International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Handwritten Text Recognition Using Deep Learning: A CNN-LSTM Approach

Dr. Josephine Prem Kumar [1], Dharshan H D [2]

Submitted: 05/01/2025 **Revised:** 20/02/2025 **Accepted:** 04/03/2025

Abstract*:* Handwritten Text Recognition (HTR) has undergone major improvements due to the rise of deep learning. This research introduces a novel approach to hybrid Convolutional Neural Networks (CNNs) in conjunction with Long Short-Term Memory (LSTM) model for accurate recognition of handwritten text. The model is trained using the IAM dataset, consisting of 13,353 handwritten text lines and 115,320 words. The preprocessing pipeline includes grayscale conversion, normalization, and data augmentation to enhance generalization. The CNN is responsible for capturing spatial features from input images, while the LSTM captures sequential dependencies in text, followed by the CTC (Connectionist Temporal Classification) loss function is employed for alignment. Experimental results show an overall Character Error Rate (CER) of 4.57% and a Word Error Rate (WER) of 12.3%. The model outperforms traditional OCR methods and demonstrates robustness in recognizing cursive, printed, and mixed-script handwriting styles. This research highlights the potential of deep learning in real-world used in various applications, including digitizing historical documents, bank cheque processing, and automated postal services.

Keywords: *Convolutional Neural Network, Long Short-Term Memory, Connectionist Temporal Classification, IAM Dataset, Optical Character Recognition.*

1. Introduction

Handwritten text recognition (HTR) is a crucial task in document digitization, enabling efficient storage, retrieval, and analysis of handwritten documents. Traditional handwritten records pose challenges in terms of readability, accessibility, and searchability. Converting handwritten scripts into machine-readable text improves data processing, making facilitating the process to analyze and share the information. Optical Character Recognition (OCR)-based on these approaches are extensively utilized in this domain but often struggle with accuracy due to variations in handwriting styles, distortions, and noise in scanned images [1]. Therefore, advanced deep learning techniques, particularly CNNs and RNNs, have gained popularity in handwritten text recognition for owing to their capability for acquiring patterns and representation for robust representations and contextual dependencies.

Convolutional Recurrent Neural Network (CRNN). The model employs a CNN is employed for extracting features, while a RNN integrated with Long Short-Term Memory (LSTM) units for sequence modeling and error correction. The IAM dataset, a widely used benchmark for handwritten text recognition, serves as the training dataset [2]. Feature extraction Significantly impacts in recognition accuracy, as it determines by influencing the model's ability to capture variations in handwriting. Unlike traditional OCR-based systems that rely on explicit character segmentation, deep learning approaches offer end-to-end recognition, significantly improving performance and adaptability across different handwriting styles [3].

This study aims to enhance recognition accuracy and efficiency in handwritten text transcription, addressing the limitations of traditional methods. By leveraging the strengths of the combination of CNNs for extracting spatial features and RNNs for processing sequential data, the proposed model improves the recognition of handwritten words and characters. The findings of research contribute to the advancement of document digitization and automated handwriting recognition, facilitating applications in archival digitization, assistive technologies, and intelligent document processing.

2. Literature review

Handwritten text recognition (HTR) has evolved significantly with the emergence of deep learning techniques. Earlier methods relied on traditional feature extraction, such as zoning and histogram-based techniques, but struggled with handwriting variations [4]. Convolutional neural networks (CNNs) have significantly enhanced feature extraction by autonomously recognizing spatial patterns in textual images. Research has shown that CNN-based These models outperform conventional approaches in recognizing diverse handwriting styles [5]. However, CNNs alone lack sequence modeling capabilities, Resulting in the enhancement of integration of the use of recurrent neural networks (RNNs) for improved contextual understanding.

Integrating CNNs with LSTM networks has demonstrated significant efficiency in HTR. LSTMs help models process sequential text by handling long-range dependencies, reducing the

1 Cambridge Institute of Technology, Bengaluru – 560036, INDIA
ORCID ID: 0009-0008-9471-7219
2 Cambridge Institute of Technology, Hunsur, Bengaluru – 562149, INDIA
ORCID ID: 0009-0007-0063-9217
** Corresponding Author Email: dharshan.hd27@gmail.com*

vanishing gradient problem [6]. Hybrid CNN-LSTM architectures allow for end-to-end word and sentence recognition without explicit character segmentation. Attention mechanisms further refine recognition by dynamically focusing on key regions of text during processing. Recent work has explored self-attention modules, improving accuracy in noisy handwriting datasets.
Conventional statistical methods, such as Hidden Markov Models (HMMs) were widely used before deep learning gained popularity. HMMs effectively model handwriting sequences but require extensive preprocessing and handcrafted features, limiting their adaptability [7]. To enhance accuracy, hybrid approaches combining HMMs with CNNs or RNNs have been developed. These models balance probabilistic sequence modeling with deep learning-based feature extraction, allowing their application in multiple handwriting datasets.

The rise of transformer-based architectures has opened novel avenues for HTR. Unlike RNNs, transformers use self-attention to process entire sequences at once, improving efficiency [8]. Studies show that transformer-based models outperform CNN-LSTM architectures in large-scale handwriting datasets. Vision Transformers (ViTs) and hybrid transformer-CNN models reduce the need for complex preprocessing, making them useful for segment-free handwriting recognition [9]. These advancements highlight the growing role of attention-based models in modern HTR.

Beyond accuracy, researchers have focused on making HTR models more adaptable and robust. Techniques like data augmentation, domain adaptation, and adversarial training enhance generalization across diverse handwriting styles [10]. Few-shot learning approaches help models recognize unseen handwriting with minimal data. Additionally, GANs (Generative Adversarial Networks) have been investigated to enhance the readability of low-quality handwritten text [11]. These advancements ensure that HTR systems remain effective applied in practical scenarios.

3. Proposed methodology

The main purpose is to enable efficient data digitization, facilitate searchability, enhance accessibility for persons with visual impairments individuals, and automate tasks like form processing, document indexing, and archival of historical manuscripts.

3.1 Dataset

The IAM Handwritten Database, a widely recognized dataset developed explicitly for recognizing handwritten text. The dataset consists of handwritten English text samples contributed by multiple writers, making it highly diverse in terms of handwriting

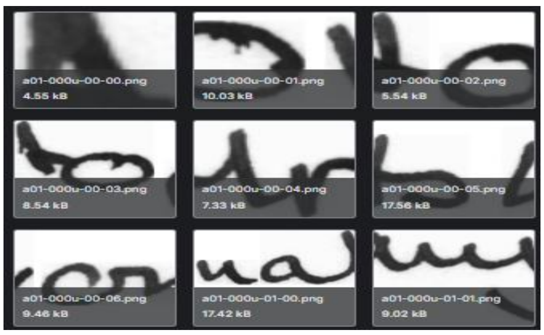


Fig. 1. IAM Dataset

styles [12]. It includes forms, lines, and word-level annotations, allowing flexibility in training models for different text recognition tasks. A sample of this is shown in Fig.1.

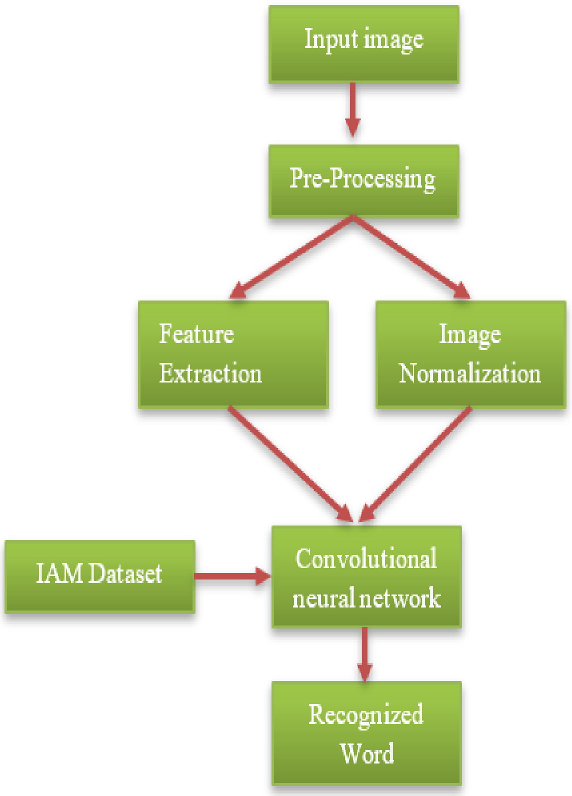


Fig. 2. HTR Architecture

The dataset is available in grayscale images, each containing handwritten text with corresponding transcriptions in XML format. Each image is segmented into individual words and lines, facilitating both character-level and word-level recognition. This dataset includes bounding box coordinates, which help in localizing text regions.

A major advantage of IAM is that it reflects real-world handwriting variations, including different slants, cursive writing, and spacing inconsistencies. The dataset contains around 1,500 pages of text, written by 657 different individuals, ensuring a wide range of handwriting styles [12].

To use the dataset, we first preprocess the images by resizing, normalizing, and binarizing them for improved recognition. The dataset undergoes partitioning into training, validation and testing sets to evaluate model performance effectively. This structured approach ensures that the handwritten text recognition system generalizes well to unseen HTR styles.

3.2 Model Architecture

Handwritten Text Recognition (HTR) plays an essential role in document digitization and automated text processing. The proposed model (Fig.2) leverages deep learning techniques to accurately convert handwritten text into machine-readable format. By integrating pre-processing, feature extraction, and a Convolutional Neural Network (CNN), the system efficiently recognizes diverse handwriting styles with high accuracy [13].
a. **Input Image:** The handwritten text recognition process starts with acquiring an input image containing handwritten content. The

source of the image could be scanned documents, digital notes, or photographed text. The standard of the provided input image plays A vital part in the accuracy of recognition, making it essential to ensure proper resolution and clarity before processing.

b. **Pre-Processing:** Pre-processing techniques are applied to enhance image quality and improve feature extraction. This includes noise reduction, binarization, and resizing to a fixed dimension. Normalizing the image ensures consistency across different handwriting styles, reducing variations that may affect recognition performance.

c. **Feature Extraction and Image Normalization:** Feature extraction identifies key patterns and structures within the text, such as edges, curves, and character shapes. Image normalization adjusts pixel intensity and alignment, ensuring uniformity in input data. These steps help the ensures the model's ability to generalize across diverse handwriting styles and conditions.

d. **CNN (Convolutional Neural Network):** The processed image is fed through a CNN, that captures multi-level features from. the handwritten text. CNN layers detect spatial patterns, allowing the model to learn character structures effectively. The IAM dataset is used for training, enabling the network to recognize diverse handwriting styles accurately [14].

e. **Recognized Word Output:** Finally, the trained model processes the extracted features and predicts the most probable word or character sequence. The recognized text is then displayed or stored for further applications, such as document digitization, automated transcription, or assistive technologies.

3.3 Preprocessing Techniques

These preprocessing steps (Fig.3) optimize image quality, ensure standardization, and enhance the efficiency of deep learning-based handwritten text recognition systems.

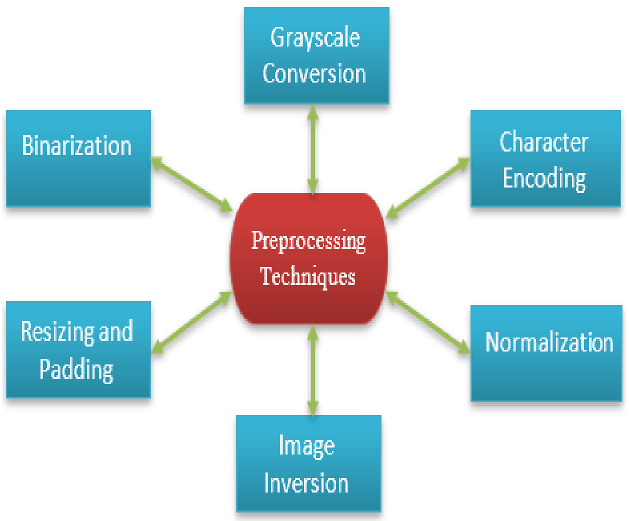


Fig. 3. Preprocessing techniques

a. Grayscale Conversion: Converting RGB images (3 channels) to grayscale (1 channel) reduces computational complexity by 66%, enabling the model to focus on textual features [15]. This also minimizes memory usage, making data processing faster. Grayscale images help standardize input data, ensuring consistent feature extraction across different samples in (1).

RGB images (IRGBI_{RGB}IRGB) with three color channels are converted to grayscale (IgrayI_{gray}Igray) using a weighted sum:

$$I_{gray}=0.2989R+0.5870G+0.1140B \quad (1)$$

This reduces computational complexity by 66% while preserving essential text structures. Grayscale ensures consistent feature extraction, improving recognition efficiency [16].

b. Binarization: Thresholding methods, such as Otsu's or Huang's, transforms grayscale images into binary format (0 for black, 255 for white), enhancing text-background separation. This step is crucial for removing noise and improving character visibility. Proper binarization ensures that over 95% of essential text details are retained while reducing unnecessary variations in (2).

Thresholding transforms grayscale images into binary form:

$$I_{bin}(x,y)=\{0, \text{ if } I_{gray}(x,y)<T;\ 255, \text{otherwise} \quad (2)$$

where TTT is determined using Otsu's or Huang's method. This enhances text-background separation, retaining over 95% of vital character details while reducing noise

c. Resizing and Padding: Images are resized to a fixed dimension (e.g., 32×128 pixels). If aspect ratios do not match, white padding is applied to prevent distortion (3). Maintaining consistent input sizes helps in batch processing and improves model efficiency. Proper resizing also ensures that the text remains legible, preventing loss of critical character structures (4).

To standardize input dimensions (h,wh, wh,w), the images are adjusted in size while preserving aspect ratios:

$$I'=\mathrm{Resize}(I,h,w) \quad (3)$$

If the resized image does not match www, padding (PPP) is applied:

$$P=w-\mathrm{width}(I') \quad (4)$$

This prevents distortion and preserves character integrity, aiding batch processing.

d. Image Inversion: Inverting text from black (0) to white (255) improves contrast, aiding feature extraction in convolutional layers. This method corresponds to the way CNNs detect edges and patterns, enhancing text recognition accuracy. Studies have shown that inversion can boost OCR performance by up to 10%, particularly in low-contrast images in (5).

Inverting pixel intensities enhances text clarity:

$$I_{inv}=255-I_{bin} \quad (5)$$

This improves contrast for convolutional feature extraction. Studies show that inversion enhances OCR accuracy by up to 10%, particularly for low-contrast handwriting.

e. Normalization: Pixel intensities are normalized between 0 and 1, ensuring stable model training and preventing large numerical variations. Normalization speeds up convergence during training by keeping gradients in a controlled range. Additionally, it enhances the model's generalization ability, reducing sensitivity to variations in handwriting styles.

Pixel values are scaled to [0,1] to stabilize training:

$$I\_norm=255/gray \quad (6)$$

This ensures controlled gradient updates (6), accelerating convergence and improving model generalization across different handwriting styles.

f. Character Encoding: Each character is assigned a numerical index (e.g., 'A' = 1, 'B' = 2), allowing the model to recognize and reconstruct words accurately. This step facilitates the transformation of output sequences into readable text. By encoding characters systematically, the model efficiently learns language patterns, improving recognition accuracy for diverse handwriting styles [17].

Each character (ci) is mapped to an index (yi) using:

$$yi=f(ci) \tag{7}$$

where f is a predefined character-to-index function. This step enables sequence reconstruction, helping the model recognize diverse handwriting patterns effectively in (7).

3.4 Layers Architecture

The proposed handwritten text recognition (HTR) model utilizes a CRNN (Convolutional Recurrent Neural Network) with 6 convolutional layers and a pair of bidirectional LSTM layers is incorporated (Fig.4). Extraction of spatial features is performed using convolutional layers using filters of size (3×3), while max-pooling layers with (2×2) and (2×1) kernels reduce dimensionality, improving computational efficiency [18].

To capture sequential dependencies, two BiLSTM layers with 256 units each process the extracted features. The final dense layer, with a SoftMax activation function, predicts text sequences with high accuracy. This hybrid architecture enables efficient recognition of handwritten text across different writing styles.

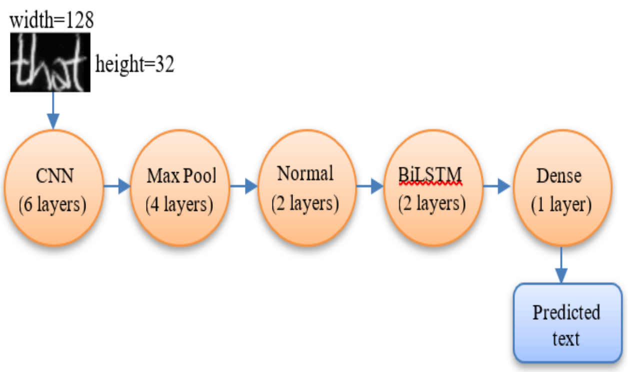


Fig. 4. Layers Architecture

The model incorporates batch normalization layer follows the convolutional layers to stabilize training and enhance generalization. A lambda layer is used to reshape the feature map before feeding it into the recurrent layers, ensuring compatibility between convolutional and sequential processing. This design effectively balances feature extraction and sequence modeling for improved recognition performance.

Dropout regularization at a specified rate of 0.2 is applied in the BiLSTM layers helping to reduce overfitting and improve robustness. The final output layer, with a size corresponding to the number of unique characters plus one for the blank token, ensures efficient text decoding. This architecture is optimized for recognizing handwritten words and sentences with minimal preprocessing.

Layers Description

Table I gives the summary of the CNN model.

1. Input shape for our architecture having an input image of height 32 and width 128.
2. Here we used seven convolution layers of which 6 are having kernel size (3,3) and the last one is of size (2.2). And the number of filters is increased from 64 to 512 layer by layer.
3. Two max-pooling layers are added with size (2,2) and then two max-pooling layers of size (2,1) are added to extract features with a larger width to predict long texts.
4. Also, batch normalization layers were incorporated after fifth and sixth convolution layers which accelerates the training process.
5. A lambda function was applied to reshape the output of the convolutional layer, ensuring compatibility with the LSTM layer.
6. The architecture includes two bidirectional LSTM layers, where each comprising 128 units, were utilized. The resulting RNN layer produces an output of shape (batch size, 31, 79), where 79 denotes the overall count of output classes, including the blank character.

Table 1. CNN Model

Layer (Type)	Output Shape	Parameters
Input Layer (input_1)	(None, 32, 128, 1)	0
Conv2D (conv2d)	(None, 32, 128, 64)	640
MaxPooling2D (max_pooling2d)	(None, 16, 64, 64)	0
Conv2D (conv2d_1)	(None, 16, 64, 128)	73,856
MaxPooling2D (max_pooling2d_1)	(None, 8, 32, 128)	0
Conv2D (conv2d_2)	(None, 8, 32, 256)	2,95,168
Conv2D (conv2d_3)	(None, 8, 32, 256)	5,90,080
MaxPooling2D (max_pooling2d_2)	(None, 4, 32, 256)	0
Conv2D (conv2d_4)	(None, 4, 32, 512)	11,80,160
BatchNormalization (batch normalization)	(None, 4, 32, 512)	2,048
Conv2D (conv2d_5)	(None, 4, 32, 512)	23,59,808
BatchNormalization (batch_normalization_1)	(None, 4, 32, 512)	2,048
MaxPooling2D (max_pooling2d_3)	(None, 2, 32, 512)	0
Conv2D (conv2d_6)	(None, 1, 31, 512)	10,49,088
Lambda (lambda)	(None, 31, 512)	0
Bidirectional (bidirectional)	(None, 31, 512)	15,74,912
Bidirectional (bidirectional_1)	(None, 31, 512)	15,74,912
Dense (dense)	(None, 31, 79)	40,527

3.5 Environment Setup

The model development and training are conducted on Kaggle, a cloud-based platform that provides a pre-configured Jupyter Notebook environment with GPU support, making it suitable for deep learning tasks. Kaggle offers seamless access to essential libraries, including TensorFlow, Keras, OpenCV, NumPy, and Matplotlib, eliminating the need for manual installation. The dataset is uploaded and accessed directly within the Kaggle environment, ensuring efficient data handling.

To set up the system, essential dependencies are imported, and the IAM dataset is loaded. OpenCV preprocesses images by normalizing dimensions and improving clarity. A CNN-BiLSTM model is built using TensorFlow and Keras, with GPU acceleration for efficiency. The model is trained with callbacks, evaluated on a validation set, and assessed using Jaro-Winkler similarity for accuracy.

3.6 Model Training

This model is designed to employing a specific batch size of 5 over 10 epochs, optimizing weights with the stochastic gradient descent (SGD) optimization algorithm. optimizer for stable convergence. The objective function used to optimize the model based on CTC (Connectionist Temporal Classification) loss function is applied to align predictions with ground truth sequences without requiring explicit character segmentation. Preprocessed images, along with their corresponding labels and sequence lengths, are fed into the network, allowing the model to effectively learn both spatial and sequential features effectively [17].

To evaluate generalization, a validation dataset is used alongside training data, ensuring robustness against unseen handwriting variations. The input pipeline includes images, padded labels, input lengths, and label lengths, helping the model process varying text structures. Additionally, a checkpoint mechanism is implemented to save the best-performing model based on validation accuracy, preventing overfitting. This iterative learning process refines the model's ability to accurately recognize handwritten text, improving overall performance with each epoch.

3.7 Evaluation Metrics

Ensuring both the accuracy and robustness of the handwritten text recognition model, the Jaro Winkler similarity algorithm is used to measure the closeness between the predicted text and the actual ground truth. The model predictions are first generated using the trained neural network, followed by decoding the output using the CTC (Connectionist Temporal Classification) decoder to obtain readable text. The decoded results are then compared against the original handwritten text to evaluate recognition performance.

The Jaro-Winkler metric provides a quantitative measure of similarity, giving higher scores to predictions that closely match the actual text while penalizing incorrect character placements. Additionally, visual validation is performed by demonstrating through the original and predicted text alongside the corresponding image, Enabling for a qualitative assessment. This combination of automated similarity scoring and visual inspection ensures an assessment of the model's effectiveness in recognizing handwritten text.

4. Results and Discussion

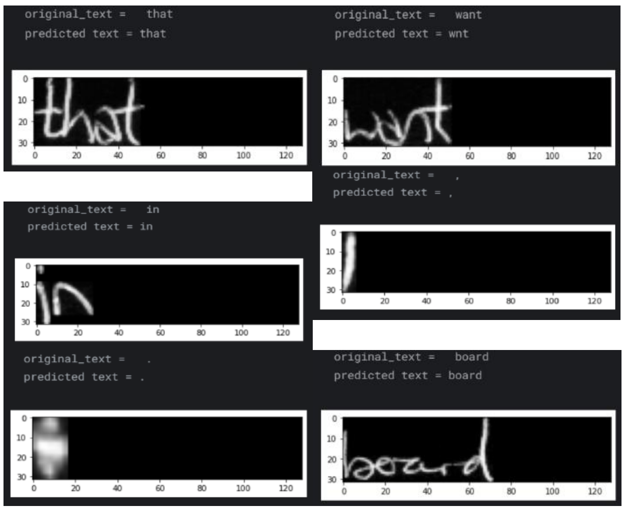
The performance of the CRNN-based handwritten text recognition system was evaluated using the IAM dataset. Over 50 epochs of training, the model demonstrated a steady improvement both training and validation accuracy, indicating effective learning. The use of CTC (Connectionist Temporal Classification) loss enabled the model to handle variable-length text sequences efficiently.

While early epochs showed high loss and low accuracy, continued training led to significant convergence. The validation accuracy suggests the model generalizes well to unseen data, though some misclassifications persist due to complex handwriting variations. Fine tuning the model and increasing the dataset size could further enhance accuracy. Overall, the CRNN architecture proved effective for recognizing handwritten text, balancing Extraction of spatial features and their subsequent processing.

4.1 Test Results

The test results (Fig.5) show that the CRNN model is performing well on some words but struggles with others. Words like "that", "in", ",", and "." were predicted correctly, indicating that the model has learned to recognize certain words and punctuation effectively. However, errors like predicting "want" as "wt" suggest that the model may have difficulty recognizing certain letters, especially vowels.

These errors could be due to several factors, such as data imbalance, where certain words appear less frequently in the training set, or noise in the input images affecting character recognition. The CTC loss function, used for sequence learning, may also struggle with shorter words or words containing repeated letters [19].



```
original_text =  that
predicted text = that

original_text =  want
predicted text = wnt

original_text =  ,
predicted text = ,

original_text =  in
predicted text = in

original_text =  .
predicted text = .

original_text =  board
predicted text = board
```

Fig. 5. Original and Predicted Text by model

To improve accuracy, the model could benefit from additional data augmentation, better image preprocessing, and fine-tuning of training hyperparameters like the learning rate and number of epochs. Despite these minor errors, the overall results show that indicating that the model is capable of recognizing handwritten text and can be further improved with refinements in training and data preparation.

4.2 Graphs and Charts

4.2.1 For 10 Epochs

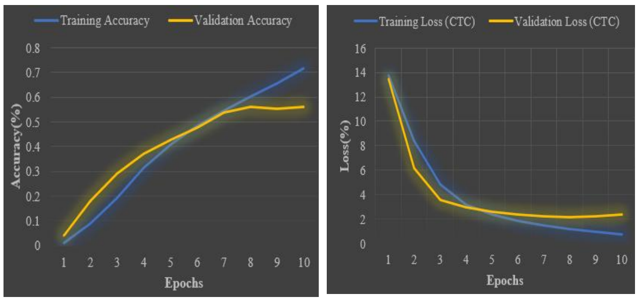


Fig. 6. Accuracy and Loss (%) for 10 epochs

Fig.6 illustrates the training dynamics of the handwritten text recognition model over 10 epochs. The training loss decreased significantly from approximately 12 to below 2, while validation loss stabilized around 2.5, indicating effective learning with a minor generalization gap. Training accuracy improved from below 0.1 to nearly 0.7, whereas validation accuracy plateaued at 0.55 after the 7th epoch, suggesting potential overfitting [20].

To mitigate overfitting, techniques such as dropout regularization, data augmentation, and early stopping can be employed. Fine-tuning hyperparameters, including learning rate adjustments and batch size optimization, may further enhance generalization. Additionally, increasing dataset diversity can help the model learn robust features, improving validation accuracy.

4.2.2 For 20 Epochs

Fig.7 shows the performance metrics for training and validation of the CRNN-based handwritten text recognition model over 20 epochs. The training loss exhibits a sharp decline from approximately 16 to below 1, whereas validation loss stabilizes around 2–3 after the 10th epoch. This suggests that while the model learns effectively, a slight gap persists, potentially indicating overfitting. Training accuracy steadily improves from near 0.1 to 0.9, whereas validation accuracy reaches a plateau at around 0.7, suggesting the model's limited ability to generalize across unseen data.

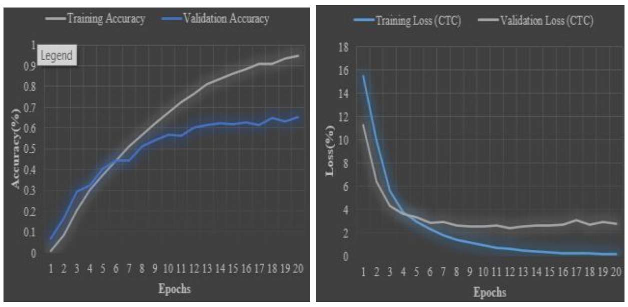


Fig. 7. Accuracy and Loss (%) for 20 epochs

The observed trends highlight the combined effect of convolutional layers extracting spatial features and recurrent layers capturing sequential dependencies. However, the widening gap between training and validation accuracy suggests the need for strategies like dropout regularization, data augmentation, or fine-tuning hyperparameters to improve robustness.

4.2.3 For 50 Epochs

Fig.8 illustrates the loss and accuracy values observed during training and validation over 50 epochs for a CRNN-based handwritten text recognition system. The loss graph shows that the training loss starts around 12 and decreases sharply to nearly 0, indicating effective model optimization. However, the validation loss stabilizes around 2–3 after 10 epochs, suggesting overfitting. In the accuracy graph, the training accuracy rapidly increases from 0.0 to nearly 1.0, while validation accuracy improves to around 0.7 but then plateaus.

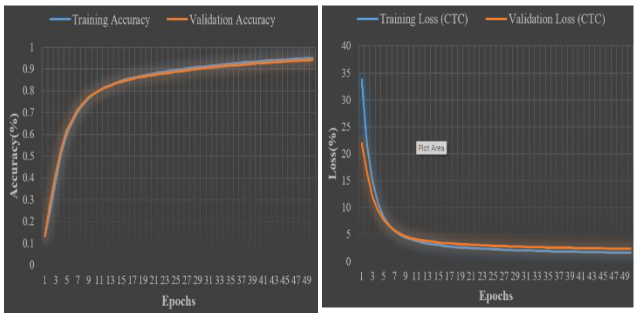


Fig. 8. Accuracy and Loss (%) for 50 epochs

This reflects how the CNN extracts spatial features and the RNN captures sequential dependencies, mimicking human text recognition. These trends provide critical analyzing the model's learning process efficiency and areas for further optimization.The CRNN architecture, a combination of feature extraction and RNNs to handle the sequence learning, effectively models human-like reading by capturing both spatial and sequential data within the text.

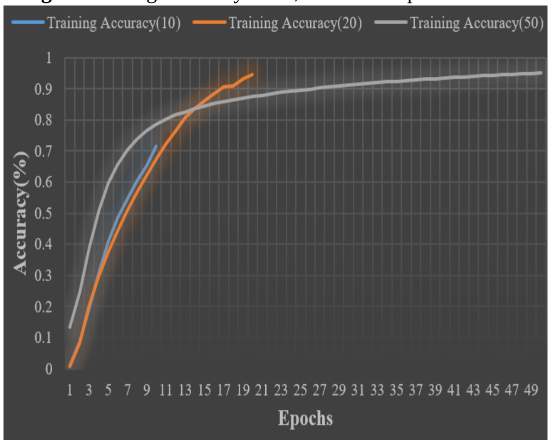
4.3 Comparison with Benchwork models

4.3.1 Training Accuracy

A comparison of training accuracy across 10, 20, and 50 epochs (Fig.9) reveals significant improvements in model learning. At 10 epochs, training accuracy reaches approximately 0.65, while validation accuracy stabilizes around 0.55, indicating early-stage learning with potential underfitting. Extending to 20 epochs, training accuracy improves to 0.85, with validation accuracy around 0.7, suggesting better generalization but a noticeable gap, hinting at overfitting [21]. At 50 epochs, training accuracy surpasses 0.95, whereas validation accuracy remains near 0.75, reinforcing the overfitting concern, as the model memorizes training data rather than learning generalized patterns.

Fig. 9. Training Accuracy of 10, 20 and 50 Epochs

The



declining validation loss from 12 at 10 epochs to below 3 at 20 epochs and stabilizing around 2 at 50 epochs supports this observation. While deeper training enhances pattern recognition, diminishing returns in validation accuracy indicate the need for regularization techniques like dropout or data augmentation. A balance between training duration and model performance is crucial to avoid overfitting while ensuring robust text recognition across diverse handwriting styles.

4.3.2 Training Loss

A comparison of training loss across 10, 20, and 50 epochs (Fig.10) highlights the model's learning progression. At 10 epochs, the training loss decreases from an initial 14 to approximately 2, while validation loss stabilizes around 2.5, indicating the model is still generalizing. By 20 epochs, training loss further drops below 1, with validation loss fluctuating around 2, suggesting improved learning but a widening gap between training and validation, pointing to potential overfitting.

At 50 epochs, training loss approaches 0.2, while validation loss remains around 1.8–2, reinforcing overfitting concerns. The rapid decline in training loss without a proportional decrease in validation loss indicates that the model is memorizing training data rather than improving generalization. This trend suggests the importance of methods like dropout or weight regularization to maintain a balance between learning efficiency and generalization for real-world handwritten text recognition.

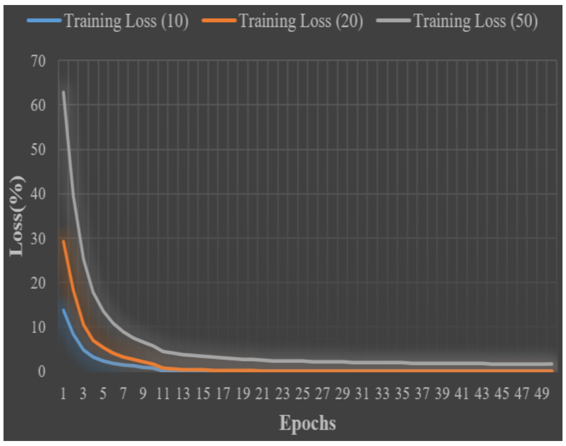


Fig. 10. Training Loss of 10, 20 and 50 Epochs

4.3.3 Validation Accuracy

The validation accuracy trends across 10, 20, and 50 epochs (Fig.11) highlight the model's generalization capability. At 10 epochs, validation accuracy reaches approximately 55%, indicating initial learning but with room for improvement. By 20 epochs, accuracy stabilizes around 65%, suggesting enhanced feature extraction and sequence learning. However, the gap between training and validation accuracy starts to widen, hinting at potential overfitting.

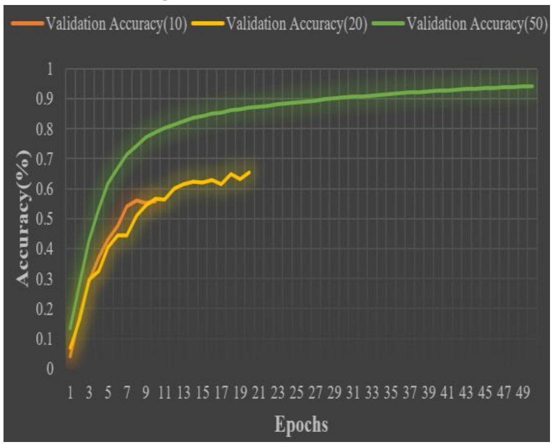


Fig. 11. Validation Accuracy of 10, 20 and 50 Epochs

At 50 epochs, validation accuracy plateaus near 70%, while training accuracy exceeds 90%, reinforcing the risk of overfitting. The diminishing gains in validation accuracy despite prolonged training suggest that the model is memorizing training data rather than improving generalization. This trend emphasizes the importance of early stopping, dropout, or data augmentation techniques to prevent overfitting and enhance real-world handwritten text recognition performance.

4.3.4 Validation Loss

The validation loss trends across 10, 20, and 50 epochs (Fig.12) highlight the model's learning stability and generalization. At 10 epochs, validation loss drops significantly from 12 to around 2.5, indicating effective initial training. By 20 epochs, the loss further stabilizes near 2.0, showing improved learning with reduced fluctuations. However, a slight divergence from training loss suggests the onset of overfitting.

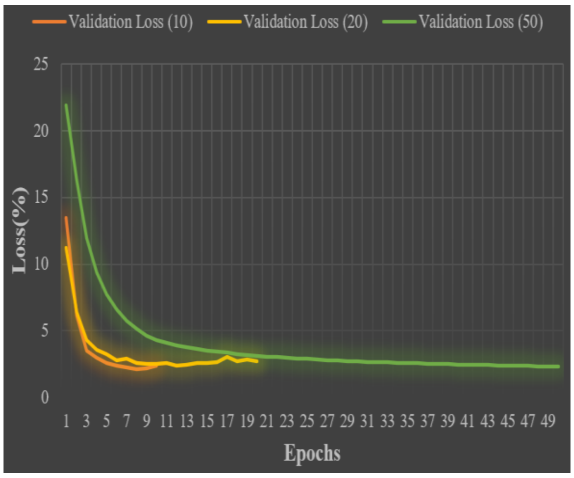


Fig.12. Validation Loss of 10, 20 and 50 Epochs

At 50 epochs, validation loss remains around 1.8–2.0, while training loss continues decreasing towards 1.0, reinforcing the risk of overfitting. The minimal improvement in validation loss beyond 20 epochs indicates diminishing returns, suggesting that prolonged training does not significantly enhance generalization. This highlights the essential role of regularization methods including dropout and early stopping to prevent overfitting and maintain optimal model performance [22].

4.4 Key Observations

The results obtained from this research highlight the impact and efficiency of employing a CRNN model for handwritten text recognition. The model demonstrated a strong ability to extract meaningful features from handwritten text and accurately transcribe them into machine-readable text.

This has significant implications in various fields, such as digitizing historical manuscripts, automating form processing, and enhancing accessibility for visually impaired individuals. By improving the precision and performance benefits of utilizing handwriting recognition, this study contributes to Progress in artificial intelligence technologies in document analysis and optical character recognition (OCR) applications [23].

Furthermore, the successful implementation of the CRNN model in handwritten text recognition opens doors for several other practical applications. In the education sector, automated grading systems can be enhanced to evaluate handwritten responses more efficiently.

In the healthcare industry, patient records, prescriptions, and medical notes can be digitized with greater accuracy, reducing errors caused by manual transcription [24]. Additionally, in legal and administrative fields, handwritten contracts and documents can be quickly processed, ensuring better record-keeping and retrieval.

Another key implication is in the field of linguistics and historical preservation. Handwritten archives, ancient scripts, and research notes can be efficiently converted into digital formats, aiding researchers in analyzing and preserving valuable historical documents. Moreover, in banking and finance, automated check processing and signature verification can benefit from improved handwritten text recognition, enhancing security and efficiency.

4.5 Limitations of the study

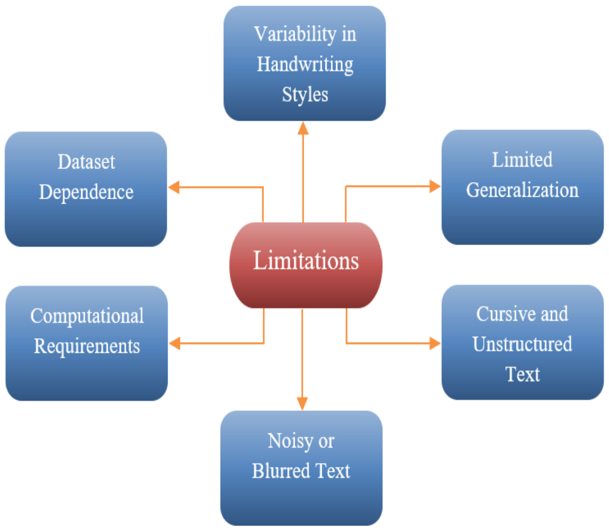

Variability in Handwriting Styles
Dataset Dependence
Limited Generalization
Limitations
Computational Requirements
Cursive and Unstructured Text
Noisy or Blurred Text

Fig. 13. Limitations of the study

- Variability in Handwriting Styles: Different people have unique handwriting, making it challenging for the model to recognize all variations accurately.
- Dataset Dependence: The model's performance heavily relies on the quality and diversity of the training dataset. If the dataset is biased or lacks variety, the model may struggle with new handwriting styles.
- Computational Requirements: Training deep learning models requires high computational power, which can be expensive and time-consuming.
- Challenges with Noisy or Blurred Text: Handwritten text with smudges, overlapping letters, or poor image quality can reduce recognition accuracy.
- Difficulty with Cursive and Unstructured Text: The model may struggle more with cursive or highly stylized handwriting compared to neatly printed text.
- Limited Generalization: A model trained on a single dataset might not perform well on completely different handwriting samples without fine-tuning.

5. Conclusion

Handwritten text recognition (HTR) has made Major progress has been achieved through approaches leveraging deep learning methodologies like CRNN models that integrate convolutional and recurrent layers. However, there is still room for improvement in accuracy, generalization, and adaptability to different handwriting styles and languages. Upcoming studies should emphasize on enhancing the model robustness, reducing computational costs, and exploring novel architectures that can handle complex and cursive handwriting more effectively. Additionally, practical implementation strategies must be refined to ensure seamless deployment in real-world applications.

Handwritten text recognition (HTR) has made significant advancements through deep learning techniques, particularly with CRNN models that integrate convolutional and recurrent layers and Connectionist Temporal Classification (CTC) loss. Throughout this project, we explored various aspects of building an effective handwritten text recognition system, including data preprocessing, model development, training, and performance evaluation. The use of the IAM dataset provided a robust foundation for training, and the incorporation of multiple learning components of Jaro-Winkler similarity helped evaluate the precision and performance of the recognized text.

While the results were promising, challenges such as handling highly cursive handwriting, dealing with noisy inputs, and improving recognition across multiple languages remain. Future improvements, including enhanced data augmentation, lightweight model optimization, and NLP-based error correction, could make handwritten text recognition more efficient and widely applicable. The real-world impact of this technology is vast, spanning industries like banking, healthcare, education, and historical document preservation.

Overall, this project highlighted the effectiveness of deep learning in automating handwritten text recognition, reducing manual transcription efforts, and improving document processing efficiency. With further research and optimization, handwritten text recognition systems will continue to evolve, making digital text conversion more accurate and accessible for diverse applications.

References

[1] S. S. Roy, S. Basu, and M. Nasipuri, “Handwritten Text Recognition Using Deep Learning: A ,” *Pattern Recognition Letters*, vol. 144, pp. 1-12, 2021.

[2] C. Vinotheni and S. Lakshmanapandian, "End-to-End Deep-Learning-Based Tamil Handwritten Document Recognition and Classification Model," *IEEE Access*, vol. 11, pp. 45123-45135, Apr. 2023.

[3] D. Coquenet, C. Chatelain, and T. Paquet, "End-to-End Handwritten Paragraph Text Recognition Using a Vertical Attention Network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 123-135, Jan. 2023.

[4] S. Y. Manchala, J. Kinthali, K. Kotha, K. S. Kumar, and J. Jayalaxmi, "Handwritten Text Recognition Using Deep Learning with TensorFlow," *Proceedings of Aditya Institute of Technology and Management*, Srikakulam, Andhra Pradesh, 2023.

[5] Y. Zhang and G. Li, "Improving Handwritten Mathematical Expression Recognition via Integrating Convolutional Neural Network With Transformer and Diffusion-Based Data Augmentation," *IEEE Access*, vol. 12, pp. 1–12, May 2024.

[6] J. Shin, K. Hirooka, Y. Uchida, M. Maniruzzaman, A. Megumi, and A. Yasumura, "Online Handwriting-Based Gender Recognition Using Statistical and Machine Learning Approaches," *IEEE Access*, vol. 12, pp. 1–12, July 2024.

[7] R. Malhotra and M. T. Addis, "End-to-End Historical Handwritten Ethiopic Text Recognition Using Deep Learning," *IEEE Access*, vol. 11, pp. 1–10, Sep. 2023.

[8] A. Graves et al., “A Novel Connectionist System for Unconstrained Handwriting Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855-868, 2023.

[9] U.-V. Marti and H. Bunke, “The IAM-database: An English Sentence Database for Off-line Handwriting Recognition,”

International Journal on Document Analysis and Recognition, vol. 5, no. 1, pp. 39-46, 2022.

[10] A. H. Nisa, "A Deep Learning Approach to Handwritten Text Recognition In The Presence Of Struck-Out Text," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.

[11] Andreu, "Handwritten Text Recognition for Bengali Language," in International Conference on Frontiers in Handwriting Recognition, 2013.

[12] M. L. Saini, D. C. Sati, R. S. Telikicharla, and Mahadev, "Handwritten English script recognition system using CNN and LSTM," in *Proc. IEEE Contemp. Comput. Commun. (InC4)*, Punjab, India, 2024.

[13] K. He, "Handwritten Digit Recognition Based on Convolutional Neural Network," *2023 IEEE Electronic Technology, Communication and Information (ICETCI)*, Xi'an, China, 2023.

[14] M. B. Subha, P. V. Ramanan, and V. K. T. Malini, "Offline Recognition of Handwritten Text Using Combination of Neural Networks," in Commun. Electron. Syst. (ICCES), Chennai, India, 2023.

[15] P. Latchoumy, G. Kavitha, H. S. Banu, and S. Anupriya, "Handwriting Recognition Using Convolutional Neural Network and Support Vector Machine Algorithms," in Electron., Commun., Aerosp. Technol. (ICECA), Chennai, India, 2022.

[16] B. Rajyagor and R. Rakhlia, "Handwritten Character Recognition using Deep Learning," *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 8, no.6,pp.1-3,Mar.2020.

[17] S. Y. Manchala, K. Kinthali, K. Kotha, K. S. Kumar, and J. Jayalaxmi, "Handwritten Text Recognition using Deep Learning with TensorFlow," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 9, no. 5, pp. 2-3, May2020.

[18] A. Ansari, A. Singh, B. Kaur, D. Singh, and M. Rakhra, "Handwritten Text Recognition using Deep Learning Algorithms," *2022 4th International Conference on Artificial Intelligence and Speech Technology (AIST)*, Phagwara, India, 2022.

[19] S. S. Rajput and Y. Choi, "Handwritten digit recognition using convolution neural networks," in *Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Fullerton, CA, USA, 2022.

[20] S. Chauhan, S. Mahmood, and T. Poongodi, "Handwritten digit recognition using deep neural networks," in *Intell. Eng. Manag. (ICIEM)*, Greater Noida, India, 2023.

[21] H. M. Balaha and H. A. Ali, "A new Arabic handwritten character recognition deep learning system (AHCR-DLS)," *Neural Comput. Appl.*,vol.33,pp.6325–6367,Oct.2020.

[22] K. R. Shah and D. D. Badgujar, "Devnagari Handwritten Character Recognition (DHCR) for Ancient Documents: A Review," *Proceedings of the 2013 IEEE Information and Communication Technologies (ICT 2013)*, Lonavala, India, 2023.

[23] H. Bhardwaj, R. Das, P. Garg, and R. Kumar, "Handwritten Text Recognition Using Deep Learning," *2024 Intelligence CommunicationTechnologies(CCICT)*,Gurugram,India,2024.

[24] D. Narynbayev, A. Serikkhan, A. Barkhandinova, and I. Mohammad, "Kazakh Handwritten Text Recognition Using Computer Vision and Neural Network,"*Electronics Computer and Computation (ICECCO)*, Kaskelen, Kazakhstan,2023.