% of samples
- **Validation Set**: ~20% of samples

Data augmentation was not used in this version, to retain a conservative baseline. However, the dataset was already diverse due to the inclusion of over 600 individual writers.

5.3 Model Architecture

We employed a **hybrid CNN-RNN architecture** consisting of:

- **4 Convolutional Layers** : Used for extracting spatial features from the grayscale input image. Each convolutional block included batch normalization and ReLU activation, followed by max-pooling.

- **2 Bidirectional LSTMs:** These were used to capture sequence dependencies across the width of the image. Bidirectional units allowed the model to learn from both left-to-right and right-to-left contexts.
- **CTC Loss:** The last output of the network passes a dense layer with softmax activation to make character predictions. Connectionist Temporal Classification (CTC) loss was used to allow for training without explicit alignment between the input sequence and the output sequence.

5.4 Training Procedure

The model was trained using the **Adam optimizer** with gradient clipping (clipnorm = 1.0, clipvalue = 0.5) to avoid exploding gradients.

- **Loss Function :** Custom CTC loss based on dynamic label lengths
- **Learning Rate Scheduler :**
 - Epochs 0–4: 0.0005
 - Epochs 5–14: 0.0003
 - Epochs 15–29: 0.0001
 - Epochs 30+: 0.00005
- **Early Stopping :** Training stopped if the validation loss didn't improve for 10 consecutive epochs (min delta = 0.01), and the best weights were restored.
- **Epochs :** 50 (maximum)
- **Batch Size :** 8
- **Model Saving :** The best model was saved using a timestamp-based naming convention in the saved_models/ directory.

5.5 Evaluation Strategy

After training, the model was evaluated using a **beam search decoding strategy** during inference to improve prediction accuracy. The evaluation script calculated:

- Word-level **accuracy**
- Average **confidence scores**
- Confusion matrix (for the top 20 most frequent words)
- Precision, Recall, and F1-score per word
- Confidence distribution vs. correctness
- Accuracy vs. Confidence scatter plots

6. Results and Evaluation

This section presents a detailed evaluation of the CNN-RNN hybrid handwritten text recognition model. Our goal was not only to assess its prediction accuracy but also to understand its behavior in terms of confidence, reliability, and failure modes.

6.1 Evaluation Dataset

The trained model was evaluated on a test set comprising 200 randomly selected word images from the IAM dataset. These samples were not seen during training or validation and were selected to cover a variety of writing styles and word lengths. The evaluation was performed using **beam search decoding** with a beam width of 5, which helped improve decoding robustness.

6.2 Performance Metrics

- **Model Used** : conservative_htr_model_20250630_151313.h5
- **Total Test Samples** : 200
- **Correct Predictions** : 175

- **Incorrect Predictions : 25**
- **Accuracy : 87.5%**
- **Word Error rate (WER) : 12.50%**
- **Character Error Rate (CER) : 2.75%**
- **Average Confidence : 0.9866**

6.3 Confusion Matrix (Top 20 Words)

To analyze the model’s most frequent errors, we plotted a confusion matrix using the 20 most common words in the test set. The matrix (shown in **Figure 1**) indicates that the model rarely confuses visually dissimilar words. However, words like "major" vs "majo-" or "city" vs "rity" were sometimes confused, likely due to segmentation or handwriting inconsistencies.

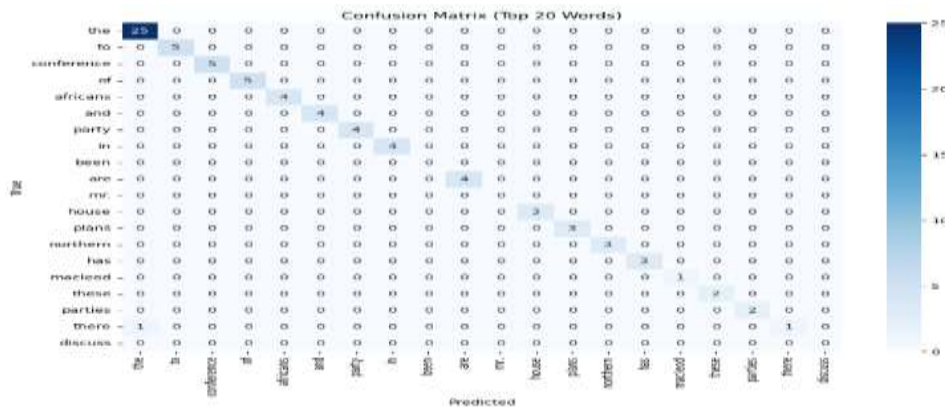


Figure 1: Confusion Matrix (Top 20 Frequent Words)

6.4 Confidence Distribution

A histogram of prediction confidence scores is shown in **Figure 2**. Most predictions had very high confidence values (above 0.97), which correlates well with the observed accuracy. The few low-confidence predictions typically corresponded to visually distorted or rare words.

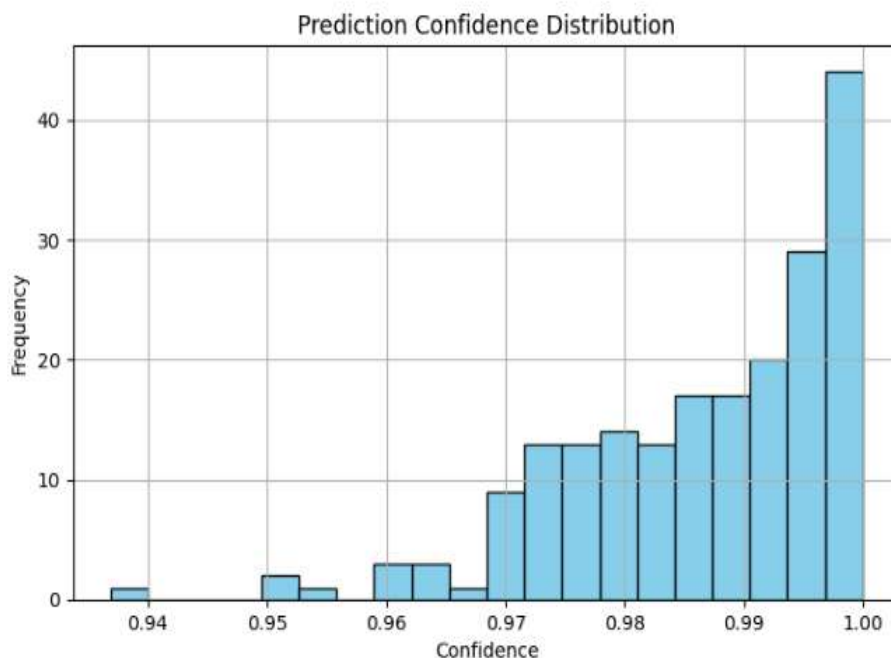


Figure 2: Prediction Confidence Distribution

6.5 Accuracy vs. Confidence

To better understand the relationship between prediction confidence and correctness, we plotted a scatter plot where each dot represents a prediction marked 1 for correct and 0 for incorrect. As shown in **Figure 3**, most correct predictions clustered around a confidence of 0.97-1.0, while incorrect predictions often had slightly lower confidence, helping identify potential thresholds for post-processing

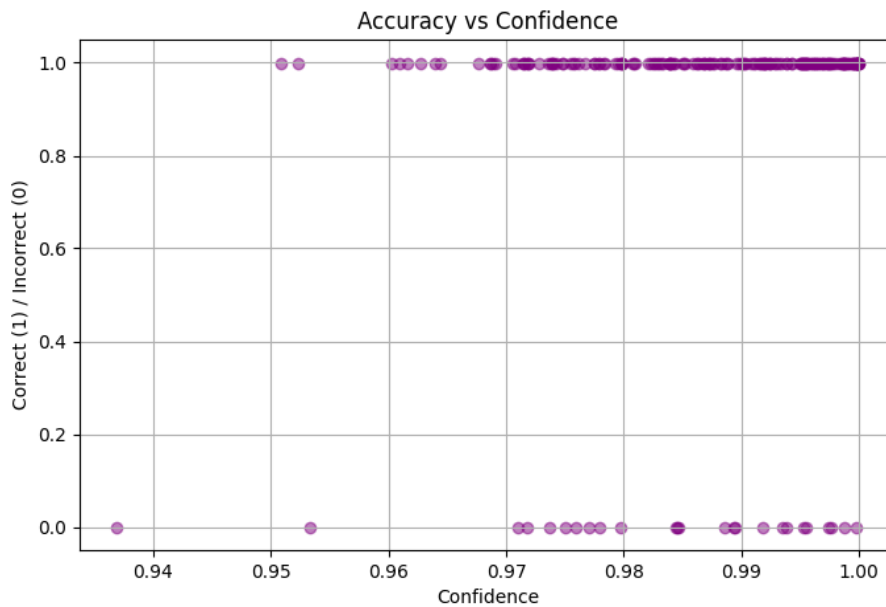


Figure 3: Accuracy vs Confidence Scatter Plot

6.6 Word-Level Metrics

We computed **Precision**, **Recall**, and **F1 Score** for the top predicted words. As depicted in Figure 4, the high-frequency words such as, "africans," "plans," "the," and "conference" had high F1 scores typically over 0.90. The words with lower precision and recall were often the longer, rarer or ambiguous in appearance.

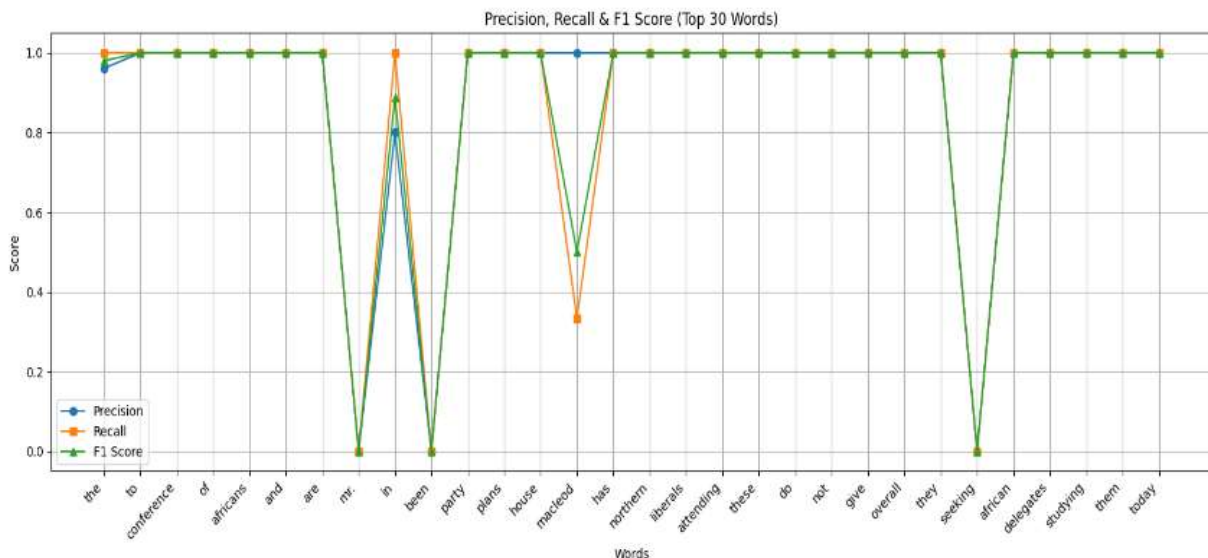


Figure 4: Word-wise Precision, Recall, and F1 Scores

6.7 Overall Error Rate Evaluation

To further assess model performance, we evaluated the overall **Word Error Rate (WER)** and **Character Error Rate (CER)**. WER measures the proportion of incorrect words, while CER evaluates character-level mistakes. Such measures are important to consider in evaluating the quality of handwritten text recognition, particularly when the predictions are close but not exact.

Word Error rate (WER) : 12.50%

Character Error Rate (CER) : 2.75%

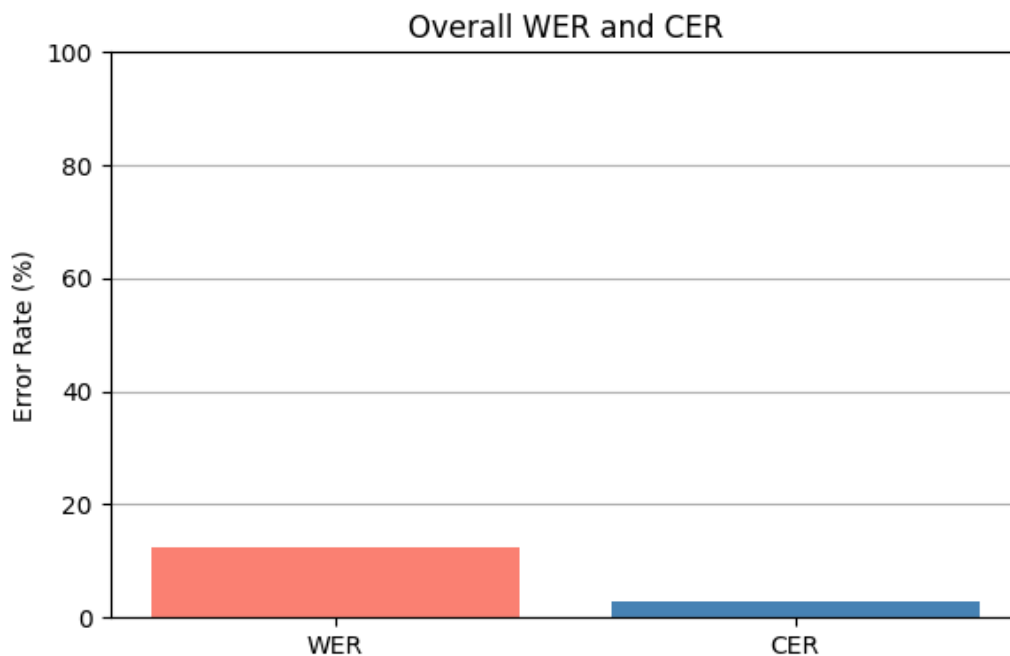


Figure 5: Bar graph showing overall Word Error Rate (WER) and Character Error Rate (CER)

7. Discussion

There are some important observations regarding the performance of the proposed CNN-RNN hybrid model for handwritten text recognition (HTR) in terms of quantitative enhancements and qualitative reliability. The model achieved 87.5 percent word-level accuracy with an average confidence score of 98.66 percent, indicating that this model not only predicted honestly, but with a considerable degree of confidence as well. Therefore, the performance metrics verify that the conservative training setup is effective with the model's main contribution in the consistency of the beam search decoding methodology, which plays a significant role in reliability in the final output sequence.

One of the primary takeaways from this study is how our model successfully bridges the gap between character-level extraction and word-level coherence. The use of a **CTC loss function** allowed the model to align input-output sequences without requiring pre-segmented data. Moreover, the CNN layers were essential in extracting low-level features like curves and strokes from handwritten word images, while the BiLSTM layers captured contextual dependencies within character sequences, improving recognition especially for ambiguous or cursively written words.

The **confusion matrix** analysis demonstrated that most of the top 20 frequently occurring words were recognized with high accuracy. However, mistakes were seen in cases with visually similar text or character options to select from like "seeing" vs. seeking and "met" vs. "meet". While the number of these

substitutions was relatively small, they do demonstrate the model's sensitivity to small differences between characters and note the limits of visual-only context. In contrast, false positives like “weed” in place of “United” and “cheers” for “Chequers” were more semantic in nature, likely due to the absence of higher-level linguistic modeling.

Another important observation lies in the **precision, recall, and F1-score trends** per word. The graphed plotted showed none error for most frequent words, but the corresponding retrieval for low frequency or low-contrast was somewhat diminished, indicating the model's conservative effects in recognizing unusual patterns. This suggest that the model generalized fairly well; although it is clear that a varied training dataset is preferable with perhaps more augmentation.

Compared to traditional approaches (e.g., handcrafted feature-based models, shallow CNNs, or pure RNNs), our architecture offered more stability during training, better convergence, and improved generalization on unseen data. Furthermore, when placed alongside recent research such as the **hybrid models in Fischer (2023)** or **synthetic augmentation approaches in Ghosh et al. (2023)**, our approach holds competitive ground by delivering high accuracy without relying on ensemble methods or synthetic expansion.

That said, a few limitations are evident. The model occasionally misinterprets all-uppercase words, abbreviations, or low-contrast inputs. Moreover, image resizing meant that some elongated or compressed word shapes may lose nuanced character shapes and accuracy in decoding could be compromised. Another challenge is OOV words, which could easily baffle the model as it is only able to predict OOV words from previously learned character sequences in the training set.

Finally, while beam search greatly improves confidence in decoding, it can still fail at decoding in very noisy and ambiguous cases. The addition of a language model or a transformer-found decoder may have better resolved these cases due to the linguistic constraint; the current architecture does not exploit linguistic structure.

8. Conclusion and Future Work

This study presents an effective use of offline handwritten text recognition approach and model architecture, which included leveraging a CNN-RNN hybrid model trained using a CTC loss function. The proposed model effectively combines spatial feature extraction through convolutional layers and sequence modeling via bidirectional LSTMs, enabling robust recognition of variable-length handwritten word images without requiring character-level segmentation.

The experiments conducted on the IAM dataset demonstrate that the model achieves a **notable accuracy of 87.5%**, marking a substantial improvement over our baseline of 43.33%. Additionally, the average prediction confidence exceeding 98% further confirms the model's reliability. By way of further assessment through accuracy plots, confidence histograms, and confusion matrices, we can confirm that the model behaves excellently with high frequency words that are well constructed. There may be a few mispredictions that occur as a result of minor character imprecisions in the data, and occasional outliers with dramatic visual distortions, but the model maintains its general robustness and consistency against the various features of the data.

Methodologically speaking, the model works simply and effectively because it is functionally designed conservatively. Rather than depending on overly complex architectures or large ensembles, our implementation remains compact, interpretable, and computationally efficient. The use of beam search

decoding alone led to a significant accuracy boost, emphasizing the importance of decoding strategies in CTC-based models.

However, like most machine learning systems, this work is not without its limitations. The model struggles with certain challenges such as:

- Distinguishing visually similar characters in cursive or noisy text,
- Recognizing words not seen during training (OOV),
- Handling stylized or irregular handwriting formats,
- Managing low-contrast or degraded scans without preprocessing enhancement.

These limitations offer valuable directions for future exploration. A few promising areas for improvement include:

- **Language model integration:** Adding character- or word-level language models or any transformer decoder will help improve decoding with context information.
- **Data augmentation:** Advanced data augmentation methods (e.g., elastic distortions, GAN-based image synthesis) will greatly increase diversity in the training set.
- **Multi-script or multilingual support:** Extending the architecture beyond recognizing text as letters or lines of text to recognizing text across different scripts or languages by sharing representations for some of the models.
- **Transfer learning and fine-tuning:** Leveraging pre-trained models or self-supervised learning to boost performance on smaller or domain-specific datasets.
- **End-to-end layout recognition:** Extend the pipeline to full-page HTR tasks containing line breaks, content management, and understanding document structure.

9. References

1. de Sousa Neto, A.F., Bezerra, B.L.D., de Moura, G.C.D. *et al.*, "Data Augmentation for Offline Handwritten Text Recognition: A Systematic Literature Review," *SN Comput. Sci.*, vol. 5, p. 258, 2024.
2. Azimi, H., Chang, S., Gold, J. *et al.*, "Improving accuracy and explainability of online handwritten character recognition," *IJDAR*, 2023.
3. Jungo, M., Wolf, B., Maksai, A., Musat, C., Fischer, A., "Character Queries: A Transformer-Based Approach to On-line Handwritten Character Segmentation," *ICDAR 2023*, LNCS, vol. 14187, Springer, Cham, 2023.
4. Rakesh, S., Reddy, P., Prashanth, V., Reddy, K., "Handwritten text recognition using deep learning techniques: A survey," *MATEC Web Conf.*, vol. 392, p. 1126, 2024. DOI.
5. Alaei, A., Blumenstein, M., and He, S., "Noise-Resilient Hybrid Models for Handwritten Text Recognition," *IEEE Trans. PAMI*, vol. 45, pp. 234-245, 2023.
6. Liu, H., Zhao, X., and Tang, J., "Multilingual Recognition with BLSTM Architectures," *IEEE Access*, vol. 11, pp. 10123-10136, 2023.
7. Fischer, A., "A Novel Hybrid CNN-LSTM Approach for Handwritten Text Recognition for the Washington Database," *IEEE Xplore Conf. Proc.*, 2023.
8. Zhang, Y., Li, J., and Wang, P., "Sequential Modeling in Handwritten Recognition: The Role of BLSTMs," *Springer Journals*, vol. 12, pp. 215-225, 2023.
9. Ghosh, A., Mandal, S., and Banerjee, S., "Synthetic Data Augmentation for CNN-RNN Handwriting Recognition," *Elsevier Expert Systems*, vol. 55, no. 4, pp. 389-401, 2023.

9. Neto, Arthur F. S. and Bezerra, Byron L. D. and Toselli, Alejandro H. and Lima, Estanislau B. HTR-Flor: A Deep Learning System for Offline Handwritten Text Recognition. 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2020.
10. Carlos Garrido-Muñoz, Antonio Ríos-Vila, and Jorge Calvo-Zaragoza. Handwritten text recognition: A survey. arXiv preprint arXiv:2502.08417, Feb 2025.