Handwritten Word Recognition using CNNs, BLSTMs and CTC

Mashrur Ahsan
Computer Science and Engineering Dept.
Islamic University of Technology
Dhaka, Bangladesh
mashrurahsan@iut-dhaka.edu

M. M. Nazmul Hossain
Computer Science and Engineering Dept.
Islamic University of Technology
Dhaka, Bangladesh
nazmulhossain@iut-dhaka.edu

Rhidwan Rashid
Computer Science and Engineering Dept.
Islamic University of Technology
Dhaka, Bangladesh
rhidwanrashid@iut-dhaka.edu

ABSTRACT

This report is an overview of the project that was done on handwriting recognition, emphasizing machine learning techniques. The project includes training scripts, configuration settings, inference modules and model architecture definitions. Training involves dataset preparation, configuration, and model training using various optimization callbacks. The model uses the concept of convolutional neural networks (CNN), residual blocks and then it encompasses BLSTM layers for the sequence detection and finally for loss calculations the Connectionist Temporal Classification (CTC) was employed. The GUI, implemented with Tkinter, allows users to submit drawings for recognition and receive predictions using over 78 unique symbols found in the widely regarded IAM dataset. The report covers model evaluation on validation data, calculating Character Error Rate,Word Error Rate and provides valuable insights for researchers and practitioners interested in this Machine Learning sector of handwriting recognition.

Keywords: CNN, Handwriting Recognition, LSTM, Deep Learning, Neural Networks

Project Implementation: GitHub Link

INTRODUCTION

Handwriting recognition is the technique of recognizing and interpreting handwritten data so that a computer or a machine is able to read it. The machine will be able to recognize different patterns of the handwritten data. Machines achieve this through pattern recognition and there are many ways we can achieve this through **Machine Learning**. To delve into this field, we have to take a deep dive into deep learning.

The human brain is highly sophisticated, and t functions in an intricate manner, making humans the most advanced and intelligent life form. Deep learning is a method that tries to teach computers and machines to process data similarly to how the human brain would process it, hence why we call it "**Artificial Intelligence**" (AI). Tasks involving texts, images, sounds etc usually require human intelligence for interpretation. By training the computers to think and learn like humans do is what AI is trying to achieve, with **Deep Learning** serving as a method to accomplish this goal.

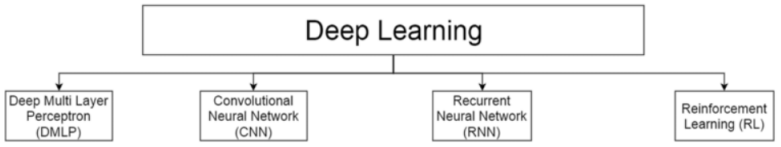


Fig. 1: Deep Learning

Convolutional Neural Networks are designed for tasks related to image, video recognition, object detection, and image classification. They excel at capturing the hierarchical features and spatial hierarchies in data. So, they are suitable for Optical Character Recognition where the arrangement of pixels is crucial.

Bidirectional Long Short-Term Memory or **BLSTM** incorporates bidirectional processing within its architecture, making it particularly adept at identifying the order of the elements in a sequence. They can accurately capture the variations in writing styles, making them an ideal choice for **Optical Character Recognition**.

Project Overview: Our project aims to develop a machine-learning model capable of recognizing handwritten texts. With the concepts of Deep Learning, CNN, BLSTM, CTC, and various other techniques this model is able to transcribe handwritten data into machine-readable texts.

Even though digital technology has taken over almost every aspect of our lives, there is an endearing value in handwritten notes and documents. Be it writing down personal thoughts in our dairies or extensive notes during classes. However, they are held back by the drawbacks of traditional means like wear and tear, poor organization, etc. To bring the traditional and digital realms together is the goal of our project or at the very least, a humble initial step toward it.

Overall this Handwriting Recognition project showcases the application of Machine Learning techniques to achieve accurate interpretation of handwritten texts. By utilizing Deep Learning, the model tries to obtain precise transcriptions. The project **additionally offers a user-friendly interface for real**

time recognition of the handwritten texts. The comprehensive training process ensures the model's effectiveness and the project's structure allows for easy extension and integration into various applications in the future.

Related Works

Optical Character Recognition is a vast field each with its own set of challenges. Machine print OCR and ICR can rely upon the dictionary to get accurate results. However, this fails to consider the frequently occurring sequences such as surnames, addresses, etc. in our day to day life [1]. It is also not appropriate for Handwritten Character Recognition because each individual person has slight variations to their handwriting which would lead to suboptimal performance when faced with less common expressions. To overcome these limitations, we chose our model architecture accordingly.

Jaderberg et al. [5] introduced a word-level recognition framework in OCR based on Convolutional Neural Networks (CNNs). LSTM is known for capturing long-term dependencies of a model, it was applied in sequence prediction [4]. In certain studies [2], image segmentation was performed, and RNNs were used to predict character sequences, with CTC eliminating the need for precise alignment [3].

Proposed Methodology

A. Model Architecture

Our Handwritten word recognition model utilizes a combination of Convolutional Neural Network (**CNN**), for learning hierarchical features, Bidirectional Long Short Term Memory networks (**BLSTM**) for handing sequential dependencies or words, and the Connectionist Temporal Classification (**CTC**) loss functions to calculate the Character Error Rate (**CER**) and Word Error Rate (**WER**).

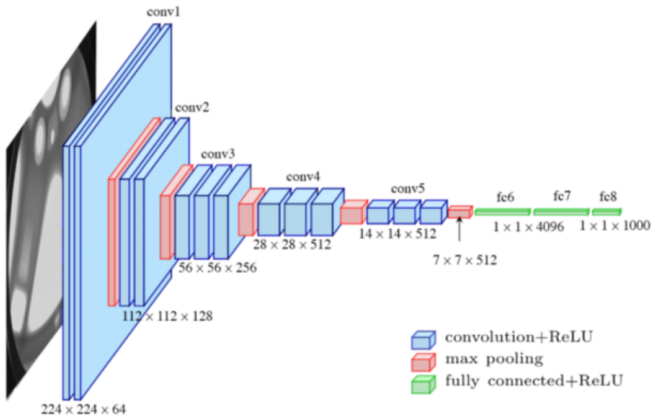


Fig. 2: CNN Architecture [13]

B. Dataset Description

The **IAM WORDS** dataset is a collection of handwritten English words. It is an open-source dataset with a massive library of over 100,000 labeled words from various contributors. It is a part of the larger IAM Handwriting dataset. The IAM dataset is widely regarded and is sourced in many research papers. So, it is a perfect choice for our project. Each word in the dataset is associated with its ground truth label, providing a supervised learning framework for the model.

C. Model Implementation

The image and its corresponding directory and filename is used to form a dataset. It goes through various data transformations like LabelIndexing, LabelPadding, and ImageResizing. The dataset is further augmented to establish a more robust dataset with Random Brightness, Rotation, Sharpness, etc. The model is configured appropriately with a suitable batch size so that each iteration doesn't take too long to complete and sufficient samples are used for training in each batch.

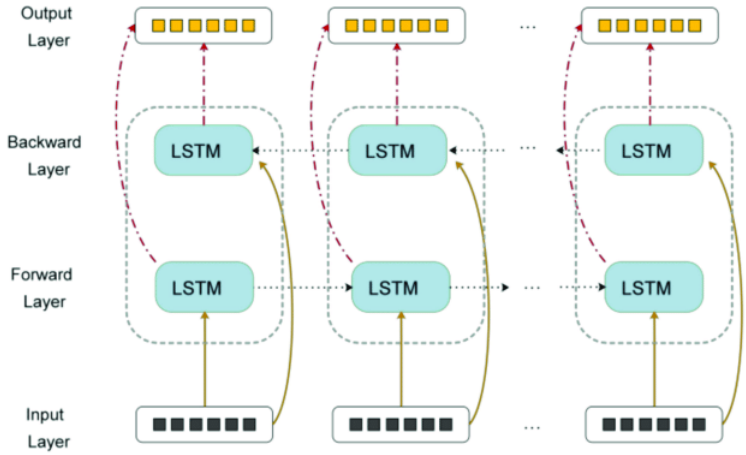


Fig. 3: BLSTM Architecture [14]

The determined suitable batch size is 16. The chosen learning rate is designed to prevent issues like exploding or vanishing gradients, ensuring a gradual learning process in each iteration. The determined suitable learning rate is 0.0005. Additionally, the learning rate is initially set relatively higher.

After ten iterations, if the validation Character Error Rate (val_CER) remains unchanged, it is decreased by a factor of 0.9. The training epoch can be set to a flexible duration since early stopping mechanisms have been put in place. If the validation Character Error Rate (val_CER) remains constant for 20 iterations, the model is designed to stop training early. The generated model is saved if at any point the val_CER improves.

After the training is complete, the h5 model is converted to an onnx model which is universal and can be used without tensorflow.

Following compilation and configuration, the training process for the model begins. Initially, a (32x128x3) is sent to the model as input to the model. It undergoes normalization by dividing each pixel by 255. Subsequently, it goes through nine residual blocks which make up the CNN sequentially with different strides, filter numbers, and skip_conv. For 16, 32, and 64 filter numbers, finer details of the provided image are extracted with the increasing number of filters.

Each residual block can perform three convolution operations with a kernel size of three. First, a convolution operation is carried out on the input. Then, it goes through the Batch Normalization and activation function. Then, a second convolution operation is performed with the result of the first one and a third convolution is carried out if the skip_conv variable is set to true and added with the result of the second convolution and sent through the activation function. The inclusion of skip connections within these blocks is instrumental in mitigating vanishing gradient issues, ensuring the smooth flow of information throughout the network. This is particularly important for OCR tasks, where preserving spatial information is vital.

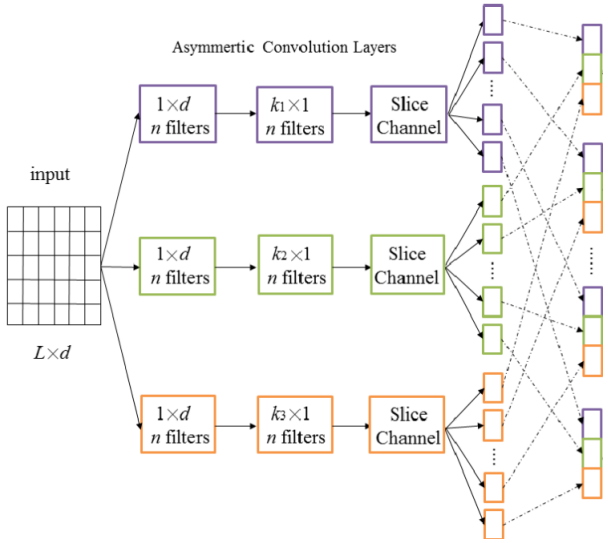


Fig. 4: First Part of the Fully Implemented Architecture [6]

The activation function remains consistent as leaky ReLU across all blocks, and a dropout rate of 0.2 is implemented to randomly deactivate neuron fractions during training, mitigating the risk of overfitting. After going through all nine residual blocks, the hierarchical features of the input image are extracted, which is then flattened or squeezed and sent through the BLSM.

The BLSTMs of 128 units process sequences bidirectionally, meaning they consider information from the forward and backward directions. They are used to separate each character sequence of a word. The return_sequences = true ensures that the complete sequence of output for each input is provided preserving the temporal aspect of the data. The result of sequences generated from the BLSTMs is used to generate a predictive result from existing recognized characters and an additional blank node in a Dense layer with softmax activation function.

These predictions are then compared to the ground truth label, and the resulting loss is calculated to update the model weights for the minimization of predicted words. To address alignment challenges, such as varying sizes or character repetitions between predicted words and ground truth labels, a blank node is introduced and the Connectionist Temporal Classification (CTC) loss considers all conceivable alignments between the predicted and ground truth sequences. The loss computation involves summing probabilities across valid alignments, incorporating the blank symbol, and accounting for repeated characters.

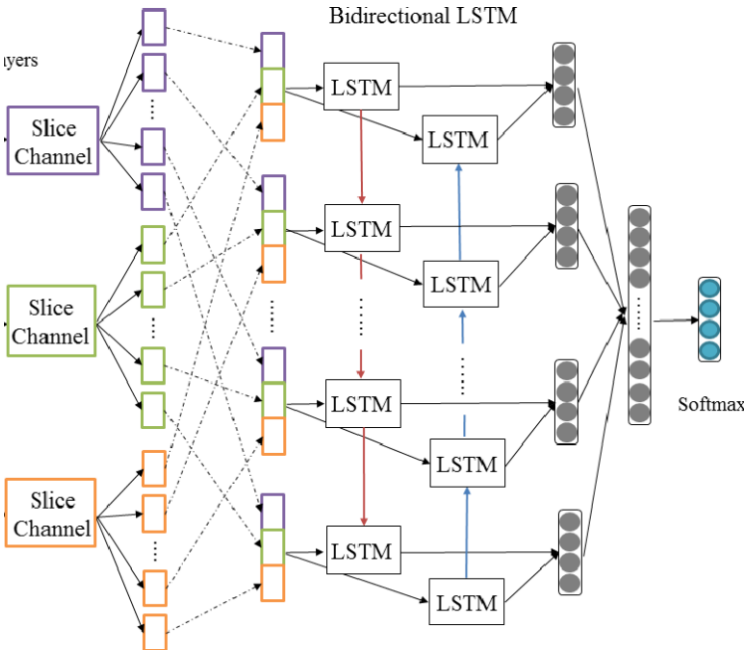


Fig. 5: Second Part of Fully Implemented Architecture [6]

Over several iterations, the model learns gradually to separate each alphabet of the word and predict them accurately, lowering the CER and eventually the WER calculated using custom functions. Thus the model learns to predict each individual letter of the provided word in the image. Eventually, the model becomes accurate enough to be able to predict text from Handwritten words which is completely outside of the dataset.

EXPERIMENTS AND RESULT ANALYSIS

Experimental Design

A handwriting recognition model was trained using the IAM Words dataset, and the experiment aimed to develop a model that can accurately recognize handwritten text. The dataset was preprocessed and a model architecture was designed using residual blocks and bidirectional LSTM layers.

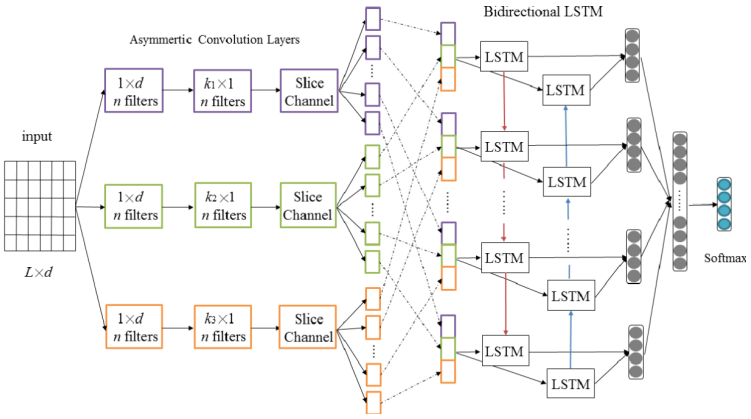


Fig. 6: Fully Implemented Architecture [6]

The focus of the experiments was to optimize the performance of the model by adjusting various parameters such as image size, learning rate, and training epochs.Data augmentation techniques such as random brightness, rotation, erosion/dilation, and sharpening were applied to improve the generalization ability of the model.

The training process used a connectionist temporal classification (CTC) loss function, which is suitable for sequence-to-sequence tasks such as handwriting recognition. Early stopping and model checkpointing callbacks were used to prevent overfitting and save the best-performing model during training.

Additionally, the experiment included a custom character word error rate (CWER) metric to evaluate the model's accuracy in recognizing handwritten words.

Execution of the Experiment

The experiment was run for 150 epochs with proper use of training and validation metrics. Key metrics tracked during training include CTC loss, character error rate (CER), and word error rate (WER).

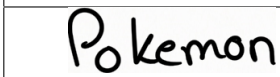
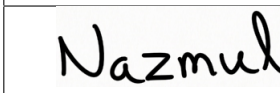
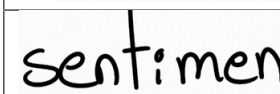
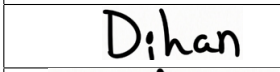
Data	Label	Prediction
	Pokemon	Pokemon
	Akash	Akash
	Nazmul	Nazmul
	sentiment	sentement
	Dihan	Dihan
	analysis	analyss

TABLE I: Model Predictions

To ensure proper generalization, model performance was evaluated on both training and validation sets. Throughout the training process, model checkpoints are now saved based on validation CERs to help select the best-performing model. The goal of the experiment is to find a balance between achieving a low CER on the validation set and preventing overfitting as well as avoiding biases or underfitting.

Model	WER	CER
Our Model	**18.54**	**6.95**
Dreuw et al. [7]	18.8	10.1
Boquera et al. [8]	15.5	6.9
Kozielski et al. [9]	13.3	5.10
Bluche et al. [10]	11.9	4.9
Doetsch et al. [2]	12.2	4.7
Voigtlaender et al. [11]	9.3	3.5
Poznanski and Wolf [12]	6.45	3.44

TABLE II: How our model fares with others

Reporting model accuracy and loss

The training log provided detailed information about the model's performance at each epoch. CTC loss, CER, and WER were reported for both training and validation sets.

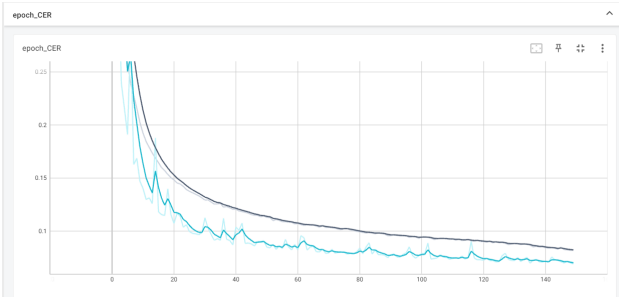


Fig. 7: CER vs Epoch (6.95)

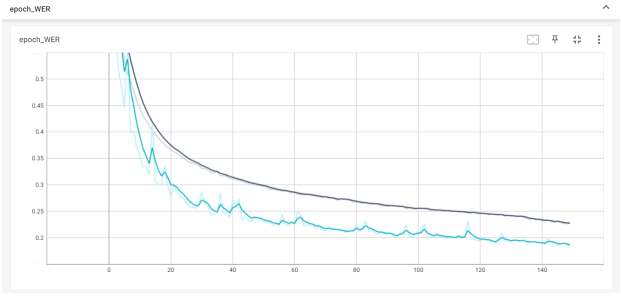


Fig. 8: WER vs Epoch (18.54)

The model shows a steady decrease in CTC loss and CER over epochs, indicating improved prediction accuracy for handwritten text. Validation CER was an important metric to assess the model's ability to generalize to unseen data. The model achieved consistently low validation CER values, reaching only 0.0695 in the final epoch. This demonstrated

the effectiveness of the model in accurately recognizing handwritten words. Overall the **Model's CER accuracy is 93.05% and WER accuracy is 81.46%**

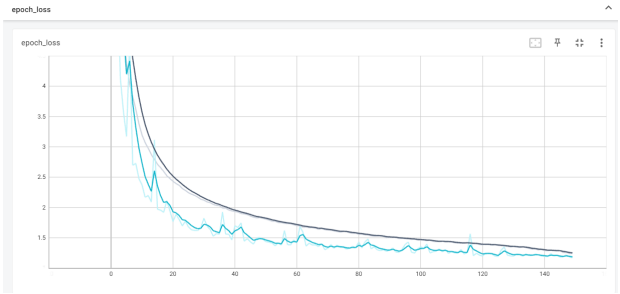


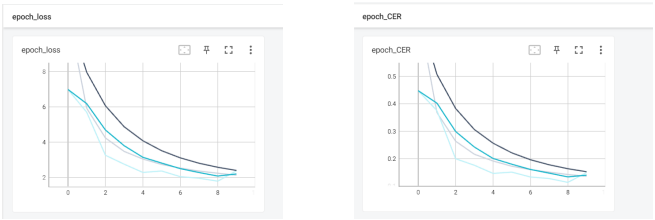
Fig. 9: Loss vs Epoch

Effect of Model Parameters

Experiments investigated the effect of various model parameters on performance. The chosen architecture, image size, and data augmentation techniques contributed to the model's ability to capture complex patterns in handwritten text. The learning rate and training epochs were adjusted to find the optimal balance between convergence and avoiding overfitting.

By using residual blocks and bidirectional LSTM layers, the model was able to capture both local and global dependencies in the input data. The effectiveness of these architectural decisions was reflected in a continuous decrease in CER and WER scores throughout training.

The CER, WER, Loss rate can change if we change different parameters of our model. If we tune down the regularization, i.e., decrease the amount of dropout in every residual blocks then the Loss vs Epoch graph and the CER vs Epoch graph would look something like this:



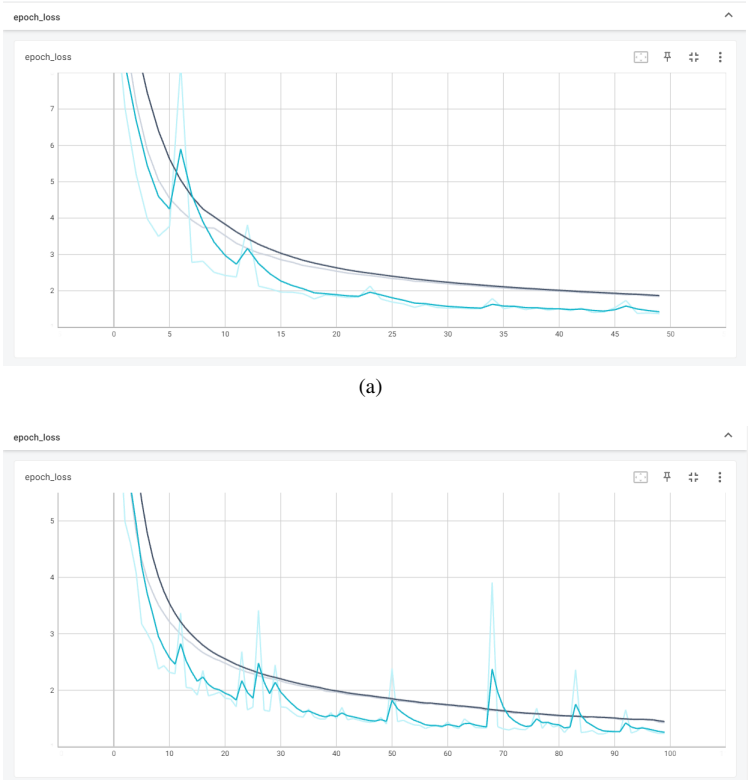
(a) (b)

Fig. 10: **(a)** Loss vs Epoch **(b)** CER vs Epoch

Now if we keep all of the parameters same but decrease the number of iterations (50 or 100 rather than 150) then we'll be able to find some dissimilarities in the graphs. What we can observe is that as the number of iteration increases the error decreases. The graphs are shown in Fig 11. We'll see that the validation set fluctuates frequently if we decrease the number of iterations.

Validity of Results

The obtained results are reflected in the training protocol and validation metrics to justify the effectiveness of the proposed handwriting recognition model. The low CER values in the validation set demonstrate the ability of the model to generalize well to previously unseen data, indicating potential for real-world applications.



(a)

(b)

Fig. 11: **(a)** Loss vs Epoch (50) **(b)** Loss vs Epoch (100)

Real Time Analysis

Our project is able perform real time handwriting recognition analysis. A user friendly and intuitive canvas was integrated into the project, offering a dynamic platform where users are able to write anything on the canvas for instant recognition.

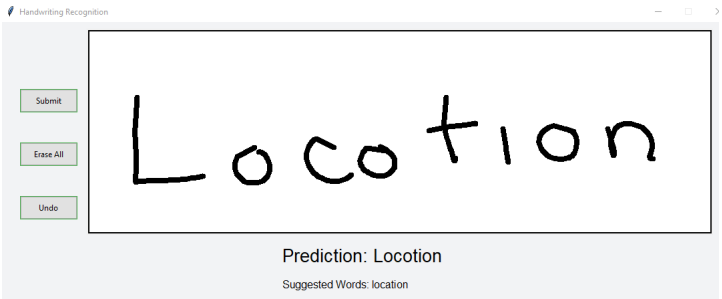


Fig. 12: Real-time Handwriting Recognition Analysis

The canvas also suggests words if the analyzed text contains any spelling mistakes or it doesn't match with any words from the dictionary. For example, here we wrote "Locotion," the

model correctly predicted ”Locotion,” but since the spelling is incorrect, it suggests potential corrections

This is how our project tries to go beyond basic recognition by incorporating a feature that suggests correct words if the predicted word is spelled incorrectly and if it isn't in the dictionary.

CONCLUSION

All in all, our handwriting recognition project grasps the power of convolutional neural networks, residual blocks, and bidirectional long-term and short-term memory layers to create robust models for interpreting various handwriting patterns. Careful experimental design, including data augmentation and monitoring of key metrics, parameters optimizes model performance, reflected in consistently low letter and word error rates during validation. Bridging the traditional and digital realms, this project's user-friendly interface demonstrates the model's real-time recognition capabilities and addresses the true value of handwritten documents.

REFERENCES

[1] Fully Convolutional Networks for Handwriting Recognition arXiv:1907.04888, 2019

[2] Patrick Doetsch, Michal Kozielski, and Hermann Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In 14th International Conference on Frontiers in Handwriting Recognition, pages 279–284, 2014.

[3] Alex Graves, Santiago Fern´andez, Faustino Gomez, and J¨urgen Schmidhuber. Connectionist temporal classification: labeling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning, pages 369–376. ACM, 2006.

[4] Sepp Hochreiter and J¨urgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997

[5] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint arXiv:1406.2227, 2014

[6] AC-BLSTM: Asymmetric Convolutional Bidirectional LSTM Networks for Text Classification Depeng Liang, Yongdong Zhang, 2016

[7] Philippe Dreuw, Patrick Doetsch, Christian Plahl, and Hermann Ney. Hierarchical hybrid mlp/hmm or rather mlp features for a discriminatively trained gaussian hmm: a comparison for offline handwriting recognition. In 18th IEEE International Conference on Image Processing, pages 3541–3544, 2011

[8] Salvador Espana-Boquera, Maria Jose Castro-Bleda, Jorge GorbeMoya, and Francisco Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. IEEE transactions on pattern analysis and machine intelligence, 33(4):767–779, 2011.

[9] Michał Kozielski, Patrick Doetsch, and Hermann Ney. Improvements in rwth's system for off-line handwriting recognition. In Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, pages 935–939. IEEE, 2013.

[10] Th´ eodore Bluche, Hermann Ney, and Christopher Kermorvant. A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. In International Conference on Statistical Language and Speech Processing, pages 199–210. Springer, 2014.

[11] Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In 15th International Conference on Frontiers in Handwriting Recognition, pages 228–233, 2016.

[12] Arik Poznanski and Lior Wolf. Cnn-n-gram for handwriting word recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2305–2314, 2016.

[13] Different Types of CNN Architectures Explained: Examplesby Ajitesh Kumar, 2023 : Link

[14] Bidirectional LSTM (BLSTM) structure diagram,ResearchGate : Link