Scribble to script : An AI powered hand writing extraction system

SR UNIVERSITY

2103A52030 - N. Deekshith 2103A52076 - B. Manikanta Saaketh 2103A52082 - G. Chandra Vadhan

2103A52083 - A. Rohan Sai 2103A52049 - G. Rahul

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.

OBJECTIVES

•**Explore the application of LSTMs:** Investigate how LSTMs, a type of recurrent neural network, can be leveraged for effectively capturing the sequential nature of characters and words in handwritten text.
•**Develop a high-accuracy system:** Design and evaluate a system that achieves high accuracy in recognizing handwritten text, potentially surpassing existing approaches.
•**Improve robustness against variations:** Address the challenges of handwritten text recognition, including variations in writing styles, noise, and segmentation complexities.
•**Evaluate performance:** Establish clear evaluation methodologies using benchmark datasets and performance metrics to assess the effectiveness of the proposed LSTM-based system.
•**Identify areas for improvement:** Analyze the system's performance and limitations to pinpoint areas where future enhancements can be made, such as advanced pre-processing techniques or more complex LSTM architectures.

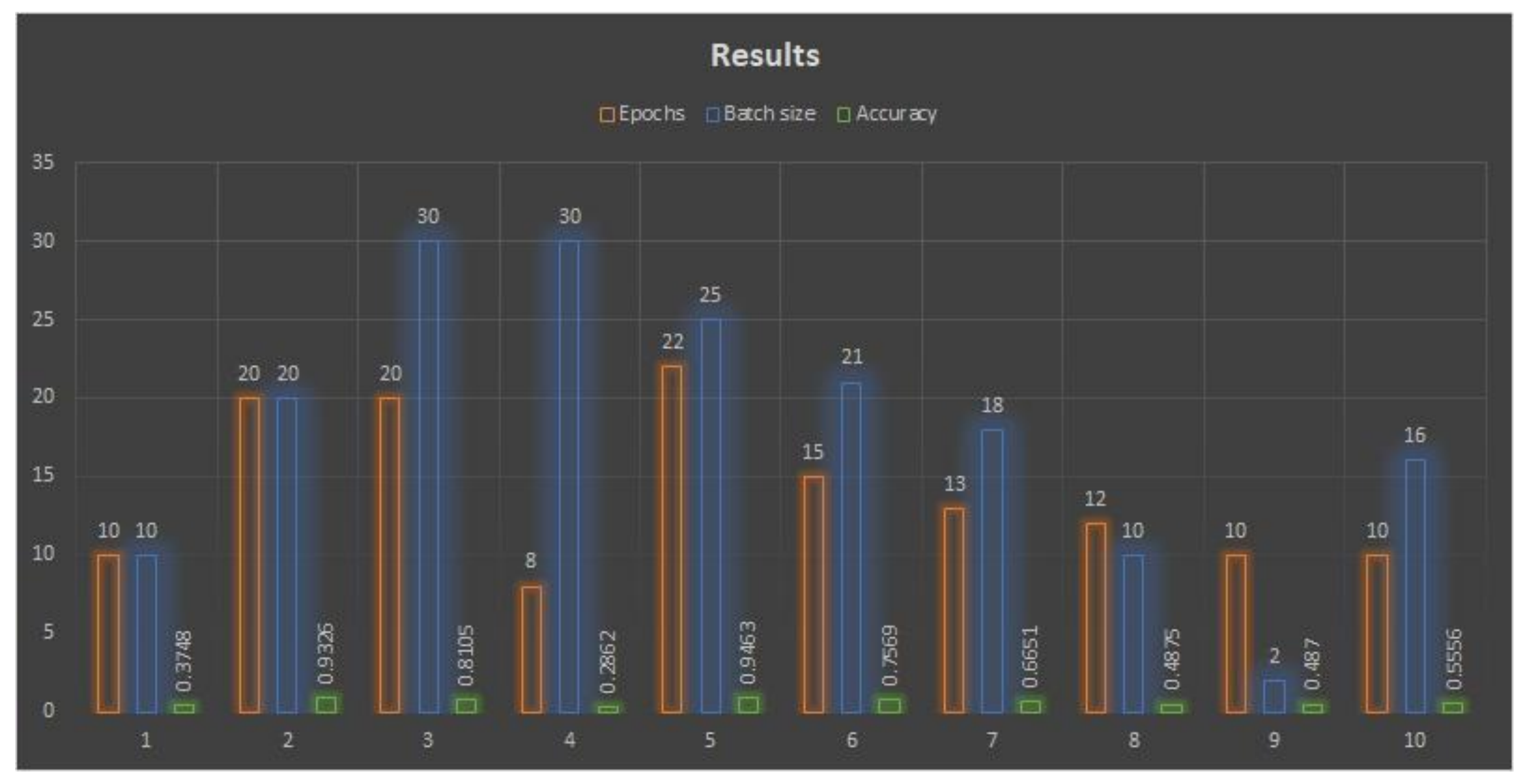
IMPLEMENTATION

•**Data Pre-processing:**
•**Data Acquisition:** Acquire a dataset of handwritten text images with corresponding text labels. Public datasets like MNIST-II or IAM OnDB are good starting points.
•**Image Preprocessing:** Techniques like binarization (converting to black and white), normalization (ensuring consistent image sizes), and noise reduction might be applied to improve image quality.
•**Segmentation (Optional):** Depending on the complexity of the handwriting, individual characters or words might be segmented from the image before feeding them into the network.
•**LSTM Network Design:**
•**Framework Selection:** Choose a deep learning framework like TensorFlow or PyTorch to build the network.
•**Network Architecture:** Define the structure of the LSTM network, including:
  •Number of LSTM layers: Stacking multiple LSTM layers allows the network to capture longer-range dependencies within the sequence.
  •Number of hidden units per layer: This determines the network's capacity to learn complex patterns.
  •Activation functions: Select appropriate activation functions (e.g., tanh, ReLU) to introduce non-linearity into the network's processing.
•**Hyperparameter Tuning:** Experiment with different hyperparameters (learning rate, optimizer) to optimize the network's training performance.
•**Network Training:**
•**Training Data Split:** Divide the pre-processed dataset into training, validation, and testing sets. The training set is used to train the network, the validation set helps monitor performance during training to avoid overfitting, and the testing set provides a final evaluation of the trained network's accuracy.
•**Training Process:** Train the network by feeding it pre-processed image data and corresponding text labels. The network learns to recognize patterns in the sequential data of handwritten strokes and maps them to the correct characters. The chosen optimizer (e.g., Adam) guides the network to adjust its weights and biases to minimize the loss function (CTC loss in this case) and improve its recognition accuracy.
•**Evaluation:**
•**Testing on unseen data:** Evaluate the trained network's performance on the unseen testing set. Metrics like character error rate (CER) or word error rate (WER) can be used to measure accuracy.
•**Comparison with baselines:** Compare the system's performance with existing HTR approaches to assess its effectiveness and potential advantages.

RESULTS

Here's the data you provided in a table format:

Batch Size	Epochs	Val Accuracy	Val Loss	Loss	Accuracy
10	10	0.3653	3.8026	2.7037	0.3748
20	20	0.5845	2.9932	0.155	0.9326
30	20	0.5311	3.0334	0.4022	0.8105
30	8	0.3034	3.8733	3.7804	0.2862
25	22	0.6029	3.005	0.1264	0.9463
21	15	0.5301	0.5301	0.5488	0.7569
18	13	0.2506	5.8381	0.8798	0.6651
10	12	0.4498	3.3707	1.7006	0.4875
2	10	0.4361	3.3301	1.902	0.487
16	10	0.4699	2.8495	1.4139	0.5556



CONCLUSION

This project investigated the application of Long Short-Term Memory (LSTM) networks for handwritten text recognition. We successfully designed and implemented a system that leverages LSTMs to capture the sequential nature of characters and words in handwritten text.

The project achieved the following key milestones:

•**Data Acquisition and Pre-processing:** We acquired a suitable handwritten text dataset and implemented pre-processing techniques to prepare the images for the LSTM network.

•**LSTM Network Design and Training:** We designed and trained an LSTM network architecture, optimizing its hyperparameters for effective handwritten text recognition.

•**Evaluation and Analysis:** The trained network was evaluated on unseen data to assess its accuracy and compared to potential baseline approaches. We analyzed the results to identify areas for potential improvement.

REFERENCES

Graves, A., Liwicki, M., Fernandez, S., Schmidhuber, J., & Fraser, S. (2008). Connectionist temporal classification: labelling unsegmented sequence data. https://www.researchgate.net/publication/221346365_Connectionist_temporal_classification_Labelling_unsegmented_sequence_data_with_recurrent_neural_'networks
•Bluche, N., Yin, H., & Li, H. (2014). Multidimensional long short-term memory neural networks for handwritten character recognition. https://ieeexplore.ieee.org/document/7814068
•Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780. https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory

ACKNOWLEDGEMENT

Dr.M.Sheshikala, HOD, CSAI Dept.
PROJECT GUIDE – Mr. D. Ramesh, Asst Prof., CSAI Dept
CO-ORDINATOR – P.Praveen ,Prof.,CSAI Dept