Devanagari Handwritten Text Recognition

COSC 6342 Machine Learning
Spring 2024
Rohith Reddy Depa – 2295660
Ben Gideon Dokiburra - 2283917

Abstract

Recent advancements in handwritten text recognition (HTR) have primarily concentrated on common scripts, with limited studies addressing the unique challenges of Devanagari script. Our research explores the application of state-of-the-art HTR architectures to the IIIT-Devanagari dataset, focusing particularly on the CRNN architecture integrated with Fully Gated convolutional blocks. Among the architectures evaluated, the Flor model demonstrated promising results. This abstract presents our findings on the effectiveness of the Flor model, underlining its potential for practical implementation in Devanagari HTR systems. Additionally, we provide a detailed character-level error analysis, considering insertion, deletion, and substitution errors, to further refine and enhance the model's performance.

1 Introduction

The process of handwritten text recognition (HTR), especially for scripts as intricate as Devanagari, presents unique challenges due to the script's complexity and significant variations in handwriting styles among individuals. In our project, we focus on developing a robust system that can effectively recognize and digitize handwritten Devanagari text. Our model employs a Convolutional Recurrent Neural Network (CRNN) architecture that combines the spatial feature extraction capabilities of Convolutional Neural Networks (CNNs) with the sequential data processing power of Gated Recurrent Units (GRUs).

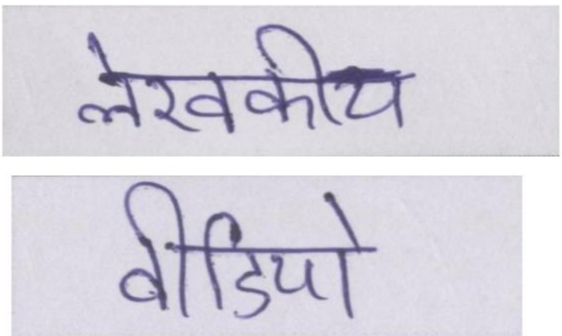
Our model is specifically tailored to address the complexities of the Devanagari script, which is used to write several major languages such as Hindi, Nepali, and Sanskrit. This script's extensive character set and the frequent occurrence of conjunct characters pose significant recognition challenges that our model manages efficiently through advanced machine learning techniques. By leveraging a comprehensive dataset of handwritten Devanagari text, our model undergoes rigorous training to optimize its accuracy and adaptability, ensuring effective performance across varied handwriting styles.

The ability of our model to reduce character error rates significantly and its computational efficiency make it a promising tool for applications requiring the digitization of historical manuscripts, educational materials, and administrative documents in Devanagari script. This not only aids in the preservation of cultural heritage but also enhances the accessibility of these documents in digital formats.

Through meticulous pre-processing of the input images, such as image enhancement and segmentation, our model is designed to handle inconsistencies in handwriting and physical conditions of the text such as smudges or faded ink. Additionally, our approach incorporates synthetic data generation to overcome the limitations posed by the scarcity of labeled data for Devanagari, further enhancing the model's training process and its ability to generalize across unseen text.

2 Dataset

The dataset consists of images in different files, train, val, test files contain the relative path to the image and its corresponding label, separated by a space. The hindi_vocab file contains the mapping b/w all the unique words present in the dataset and a vocab ID. The lexicon file contains the lexicon that was used while testing the test set. Inside the HindiSeg folder there are the train, val and test folders. Inside each of these 3 folders there are folders which specify the unique writer ID. Inside each of the unique writer ID folder, there are folders going from 1,2,3 ... x, where x is the number of pages that author wrote. Inside each of these folders there is a text file, which tells what the vocab ID for image "n.jpg" on the line number "n". From the hindi_vocab file we can get the actual label corresponding to that word.



An Image from the dataset

Training Data	69803
Validation Data	12713
Testing Data	10000

Top 5 substituted characters in Flor model

3 Data Preprocessing

Data preprocessing improves the quality of elements of the digital images called pixels.
Devanagari script has lot of loops and curves and preprocessing the data is very much important.
1) Binarization: Binarization means converting a colored image into an image which consists of only black and white pixels (Black pixel value=0 and White pixel value=255).
2) Skew Correction: While extracting the information from the scanned image, detecting & correcting the skew is crucial. There are different types of skew correction techniques one of them is Projection profile method.
3) Adaptive thresholding: Adaptive thresholding is a method used in image processing to distinguish between an image's foreground and background by establishing a threshold value. Adaptive thresholding is used in text image pre-processing to binarize text images, which is to say, turn them into black-and-white versions with the text in black and the backdrop in white. A fixed threshold value is applied to the whole picture when using traditional thresholding. However, in adaptive thresholding, a small region of local pixel values in and around each pixel are used to determine the threshold value. This method accounts for the differences in lighting and contrast present throughout the image, which may have an impact on the threshold value required to distinguish the foreground from the background precisely.

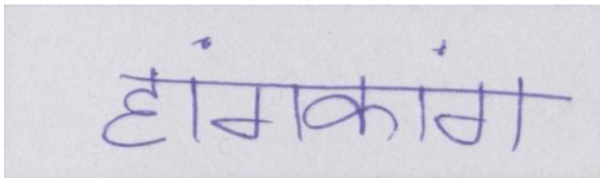


Image before Pre-processing

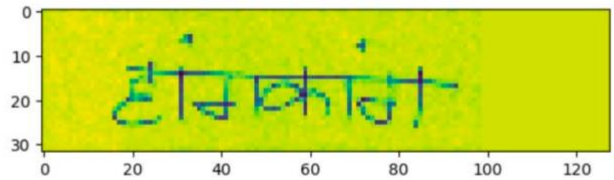


Image before adaptive threshold

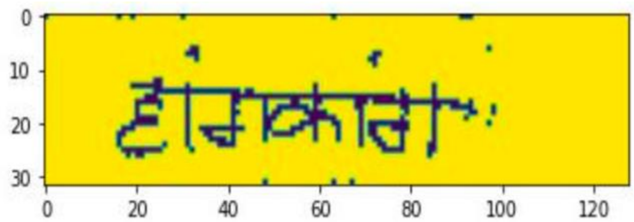


Image after Pre-processing

4 Literature Review

CRNN Model

The Convolutional Recurrent Neural Network (CRNN) integrates the spatial feature extraction capabilities of Convolutional Neural Networks (CNNs) with the temporal processing strengths of Recurrent Neural Networks (RNNs). This hybrid model is structured to include a CNN segment for initial image feature extraction, followed by an RNN segment to handle the sequential information derived from the CNN outputs. The sequence is finalized through a fully connected layer that classifies the textual data. CRNNs are particularly adept at managing variable-length input sequences and are capable of learning both spatial and temporal characteristics of data, making them well-suited for applications like handwriting recognition.

Bluche Model

The Bluche model presents a contemporary architecture designed specifically for handwriting recognition tasks. It uniquely combines a deep convolutional encoder with a bi-directional LSTM decoder. The encoder uses multiple convolutional layers to process input images and extract relevant features, which are then decoded by the LSTM layers that predict character sequences. This model emphasizes its ability to perform across different languages and scripts, leveraging transfer learning to enhance its adaptability and performance on diverse datasets.

Puigcerver Model

The Puigcerver model is noted for its robust performance in offline handwriting recognition, featuring a large set of parameters to boost its accuracy. It employs a modified CRNN framework, where the focus is shifted to a streamlined combination of one-dimensional convolutional and recurrent layers. The architecture includes convolutional blocks for spatial feature extraction and bidirectional LSTM layers for sequential processing. The output from these layers is directed to a linear layer that performs the final classification. This model's structure is optimized to improve recognition rates while maintaining efficient processing speeds.

HTR FLOR Model

Inspired by both the Bluche and Puigcerver models, the HTR FLOR model is designed to recognize text from scanned documents or images with high accuracy while maintaining a lower parameter count. This deep learning-based system utilizes a convolutional neural network for detailed feature extraction, followed by gated recurrent units (GRUs) for advanced sequence modeling. The model culminates in a fully connected layer that classifies the extracted features into textual outputs. The inclusion of bidirectional gated recurrent units (BGRUs) enhances its ability to capture long-term dependencies within the text, making it highly effective for processing complex and varied handwriting styles.

5 Approach

Over the years there is not much research done on the Devanagari handwritten text recognition. So, we are using the model proposed in HTR FLOR paper with the change in the filters used to extract features.

The photos in the dataset are binarized or changed from color to grayscale. After binarization, the pictures are subjected to noise reduction, which eliminates white dots in the dark area of the image and vice versa. Because every writer has a different writing style, every writer uses a different stroke width.

Now coming to the convolution block. We employ filters to identify the horizontal line, vertical line, loops, and curves in the first layer of the convolutional block. After the features are extracted, they are given to additional layers to extract various combinations, such as a vertical line, a loop on the left, and a curve on the right. The output of a max pooling layer is applied after all convolutional layers, and the output of this layer is applied to a recurrent block to predict the word. PreLU activation function, which yields the highest value of x, 0.001x, is employed in each convolution layer of the convolutional block.

The model's recurrent block is made up of two GRU layers and a dense layer after that. Finally, the SoftMax activation layer receives the output. The term "GRU" refers to a gated recurrent unit, which aims to exploit connections made across a series of nodes to carry out memory-related machine learning tasks.

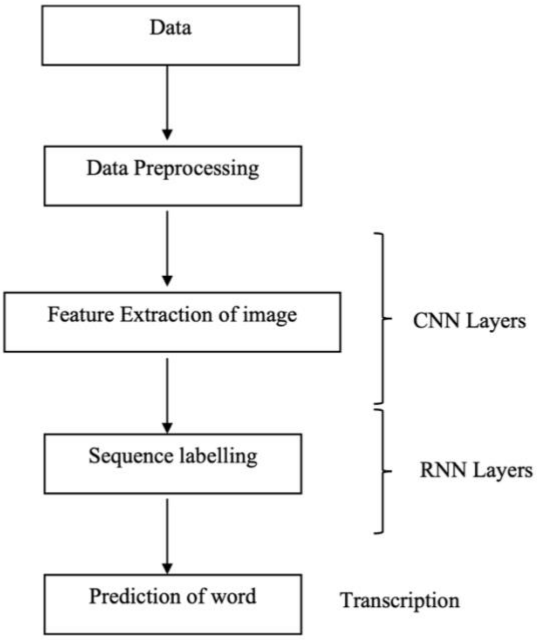
Filters for the feature extraction

The purpose of feature extraction is to invent feature vector which recognize the patterns. We extract the following features to detect a letter.

- Horizontal line
- Vertical line
- Loops
- curves

Since every word consists of horizontal line and most of the letters contain vertical lines they must be extracted. Every letter has a different combination of the above features so extracting those features help to predict the letter easily.

6 Methodology



Architecture of Devanagari text recognition model

In CRNN model, the component of convolutional layers is constructed by taking the convolutional and max-pooling layers from a standard CNN model (fully connected layers are removed). An input picture is utilized to extract a sequential feature representation using such a component. All the photographs must be resized to the same height before being supplied into the network. The input for the recurrent layers is then created by extracting a series of feature vectors from the feature maps generated by the convolutional layer component. On the feature maps by column, each feature vector in a feature sequence is created from left to right.

Convolutional Layers: The first step in feature extraction using CNN blocks is to apply a series of convolutional layers to the input data. The Hindi data is convolved using a series of learnable filters in each convolutional layer to create a collection of feature maps. These feature maps show which visual patterns are present in the input data.

Activation Function: After each convolutional layer, an activation function is applied to the output feature maps. Sigmoid and ReLU (rectified linear unit) are frequent activation functions. The activation function's goal is to provide non-linearity to the model so that it can understand more intricate links between the input data and the desired output.

Pooling Layers: In between the convolutional layers, pooling layers are often applied. The spatial dimensions of the feature maps are reduced by pooling layers, which combine nearby features into a single value. As a result, the model has fewer parameters and is better able to generalize to new data.

Dropout: Dropout is a regularization technique that is commonly applied to CNN models to prevent overfitting. Dropout randomly sets a fraction of the output of a layer to zero during training. This forces the model to learn more robust features that are not dependent on any single neuron in the previous layer.

Flatten: Once the feature maps have been generated by the convolutional and pooling layers,
the output is flattened into a 1D vector. This vector is then fed into a fully connected layer for further processing.

Fully Connected Layers: Fully connected layers are used to perform the final classification task. These layers receive the feature vectors that have been flattened as input and provide a vector of scores for each potential class. The anticipated output is then chosen as the class with the greatest score.
In the initial layers horizontal and vertical lines are extracted. while in the deeper layers curves and circles are detected. So, at last a complete feature of character is extracted.

B. Sequence labelling

The recurrent block contains 2 BGRU with dropout (probability 0.5) in the GRU cells alternated by a dense layer. A thick layer with a size equal to the charset size + 1 (the CTC blank symbol) exists in the model. The final sequence labelling operation is carried out by a group of fully connected layers after being fed the output feature sequence. A vector scores for each potential label is the output of each completely linked layer. The likelihood of each label is then calculated using these scores, and the label with the greatest probability is chosen as the anticipated result for that component of the input sequence.

C. Transcription

Transcription is the process of converting the per-frame predictions made by RNN into a label sequence. Mathematically, transcription is to find the label sequence with the highest probability conditioned on the per-frame predictions.

Probability of label sequence

We use the conditional probability defined in the Connectionist Temporal Classification (CTC) layer.

It is irrelevant where each label in label sequence l is situated because the probability is determined for label sequence l conditioned on the per-frame predictions y = y1,..., yT. Because of this, labelling the locations of individual letters is unnecessary when using the negative log-likelihood of this probability as the aim to train the network. Instead, we simply require pictures and their accompanying label sequences.

7 Implementation

The FLOR model is a sophisticated neural network designed specifically for the task of offline handwritten text recognition. It features a deep architecture that combines convolutional neural networks (CNNs) with bidirectional gated recurrent units (BGRUs), structured to efficiently process and recognize handwritten characters from images.

Convolutional Block

The convolutional block of the FLOR model is meticulously structured into six mini-blocks, each tailored to perform sequential transformations on input data to extract and refine features. Each mini block integrates standard and gated convolutions along with activation and normalization layers to enhance feature extraction capabilities:

<u>First Block:</u> Initiates with a 3x3 convolution, followed by Parametric ReLU (PReLU) for activation, batch normalization for stabilizing the learning process, a dropout with a probability of 0.2 to prevent overfitting, and concludes with a 3x3 gated convolution that introduces 16 feature maps. This block contains a total of 4,928 parameters.

<u>Second Block:</u> Consists of a similar structure with a 3x3 convolution and 32 feature maps, escalating the parameter count to 23,392.

<u>Third Block:</u> Adjusts the convolution to a 2x4 configuration and incorporates a dropout followed by a 3x3 gated convolution with 40 features, resulting in 39,480 parameters.

<u>Fourth Block:</u> Replicates the design of previous blocks but increases the features to 48, leading to 59,280 parameters.

<u>Fifth Block:</u> Features a 2x4 convolution and a final 3x3 gated convolution with 56 features, amounting to 78,568 parameters.

<u>Sixth Block:</u> Caps the convolutional series with a 3x3 convolution, employing 64 features and solidifying the feature extraction process with 32,832 parameters.

Recurrent Block

Following the extensive feature extraction by the convolutional layers, the recurrent block utilizes the power of bidirectional gated recurrent units (BGRUs) to analyze and interpret the sequential data extracted from the images:

First BGRU Layer: This layer uses 128 hidden units in a bidirectional setup, enhancing the model's ability to understand context from both forward and backward sequences, incorporating a dropout of 0.5 for robustness against overfitting, and consists of 625,920 parameters.

Second BGRU Layer: Mirrors the first in its configuration but increases the complexity and capacity, containing 846,447 parameters.

Overall, the FLOR model is engineered with a total of 1,710,847 trainable parameters, reflecting its capacity to handle complex patterns in handwritten texts. This architecture not only ensures high accuracy in text recognition but also provides a framework robust enough to handle variations in handwriting styles.

8 Evaluation Metrics

The error rate is calculated by considering 3 kinds of errors present in the transcribed text: Insertion error refers to the error that occurred because of wrongly inserted symbols or words in the transcribed text when compared to ground truth. Deletion error refers to errors caused by those symbols or words which are missing in the transcribed text when compared to the ground truth. Substitution error refers to the error induced by the model due to wrongly transcribing the symbols or words.

Character Error Rate (CER): it is calculated by adding the edit distance of each predicted text to original text and dividing it with total no of characters. CER specifies the minimum number of insertions, deletions, and substitutions of symbols that are required to convert the ground truth text to predicted text [30].

$$CER = (Sc + Dc + Ic)/Nc$$

where Sc indicates the number of characters to be substituted, Dc indicates the number of characters to be deleted and *Ic* indicates the number of characters to be inserted into the ground truth text for transforming it into predicted text. Nc indicates the total number of characters present in the ground truth text.

9 Results

The model is trained on training data on 40 epochs with batch size of 8 and achieved a value of 0.206149 for the CER.

Original character	Replaced character	No of times
न	ल	161
स	र	124
य	र	110
म	ग	109
म	स	79

Top 5 substituted characters in Flor model

10 Conclusion

The project explores the application of Convolutional Recurrent Neural Network (CRNN) models to recognize handwritten text in the Devanagari script, a challenging area due to its intricate character formations and significant stylistic variations among writers. The research focused on the Flor model, which demonstrated superior performance in our evaluations, achieving a character error rate (CER) of 0.206 and a word accuracy rate of 36.9%. This performance is notable for its high accuracy and was accomplished with fewer parameters, underscoring the Flor model's efficiency and effectiveness in processing and recognizing Devanagari script.

Furthermore, the project utilized a comprehensive approach to data preprocessing, which is critical in enhancing the quality of input images for better model training and performance. Techniques such as binarization, skew correction, and adaptive thresholding were employed to refine the input data, addressing common issues such as smudges and variations in ink density.

Overall, this research contributes significantly to the field of handwritten text recognition by providing insights into the capabilities of advanced CRNN models, particularly in handling complex scripts like Devanagari. The findings suggest promising directions for future enhancements and potential applications in digitizing historical manuscripts and other documents in Devanagari script, thereby aiding in the preservation and accessibility of cultural heritage.

11 References

[1] Arthur Flor de Sousa Neto, Byron Leite Dantas Bezerra, Alejandro Hector Toselli, Estanislau Baptista Lima, "HTR-Flor: A Deep Learning System for Offline Handwritten Text Recognition", In 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI) (pp. 54-61). IEEE.

[2] B. Shi, X. Bai, C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 11, pp. 2298-2304, 1 Nov. 2017.

[3] Joan Puigcerver, "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?", 2017 14th IAPR international conference on document analysis and recognition (ICDAR).

[4] Theodore Bluche, Ronaldo Messina, "Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition", 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR).

[5] Alex Graves, Santiago Fernández, Faustino Gomez, Jürgen Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks", Conference: Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006.

[6] Ralf C. Staudemeyer, Eric Rothstein Morris, "Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks", arXiv preprint arXiv:1909.09586 (2019).

[7] Sherstinsky, Alex, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network", Physica D: Nonlinear Phenomena.

[8] Keiron O'Shea, Ryan Nash, "An Introduction to Convolutional Neural Networks", 2015, Department of Computer Science, Aberystwyth University, Ceredigion, SY23 3DB.School of Computing and Communications, Lancaster University, Lancashire, LA1.

[9] Chung, Junyoung & Gulcehre, Caglar & Cho, KyungHyun & Bengio,"Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling" (2014).

[10] Yin, Wenpeng, Katharina Kann, Mo Yu, and Hinrich Schütze, "Comparative study of CNN and RNN for natural language processing", arXiv preprint arXiv:1702.01923 (2017).