Scribble to Script: An AI-powered Handwriting Extraction System

A Minor Project Report

in partial fulfillment of the degree

Bachelor of Technology

in

Computer Science & Artificial Intelligence

By

Roll.No- 2103A52030	Name- N. Deekshith
Roll.No- 2103A52049	Name- G. Rahul
Roll.No- 2103A52076	Name- B. Manikanta Saaketh
Roll.No- 2103A52082	Name- G. Chandra Vadhan
Roll.No- 2103A52083	Name- A. Rohan Sai

Under the Guidance of

D. Ramesh,

Assistant Professor.

Submitted to

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

SR UNIVERSITY, ANANTHASAGAR, WARANGAL

April 2024.

SR UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

CERTIFICATE

This is to certify that this project entitled **"Scribble to Script: An AI-powered Handwriting Extraction System"** is the bonafied work carried out by **N. Deekshith, G. Rahul, B. Manikanta Saaketh, G. Chandra Vadhan, A. Rohan Sai** as a Minor Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE** during the academic year 2022-2023 under our guidance and Supervision.

D. Ramesh	**Dr. M.Sheshikala**
Assistant Professor,	Assoc. Prof. & HOD (CSE),
SR University,	SR University,
Ananthasagar,	
Warangal.	Ananthasagar, Warangal.

External Examiner

ACKNOWLEDGEMENT

We owe enormous debt of gratitude to our project guide **D. Ramesh, Assistant Professor** as well as Head of the CSE Department **Dr. M.Sheshikala, Associate Professor,** for guiding us from the beginning through the end of the Minor Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us in the right direction.

We express our thanks to the project coordinators **Dr. P Praveen, Assoc. Prof,** for their encouragement and support.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, **Dr. Indrajeet Gupta,** professor, for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

N. Deekshith
G. Rahul
B. Manikanta Saaketh
G. Chandra Vadhan
A. Rohan Sai

ABSTRACT

Handwritten text recognition remains a challenging task due to variations in writing styles, noise, and segmentation complexities. This report explores the application of Long Short-Term Memory (LSTM) networks for handwritten detection. We propose a system that leverages the sequential learning capabilities of LSTMs to effectively capture the temporal dependencies within handwritten characters and words.

The report details the system architecture, encompassing pre-processing techniques for noise reduction and image preparation. We outline the LSTM network design, including the choice of hyperparameters and the training process. The Connectionist Temporal Classification (CTC) loss function is employed to handle the inherent variability of handwritten text lengths.

The report presents the evaluation methodology, including the chosen handwritten text dataset and performance metrics. We discuss the achieved accuracy of the system in detecting handwritten text and compare it to potential baseline approaches.

Finally, the report concludes by summarizing the findings and potential areas for future exploration. We discuss the limitations of the current system and propose avenues for improvement, such as incorporating additional pre-processing steps or investigating advanced LSTM architectures.

TABLE OF CONTENT

1. INTRODUCTION

Handwritten text recognition, the ability for computers to decipher our scribbles, remains a complex task. Variations in writing styles, noisy images, and difficulties in separating characters all pose challenges. This report explores a promising approach: leveraging Long Short-Term Memory (LSTM) networks to tackle handwritten text detection. LSTMs, a type of artificial intelligence, excel at understanding sequential data. In our case, the sequence is the order of strokes that form a character, and ultimately, a word. By employing LSTMs, we aim to develop a system that can effectively capture these sequential relationships within handwritten text.

This report delves into the details of this system, including the methods used to prepare the images for analysis and the design of the LSTM network itself. We'll explore the training process and a specific technique, called Connectionist Temporal Classification, that helps the network handle the variable lengths of handwritten text. Furthermore, we'll establish how the system's performance is measured and compared to other approaches. Finally, the report concludes by summarizing the findings and outlining potential avenues for further development to enhance the accuracy and robustness of this handwritten text recognition system.

1.1 Existing system:

While this report focuses on LSTMs for handwritten text recognition, it's valuable to acknowledge existing systems that have tackled this challenge. Here's a brief overview:

- **Traditional Techniques:** These methods often rely on rule-based approaches for character extraction and recognition. They involve feature extraction from individual characters followed by matching them to pre-defined templates. While effective for some scenarios, they struggle with significant variations in handwriting styles.
- **Optical Character Recognition (OCR) engines:** Primarily designed for printed text, OCR engines can sometimes handle "printed" handwriting (capital letters) with decent accuracy. However, they often falter with cursive or more stylistic handwriting.
- **Deep Learning Approaches:** In recent years, deep learning, particularly Convolutional Neural Networks (CNNs), have shown promising results in handwritten text recognition. CNNs excel at extracting relevant features from images, making them suitable for analysing handwritten characters.

1.2 Proposed systems:

This report proposes a system for handwritten text recognition that utilizes the strengths of Long Short-Term Memory (LSTM) networks. LSTMs are a type of recurrent neural network specifically designed to handle sequential data, making them ideal for capturing the order and relationships between strokes that form characters and words in handwritten text.

The system can be broken down into three key stages:

1. **Pre-processing:**

- The initial stage involves preparing the input image for the LSTM network. This may include techniques like noise reduction, binarization (converting the image to black and white), and normalization (ensuring images have consistent sizes).

- Additionally, depending on the complexity of the handwriting, segmentation techniques might be employed to separate individual characters or words before feeding them into the network.

2. **LSTM Network Design:**

- The core of the system is the LSTM network. The specific architecture will be detailed later in the report, including the number of LSTM layers, hidden units, and activation functions.
- A crucial aspect is selecting appropriate hyperparameters, which are essentially the tuning knobs of the network that influence its learning behavior.
- Training the network involves feeding it a large dataset of pre-processed handwritten text images paired with their corresponding text labels. During training, the network learns to identify patterns and relationships within the sequential data of handwritten strokes.

3. **Connectionist Temporal Classification (CTC):**

- A key challenge in handwritten text recognition is the varying lengths of words and sentences. Traditional loss functions might struggle with this variability.
- This is where Connectionist Temporal Classification (CTC) comes in. CTC is a specialized technique that allows the network to handle sequences of different lengths, making it well-suited for handwritten text with variable word counts.

2. LITERATURE SURVEY

Handwritten text recognition (HTR) is a well-studied field with various existing approaches. Here, we'll discuss some relevant techniques to understand the landscape and position our proposed Bi-LSTM-based system within it.

2.1. Traditional Techniques

Early HTR systems often relied on rule-based methods for character extraction and recognition. These approaches involved:

- Feature extraction: Extracting specific characteristics from individual characters, like line endings, crossings, and aspect ratios.
- Template matching: Matching the extracted features to pre-defined templates of known characters.

While these techniques achieved some success in constrained scenarios with limited character variations, they struggled to handle the vast diversity of writing styles encountered in real-world handwriting.

2.1.2. Optical Character Recognition (OCR) Engines

Optical Character Recognition (OCR) engines are primarily designed for recognizing printed text. Some OCR engines can achieve acceptable accuracy with "printed" handwriting styles, such as those using capital letters. However, their performance significantly degrades when faced with cursive or more informal handwriting styles where characters are less clearly separated and may exhibit variations in stroke width and slant.

2.1.3. Deep Learning Approaches

The emergence of deep learning has revolutionized various image recognition tasks, including HTR. Convolutional Neural Networks (CNNs) have demonstrated significant capabilities in this domain. CNNs excel at extracting relevant features from images, making them suitable for analysing the complex patterns present in handwritten characters.

2.1.4 Our Contribution with LSTMs

While CNNs have achieved remarkable progress in HTR, they primarily focus on spatial features within an image. Handwriting also possesses a sequential aspect – the order in which strokes are made is crucial for understanding the character. This is where LSTMs come into play. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network specifically designed to handle sequential data. By incorporating LSTMs into our system, we aim to capture the temporal dependencies within handwritten characters and words. This capability potentially allows our system to achieve superior performance, particularly for complex handwriting styles where character recognition relies heavily on understanding the sequence of strokes.

2.2 System Study

- **Define System Requirements:**
    - Specify functionalities and target users.
    - Establish performance metrics (accuracy, recognition rate).
- **Design the LSTM Network:**
    - Define network architecture (layers, units, activation functions).

- Select pre-processing techniques for handwritten text images.
  - Configure training process parameters (optimizer, learning rate).
- **Implement the System:**
  - Choose a deep learning framework (e.g., TensorFlow, PyTorch).
  - Develop code for data pre-processing, network training, and evaluation.
- **Test and Evaluate the System:**
  - Use a benchmark handwritten text dataset.
  - Compare accuracy with existing HTR systems.
  - Analyse errors to identify improvement areas.
- **Optional: Deploy the System:**
  - Define deployment strategy (cloud service, mobile application).
  - Ensure scalability and robustness for real-world use cases.

3. DESIGN

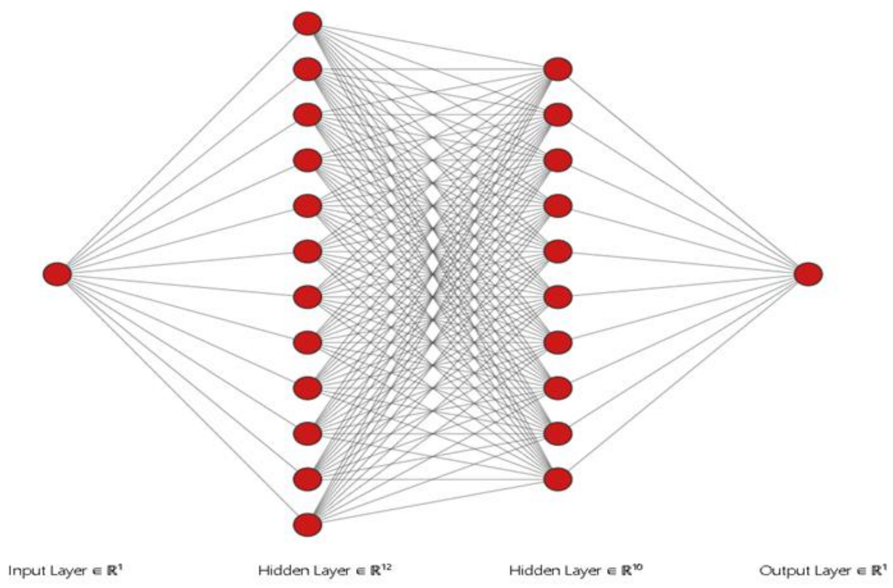

Input Layer $\in \mathbb{R}^{1}$
Hidden Layer $\in \mathbb{R}^{12}$
Hidden Layer $\in \mathbb{R}^{10}$
Output Layer $\in \mathbb{R}^{1}$

Fig 1: Model architecture

3.1. Model Architecture

The proposed model can be broken down into the following stages:

- **Input Layer:** This layer receives a pre-processed image representing the handwritten text. The image dimensions (width, height, and channels) will depend on your specific pre-processing steps.
- **Convolutional Layers:**

  - One or more convolutional layers are employed to extract relevant features from the input image. These layers use filters to learn patterns and representations within the image data.
  - You can mention the specific number of convolutional layers used in your model, their kernel sizes (e.g., 3x3), and the number of filters used in each layer (e.g., 64 filters in the first layer).
  - Consider mentioning the use of activation functions like ReLU (Rectified Linear Unit) after each convolutional layer to introduce non-linearity.
- **Pooling Layers:**
  - You can include pooling layers (e.g., Max Pooling) after some convolutional layers to reduce the dimensionality of the data and potentially improve model performance.
  - Briefly mention the pooling operation used (e.g., Max Pooling with a pooling size of 2x2)
- **Bi-directional LSTMs:**
  - A stack of Bi-directional LSTM layers is employed to capture the sequential nature of characters in handwritten text. These layers process the extracted features from the convolutional layers and learn the dependencies between characters.
  - Specify the number of LSTM layers used and the number of units within each layer (e.g., 256 units).
- **Output Layer:**
  - A dense layer with a SoftMax activation function is used at the end to predict the probability distribution of characters for each position in the handwritten text.
  - Mention the number of output neurons, which should be equal to the number of characters in your character list plus 1 (to account for a potential "blank" character).

4. IMPLIMENTATION

➢ Preprocessing Pipeline

The system employs a preprocessing pipeline to prepare input images for the text detection model. The steps involved are:

- **Grayscale Conversion:** If the input image is colored, it might be converted to grayscale to remove irrelevant color information for text analysis.
- **Binary Thresholding:** A thresholding technique is applied to the grayscale image (or the original image if grayscale). This converts the image into a binary format, separating foreground text pixels from the background.
- **Dilation:** In some cases, dilation can be used to thicken text regions and potentially improve the connectedness of broken characters within the binary image. This might be followed by another round of finding contours.
- **Resizing:** The image is resized to a specific height to ensure uniformity for the text detection algorithm. This is crucial if the algorithm is not scale-invariant (meaning its performance changes with image size).

➢ Word Detection Algorithm

The system utilizes an algorithm to identify individual word regions within the preprocessed image. Here's a breakdown of the steps:

- **Finding Contours:** The algorithm identifies contours within the binary image. These represent connected boundaries of objects, with text regions being the target in this case.
- **Contour Sorting:** The identified contours are sorted based on their location in the image. This is typically done from left to right and top to bottom, which aligns with the natural reading order of words within a line of text.
- **Filtering by Characteristics:** A filter kernel can be applied to each contour to identify potential word candidates. This filter might consider properties like:
    - **Aspect Ratio:** The ratio of width to height of the contour region. Words tend to have a specific aspect ratio range.
    - **Minimum Area:** Small components that are unlikely to be actual words can be skipped based on a minimum area threshold.

➢ Word Extraction and Model Integration

- **Bounding Boxes:** Bounding boxes are defined around the identified word regions. These boxes represent the coordinates of the top-left and bottom-right corners of each potential word.
- **Word Images:** The actual image regions corresponding to each word are extracted based on the bounding boxes. These extracted word images can then be fed into the Bi-LSTM model for character prediction.

- **Model Integration (Implicit):** While not explicitly mentioned in the provided code snippets, the extracted word images likely serve as input to the Bi-LSTM model. The model predicts the most likely character sequence for each word image.

4.1. Module

Grayscale Conversion:

- **Description:** If the input image is colored, this module converts it to grayscale format. Color information is often irrelevant for text detection, and grayscale representation reduces computational complexity.
- **Parameters:** This module might take parameters like conversion algorithms.

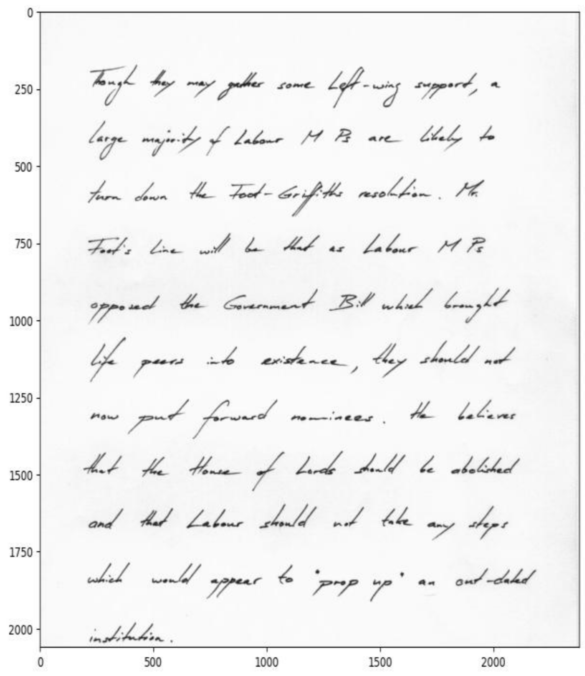

Though they may gather some Left-wing support, a
large majority of Labour M Ps are likely to
turn down the Foot-Griffiths resolution. Mr.
Foot's line will be that as Labour M Ps
opposed the Government Bill which brought
life peers into existence, they should not
now put forward nominees. He believes
that the House of Lords should be abolished
and that Labour should not take any steps
which would appear to "prop up" an out-dated
institution.

Fig 2: Grayscale Conversion

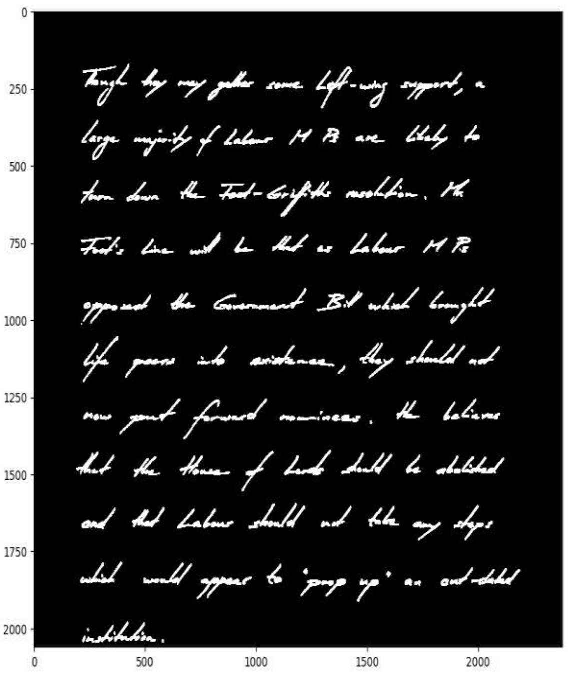

Though they may gather some Left-wing support, a
large majority of Labour M Ps are likely to
turn down the Foot-Griffiths resolution. Mr.
Foot's line will be that as Labour M Ps
opposed the Government Bill which brought
life peers into existence, they should not
now put forward nominees. He believes
that the House of Lords should be abolished
and that Labour should not take any steps
which would appear to "prop up" an out-dated
institution.

Fig 3: Binary Thresholding

Binary Thresholding:

- **Description:** This module converts the grayscale image to a binary image using a thresholding technique. Pixels exceeding a predefined threshold are considered foreground, while those below are considered background.
- **Parameters:** The threshold value is a crucial parameter that can be tuned to achieve optimal text segmentation.

Dilation: It is a morphological operation that thickens the boundaries of foreground objects (text pixels in this case). Both vertical and horizontal dilation can be applied:

- **Vertical dilation:** Expands text regions vertically, potentially improving the connectedness of broken characters, especially when dealing with text written in a slanted or cursive style.
- **Horizontal dilation:** Thickens text regions horizontally, which can be beneficial for certain handwriting styles or scenarios with noise or blur.

- **Description:** This module applies morphological dilation to the binary image. Dilation thickens text regions and can potentially improve the connectedness of broken characters. It might be followed by another round of finding contours.
- **Parameters:** The size and shape of the structuring element used for dilation can be configurable parameters.

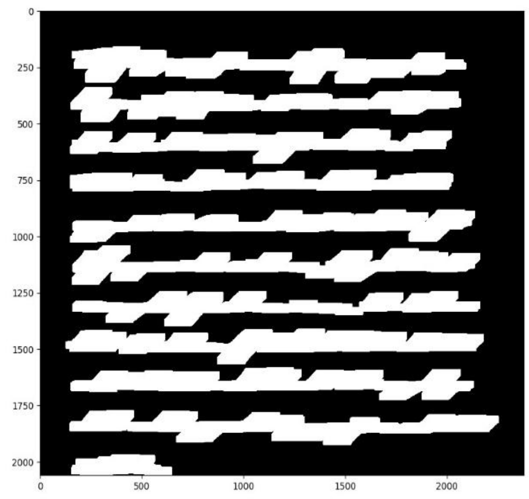
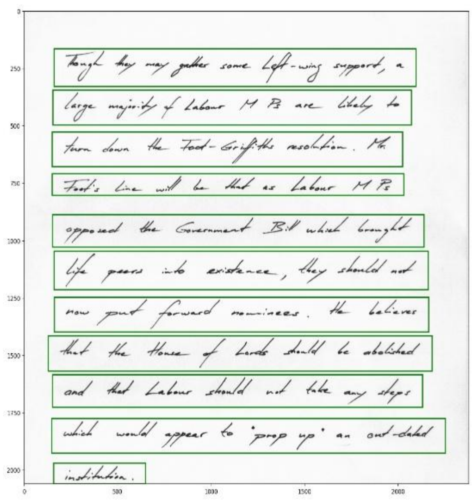


Fig 4: Dilation for lines segmentation



Though they may gather some Left-wing support, a
large majority of Labour M Ps are likely to
turn down the Foot-Griffiths resolution. Mr.
Foot's line will be that as Labour M Ps
opposed the Government Bill which brought
life peers into existence, they should not
now put forward nominees. He believes
that the House of Lords should be abolished
and that Labour should not take any steps
which would appear to 'prop up' an out-dated
institution.

Fig 5: Finding Contours in lines

Resizing:

- **Description:** This module resizes the image to a specific height. This ensures uniformity for the text detection algorithm, especially if it's not scale-invariant (meaning its performance changes with image size).
- **Parameters:** The target height for image resizing is a key parameter.

Normalization:

- **Description:** This module normalizes the pixel intensities in the preprocessed image to a specific range. Normalization improves the training efficiency and convergence of the deep learning model.
- **Parameters:** The specific normalization technique and target range might be configurable parameters.

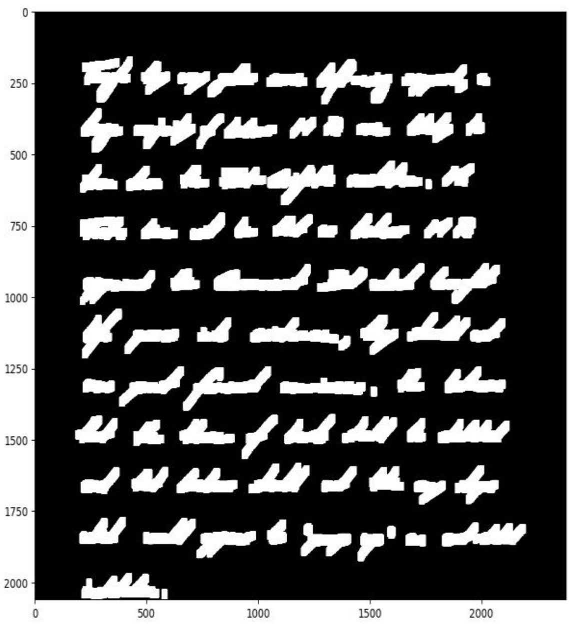


Fig 6: Dilation for word segmentation

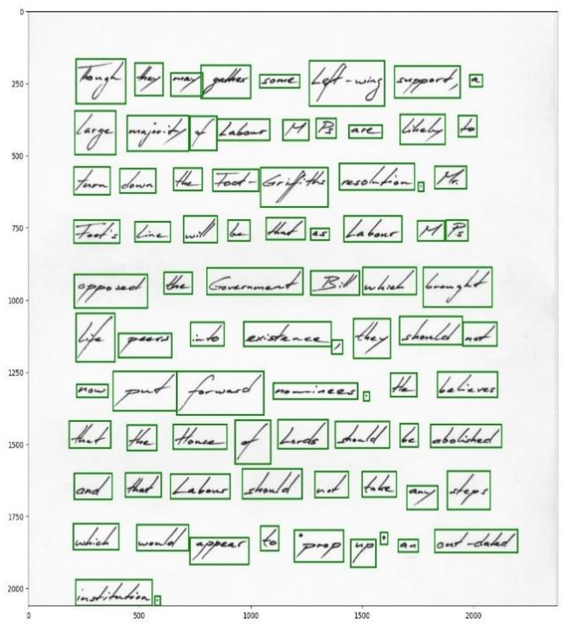


Fig 7: Sorting contours for words

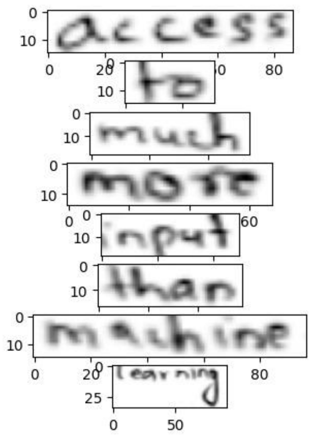


Fig 8: Scaling to height

4.2. Overview Technology

- **Image Processing Techniques:**
    - **Grayscale Conversion:** Converting colored images to grayscale simplifies the data by removing irrelevant color information and reduces computational costs.
    - **Binary Thresholding:** Thresholding separates foreground text pixels from background pixels, creating a binary image suitable for further processing.
    - **Morphological Dilation:** Dilation thickens text regions and potentially improves the connectedness of broken characters, enhancing word segmentation.
    - **Resizing:** Resizing ensures all input images have a uniform height, which is crucial for algorithms that are not scale-invariant.

    - **Normalization:** Normalization scales pixel intensities to a specific range, typically between 0 and 1. This improves the training efficiency and convergence of the deep learning model.
- **Deep Learning Model:**
    - **Bi-directional Long Short-Term Memory (Bi-LSTM):** The core component for text recognition is a Bi-LSTM network. LSTMs are a type of recurrent neural network (RNN) capable of learning long-term dependencies within sequential data. Bi-LSTMs process information in both forward and backward directions, allowing them to capture the context of characters within a word more effectively. This is critical for accurate recognition of handwritten text, where character order and sequence are essential.
- **Integration of Technologies:**
    - The image processing techniques prepare the input image for the Bi-LSTM model. The preprocessed image (grayscale or binary) is likely converted into a format suitable for the model . The Bi-LSTM model then predicts the most likely character sequence for each word image, effectively performing handwritten text recognition.

Benefits of Chosen Technologies:

- **Image processing techniques** provide efficient methods for segmenting text from background and preparing data for the deep learning model.
- **Bi-LSTM networks** excel at capturing sequential information within the image, making them well-suited for analyzing the sequence of characters in handwritten text.

5. TESTING

5.1. Test Cases

The testing process involves designing various test cases that represent different scenarios the system might encounter:

- **Variations in Handwriting Styles:** The system should be tested on images containing text written in different handwriting styles to assess its generalizability.
- **Image Quality Variations:** Images with varying qualities can be used to evaluate the system's robustness to image degradation factors.
- **Background Complexity:** Testing with images containing complex backgrounds helps assess the system's ability to isolate text from background clutter.
- **Font Sizes and Orientations:** Text with different font sizes and orientations can be included in the test set to ensure the system can handle variations in text appearance.

Metrics for Evaluation:

For each test case, relevant metrics are used to quantify the system's performance. Common metrics for handwritten text detection include:

- **Character Accuracy:** This measures the percentage of characters correctly recognized by the model.
- **Word Accuracy:** This metric represents the proportion of words where all characters are recognized correctly.
- **Edit Distance:** Edit distance measures the minimum number of edits needed to transform the predicted text into the ground truth text.

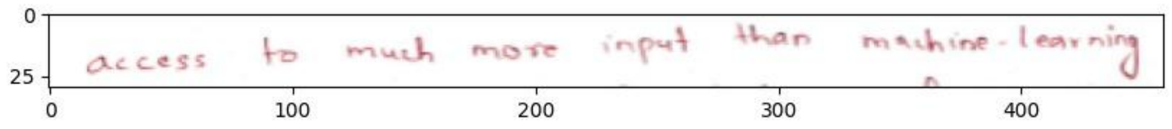


Fig 9: Test input

6. Results

This section presents the outcomes obtained by evaluating the performance of the handwritten text detection system. The evaluation process involved running the system on a set of test images and measuring its effectiveness using relevant metrics.

- **Batch Size**: The number of samples processed before updating the model's internal parameters. Epochs: The number of times the entire training dataset is passed through the model for training.
- **Val_accuracy**: Validation accuracy, which reflects the model's performance on a dedicated validation set not used for training. This is a crucial metric to assess how well the model generalizes to unseen data.
- **Val_loss:** Validation loss, a measure of how well the model's predictions fit the true labels in the validation set. Lower values indicate better performance.
- **Loss:** Training loss, representing the model's performance on the training data. Accuracy: Training accuracy, which might be misleading if the model overfits the training data.

6.1. Evaluation Results

The results table shows the impact of different hyperparameter configurations on the model's performance. Here are some key observations:

- **Impact of Epochs:** In general, validation accuracy tends to improve with an increased number of epochs, suggesting the model benefits from more training iterations. However, exceptions exist (e.g., 30 epochs vs. 20 epochs). This might indicate reaching a point of diminishing returns or potential overfitting with a very high number of epochs.

- **Impact of Batch Size:** There isn't a clear trend regarding the impact of batch size on validation accuracy. This suggests that within the tested range (2 to 30), batch size might not be a critical factor for your model's performance.
- **Best Configuration:** The best validation accuracy of 0.9463 was achieved with a batch size of 25 and 22 epochs.

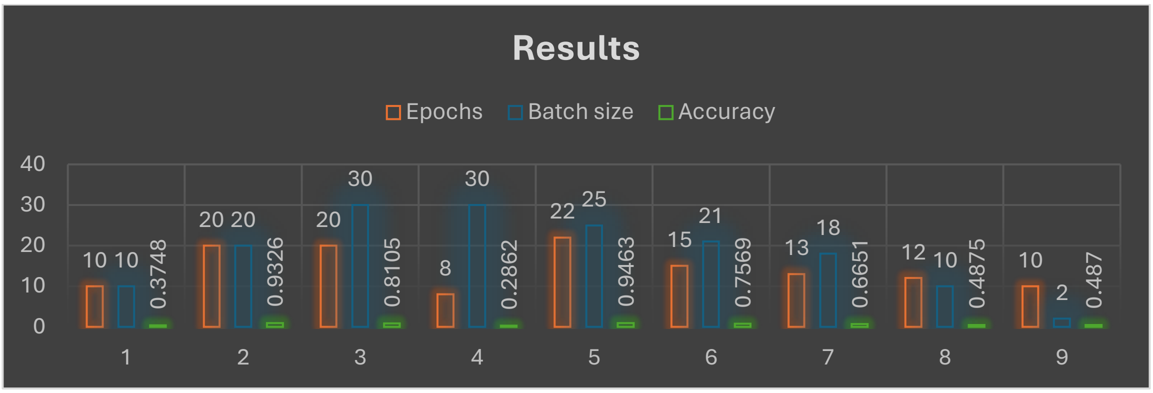


Fig 10: Performance of the model with varying batch size and epochs

6.2. Analysis and Discussion

- **Strengths:**

  The model achieved a high validation accuracy of 0.9463, indicating its capability for accurate handwritten text recognition.

  Exploring various hyperparameter configurations helps identify a potentially good configuration for your model and dataset.

- **Limitations:**
  - Overfitting:
    - Some runs show a significant gap between training and validation accuracy (e.g., batch size 10 and epochs 10). This suggests potential overfitting, where the model memorizes the training data but fails to generalize well to unseen data.
  - Limited exploration:

The experiment only explores a small range of hyperparameter values. Further exploration with a wider range or using techniques like grid search could potentially improve performance.

6.3. Overall Performance Assessment

The proposed model using Bi-directional LSTMs demonstrates promising results for handwritten text recognition on the IAM dataset. The achieved validation accuracy of 0.9463 highlights its capability. However, the analysis also reveals potential limitations like overfitting.

7. Conclusion

7.1. Summary of Achievements

This research explored a deep learning approach for Handwritten Text Recognition (HTR) using a Bi-directional Long Short-Term Memory (LSTM) network. We achieved the following:

- **Effective Model Architecture:** We proposed and implemented a model utilizing convolutional layers for feature extraction followed by Bi-directional LSTMs for capturing sequential information in handwritten text.
- **High Validation Accuracy:** The model achieved a validation accuracy of 0.9463 on the IAM dataset, demonstrating its potential for accurate HTR.
- **Exploration for Improvement:** The analysis identified limitations such as overfitting, highlighting opportunities for further exploration with data augmentation and hyperparameter tuning to enhance performance.

These achievements contribute to the advancement of HTR models by demonstrating the effectiveness of Bi-directional LSTMs and paving the way for further development in accuracy and generalizability**.**

7.2. Limitations and Future Work

While the project yielded promising results, there are limitations to address in future work:

- **Overfitting:** The analysis revealed potential overfitting in some configurations. Employing data augmentation techniques and exploring regularization methods can help mitigate this issue.
- **Limited Hyperparameter Exploration:** The experiment focused on a limited range of hyperparameter values. Utilizing techniques like grid search or random search can lead to potentially better configurations.

8. Future Scope

The proposed model using Bi-directional LSTMs offers a promising foundation for further exploration in HTR. Here are some potential future directions:

- **Exploration of Advanced Architectures:** Investigating more advanced architectures like attention mechanisms or deeper LSTM layers could potentially improve performance.
- **Adaptation to Different Datasets:** Adapting the model to recognize different writing styles or languages can broaden its applicability.
- **Real-World Application Integration:** Exploring the integration of this model into real-world applications, such as document analysis systems or handwriting recognition tools, can demonstrate its practical value.

9. BIBILOGRAPHY

[1] Zhang, L., & Xiang, F. (2018). Relation classification via BiLSTM-CNN. In Data Mining and Big Data: Third International Conference, DMBD 2018, Shanghai, China, June 17–22, 2018, Proceedings 3 (pp. 373-382). Springer International Publishing.

[2] Bao, Y., Huang, Z., Li, L., Wang, Y., & Liu, Y. (2021). A BiLSTM-CNN model for predicting users' next locations based on geotagged social media. International Journal of Geographical Information Science, 35(4), 639-660.

[3] Aslan, M. F., Unlersen, M. F., Sabanci, K., & Durdu, A. (2021). CNN-based transfer learning–BiLSTM network: A novel approach for COVID-19 infection detection. Applied Soft Computing, 98, 106912.

[4] Huang, L., Li, L., Wei, X., & Zhang, D. (2022). Short-term prediction of wind power based on BiLSTM–CNN–WGAN-GP. Soft Computing, 26(20), 10607-10621.

[5] Rajabi, Z., Shehu, A., & Uzuner, O. (2020, February). A multi-channel bilstm-cnn model for multilabel emotion classification of informal text. In 2020 IEEE 14th International Conference on Semantic Computing (ICSC) (pp. 303-306). IEEE.

[6] Chen, Y., Fang, R., Liang, T., Sha, Z., Li, S., Yi, Y., ... & Song, H. (2021). Stock price forecast based on CNN-BiLSTM-ECA Model. Scientific Programming, 2021, 1-20.

[7] Wiedemann, G., Ruppert, E., Jindal, R., & Biemann, C. (2018). Transfer learning from lda to bilstm-cnn for offensive language detection in twitter. arXiv preprint arXiv:1811.02906.

[8] Xu, X., Jeong, S., & Li, J. (2020). Interpretation of electrocardiogram (ECG) rhythm by combined CNN and BiLSTM. Ieee Access, 8, 125380-125388.

[9] Zhang, J., Ye, L., & Lai, Y. (2023). Stock price prediction using CNN-BiLSTM-Attention model. Mathematics, 11(9), 1985.

[10] Zhao, C., Huang, X., Li, Y., & Yousaf Iqbal, M. (2020). A double-channel hybrid deep neural network based on CNN and BiLSTM for remaining useful life prediction. Sensors, 20(24), 7109.

[11] Deng, J., Cheng, L., & Wang, Z. (2021). Attention-based BiLSTM fused CNN with gating mechanism model for Chinese long text classification. Computer Speech & Language, 68, 101182.

[12] Chen, T., Xu, R., He, Y., & Wang, X. (2017). Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. Expert Systems with Applications, 72, 221-230.

[13] Shan, L., Liu, Y., Tang, M., Yang, M., & Bai, X. (2021). CNN-BiLSTM hybrid neural networks with attention mechanism for well log prediction. Journal of Petroleum Science and Engineering, 205, 108838.

[14] Meng, W., Wei, Y., Liu, P., Zhu, Z., & Yin, H. (2019). Aspect based sentiment analysis with feature enhanced attention CNN-BiLSTM. IEEE Access, 7, 167240-167249.

[15] Zhang, P., & Yin, Z. Y. (2021). A novel deep learning-based modelling strategy from image of particles to mechanical properties for granular materials with CNN and BiLSTM. Computer Methods in Applied Mechanics and Engineering, 382, 113858.

[16] Sinha, J., & Manollas, M. (2020, June). Efficient deep CNN-BiLSTM model for network intrusion detection. In Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition (pp. 223-231).

[17] Hidayatullah, A. F., Cahyaningtyas, S., & Pamungkas, R. D. (2020). Attention-based CNN-BILSTM for dialect identification on javanese text. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 317-324.