DEVANAGARI HANDWRITTEN TEXT RECOGNITION

by

PARASA NAGA VENKATA PAVAN	***411961***
MUNJETI PRUDHVI SRAVAN	***411953***
R KRISHNA VAMSHI	***411967***

Under the guidance of

Dr. S. NAGESH BHATTU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH

TADEPALLIGUDEM - 534101, INDIA

MAY, 2023

DEVANAGARI HANDWRITTEN TEXT RECOGNITION

Thesis submitted to
National Institute of Technology Andhra Pradesh
for the award of the degree
of
Bachelor of Technology

by

PARASA NAGA VENKATA PAVAN	***411961***
MUNJETI PRUDHVI SRAVAN	***411953***
R KRISHNA VAMSHI	***411967***

Under the guidance of

Dr. S. NAGESH BHATTU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGYANDHRA PRADESH
TADEPALLIGUDEM-534102, INDIA

MAY, 2023

APPROVAL SHEET

This project work entitled **"DEVANAGARI HANDWRITTEN TEXT RECOGNITION"** by PARASA NAGA VENKATA PAVAN, MUNJETI PRUDHVI SRAVAN and R KRISHNA VAMSHI, is approved for the degree of Bachelor of Technology in the department of Computer Science and Engineering.

Examiners

_____

_____

_____

Supervisor

_____

Dr. S. Nagesh Bhattu
Assistant Professor
Dept. of Computer Science and Engineering
National Institute of Technology, Andhra Pradesh

Head of Department

_____

Dr. K. Himabindu
Assistant Professor & HOD
Dept. of Computer Science and Engineering
National Institute of Technology, Andhra Pradesh

Date: _____

Place: _____

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Parasa Naga Venkata Pavan	Munjeti Prudhvi Sravan	R Krishna Vamshi
Roll No: 411961)	Roll No: 411953	Roll No: 411967
Date: _____	Date: _____	Date: _____

CERTIFICATE

It is certified that the work contained in the thesis titled "**Devanagari handwritten text recognition**," by "**Parasa Naga Venkata Pavan**, bearing Roll No: 411961" "**Munjeti Prudhvi Sravan**, bearing Roll No: 411953" "**R Krishna Vamshi**, bearing Roll No: 411967" has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree*

Dr. S. Nagesh Bhattu
Dept. of Computer Science and engineering
N.I.T. Andhra Pradesh
May, 2023

ACKNOWLEDGEMENT

The success and outcome of this project required a lot of guidance and assistance from many people, and we are privileged to have got this all along with the completion of our project. Everything we have done is because of such guidance and help, and we will never hesitate to thank them.

We owe our sincere gratitude to our project guide **Dr. S. Nagesh Bhattu**, Department of Computer Science, National Institute of Technology, Andhra Pradesh, who took keen interest and guided us all along, till the completion of our project work by providing all the necessary information.

We are grateful and lucky enough to receive consistent motivation, support, and guidance from all the staff of the Computer Science Department. They have helped us to complete our project work successfully. We would also like to extend our sincere gratitude to all my friends for their timely support.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
CRNN	Convolutional Recurrent Neural Network
LSTM	Long Short-term Memory
GRU	Gated Recurrent Unit
HTR	Handwritten Text Recognition
CER	Character Error Rate
BiLSTM	Bidirectional Long Short-term Memory
CTC	Connectionist Temporal Classification
BGRU	Bidirectional Gated Recurrent Units

ABSTRACT

Even though various new architectures were developed for the HTR in recent years, there are very few works done in recognizing Devanagari handwritten data. We aim to deploy various state of the art HTR architectures to the Devanagari handwritten data by considering IIIT-Devanagari dataset. The deployed CRNN based architectures yield a CER of 0.206 ,0.238264and 0.407458 respectively for Flor[1], Puigcerver [2] and Bluche[3] model. The results conclude that the CRNN architectures with Fully Gated convolutional blocks produce fair results and can be deployed in implementation of Devanagari HTR. We also present a detailed error analysis for these 3 architectures at character level by considering insertion, deletion and substitution errors.

Chapter 1 provides introduction to the Devanagari handwritten text recognition. Chapter 2 includes the literature review. Chapter 3 describes the proposed approach that includes the methodology. Chapter 4 includes the experiments and implementation details. Chapter 5 provides the results and future scope. Chapter 6 describes the summary of the work and conclusions.

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 General

Handwritten text recognition, also known as handwriting recognition, is the process of automatically identifying and converting written text that has been written by hand into digital text. In the current digital era, this technique has grown in significance since it enables the digitalization of written items like old books, notes, and papers [12]. It may also be applied in a variety of ways, including as increasing the effectiveness of data entry activities, enhancing the accessibility of printed materials for those who are blind, and automating the filling out of forms [13].

Devanagari is an ancient script used to write many languages such as Hindi, Nepali, and Sanskrit It is becoming more and more crucial in today's digital age to digitize handwritten manuscripts for ease of access and preservation [11]. The automated recognition of handwritten text is known as handwritten text recognition (HTR). The aim of this Devanagari Handwritten Text Recognition project is to develop a system that can recognize and transcribe the handwritten Devanagari text accurately [11].

This technique may be used for a number of purposes, including digitizing old papers, enhancing data input efficiency, and improving textual materials' accessibility for those with visual impairments [14]. Machine learning methods will be used in the project to train the system using a sizable dataset of handwritten Devanagari text. The system will then be tested on a separate set of handwritten text to evaluate its performance [15].

The variety of handwriting styles is one of the main obstacles in this project. Different people have different writing styles, and even the same person's writing can change depending on things like the pen they are using, the paper they are writing on, how they are feeling physically, and more. Furthermore, it might be challenging for the system to correctly recognize the handwritten text due to noise and distortions in the images of the text [16].

To overcome these challenges, the system will need to be robust enough to handle variations in handwriting and produce accurate results [15]. This will involve pre-processing of the images of the handwritten text, such as image enhancement and segmentation to improve the quality of the input images [17].

The absence of labelled data is another issue. A significant quantity of labelled data must be gathered, yet doing so requires time and labor-intensive work. To get around this, false data may be generated using synthetic data generation techniques, which can then be used to train the model. Due to a dearth of publicly accessible data, less study has been done on Devanagari handwritten text recognition than has been done on the English language. We utilized the IIT-HW-DEV dataset for this study. The dataset contains just word-level pictures [19].

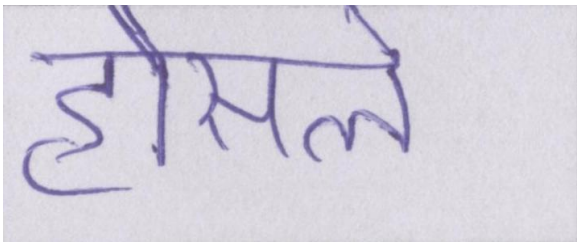

Fig.1.1 Devanagari handwritten text

The project will also involve the use of various machine learning techniques such as deep learning, specifically Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) which have been shown to be effective in image and sequence recognition tasks [7,8,20].

Overall, this project to recognize Devanagari handwritten text will be difficult but rewarding, and it has the potential to have a big impact on the preservation and accessibility of Devanagari script written works. A major increase in handwriting recognition systems' accuracy and dependability as a result of technological advancements has made it feasible to digitize a lot of written documents and make them more widely available [15].

1.2 Objective

The objective of this project is to develop a machine learning-based system that can accurately recognize and transcribe handwritten Devanagari text. The system will use computer vision techniques to preprocess the images and convolutional neural networks to recognize individual characters. A sequence model will be used to recognize the sequence of characters in a word. The developed system aims to address the lack of OCRs for Hindi language and improve the accessibility of written communication in Hindi [19].

CHAPTER-2

Literature Review

2.1. Convolutional Recurrent Neural Network (CRNN) based architecture

2.1.1 CRNN Model

Convolutional Recurrent Neural Networks (CRNNs) are a subclass of deep learning models that combine the advantages of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to handle image-based sequence recognition tasks like handwritten word recognition [1,7,8].

The CRNN model is composed of three main components: a convolutional neural network (CNN) for image feature extraction, a recurrent neural network (RNN) for sequence modeling, and a fully connected layer for classification [10]. In order for the RNN to model the sequential information of the text, the CNN is employed to extract features from the input image. RNNs can be implemented using a variety of unit types, including LSTM and GRU [4,6]. The final output is produced by passing the RNN's output via a fully linked layer [7].

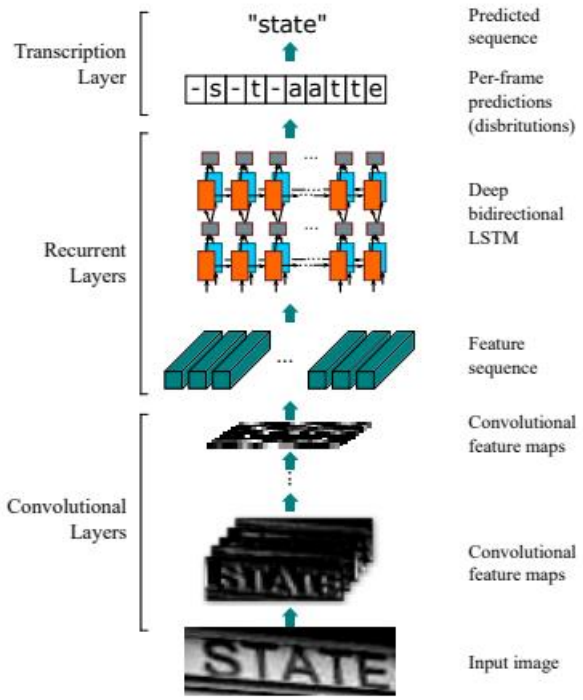

Fig. 2.1 CRNN model

The CNN part of the model is used to extract features from the image, it takes an image as input and applies a series of convolutional and pooling layers to extract features from the image [8,21,22]. These features are then passed to the RNN part of the model [7].

The RNN part of the model takes the features extracted by the CNN as input and models the sequential information of the text by processing the features in a recurrent manner. The output of the RNN is then passed through a fully connected layer for classification [1,2].

CRNN models have been shown to be effective in handwriting recognition tasks, achieving high accuracy and fast recognition speeds [2]. The CRNN model has the benefit of being able to handle input sequences of varying lengths and of learning both spatial and temporal characteristics from the input image, making it suitable for jobs like handwritten word recognition [1,2,3,4].

In conclusion, CRNN is an effective model for problems requiring image-based sequence recognition. A CNN for feature extraction, an RNN for sequence modelling, and a fully connected layer for classification make up its three primary parts [7,8]. The benefits of CNN and RNN are combined to create CRNN, which is well suited for applications like handwritten text recognition since it can model the sequential information of the text and extract features from the input image [1,2].

2.1.1.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is commonly used in computer vision tasks, such as image recognition, object detection, and segmentation. Due to their ability to automatically learn and extract features from images without needing to be explicitly coded, CNNs are extremely excellent at doing these tasks [8,10].

At a high level, a CNN consists of multiple layers, each of which performs a different type of computation on the input data. The three main types of layers in a typical CNN are:

1. Convolutional layers
2. Pooling layers
3. Fully-connected layers

2.1.1.2 Recurrent Neural Network (RNN)

The basic goal of RNNs is to give the network the ability to keep a hidden internal state that permits sequential input processing [7]. In other words, the prior inputs that were processed as well as the present state of the network are both influenced by the current input. RNNs are highly suited for a variety of real-world applications due to their capacity to handle sequential inputs [5].

An RNN is composed of a chain of repeating neural network blocks, or cells, where each cell receives input, modifies its internal state, and outputs data. The subsequent cell in the sequence receives each cell's output as an input. Up until all of the input data has been processed, this process continues [5,10].

2.1.2 Bluche Model

Bluche model suggests a novel neural network architecture for contemporary handwriting detection. The model is entirely built on a bi-directional LSTM decoder that predicts character sequences and a deep convolutional input image encoder that encodes input images. Its objective is to produce uniform, multilingual, and reusable tasks in this paradigm by utilizing more records for transfer learning [4,6].

The encoder in the Bluche model contains 3x3 Conv layer with 8 features, 2x4 Conv layer with 16 features, a 3x3 gated Conv layer, 3x3 Conv layer with 32 features, 3x3 gated Conv layer, 2x4 Conv layer with 64 features and 3x3 Conv layer with 128 features [4].

The decoder contains 2 bidirectional LSTM layers of 128 units and 128 dense layers between the LSTM layers [7]. Figure 5 shows the Bluche architecture

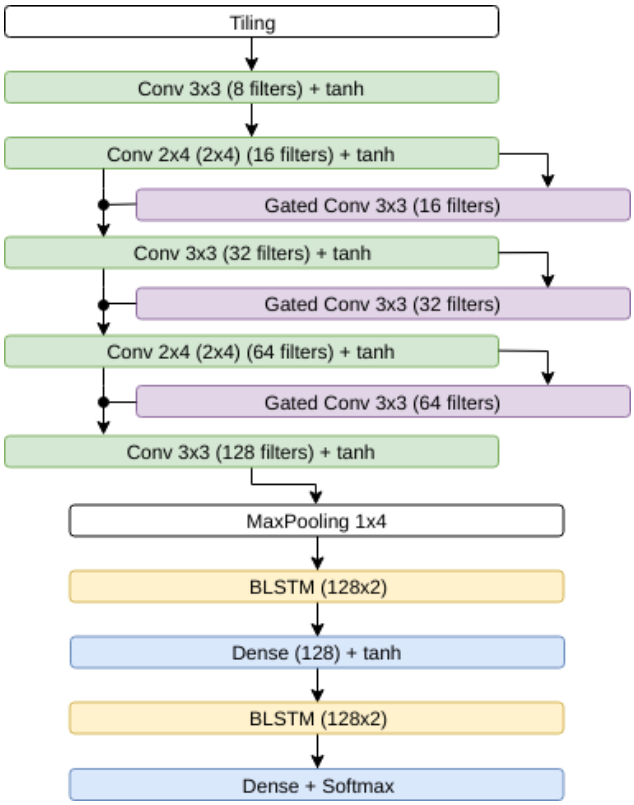

Fig. 2.2 Bluche HTR model

2.1.2.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM)'s primary characteristic is its capacity to selectively recall and forget information over extended periods of time, which makes it particularly useful for activities requiring the capture of long-term dependencies. This is accomplished through the LSTM architecture's usage of input, output, and forget gates in addition to memory cells [6,7].

The memory cell, which is the main part of the LSTM, is where the knowledge amassed over time is stored. While the output gate regulates information flow out of the memory cell, the input gate regulates the flow of fresh information into the memory cell. Which data should be removed from the memory cell is decided by the forget gate [6].

The LSTM's ability to selectively remember and forget information is what distinguishes it from other RNNs, which can suffer from the vanishing gradient problem, where the gradients become so small that the model fails to learn long-term dependencies. By selectively retaining important information, the LSTM is able to mitigate this problem and maintain its effectiveness even in long sequences [7].

2.1.3 Puigcerver Model

The Puigcerver model is a popular HTR model that has a high recognition rate and a large number of parameters, around 9.6 million [3]. Multidimensional LSTM networks are the foundation of contemporary Puigcerver model offline HTR methods [7,15]. Multidimensional recurrent layers, at least in the system's bottom levels, might not be enough, it has been suggested, to attain high recognition accuracy. Better results are obtained with an alternative model that just uses convolutional and one-dimensional recurrent layers [3,2]. Convolutional neural networks (CNN), one-dimensional recurrent neural networks (RNN), and a connectionist temporal classification (CTC) loss function are the three key components of the Puigcerver model [5,7,8].

- Convolutional blocks: they include 2-D Conv layer with 3x3 kernal size and 1 horizontal and vertical stride. number of filters is equal to 16n at the n-th layer of Convolution layer [8,21,23].
- Recurrent blocks: Bidirectional 1D-LSTM layers form re- current blocks, that transfer the input image column-wise from left to right and from right to left. The output of the two directions is concatenated depth-wise [6,7].
- Linear layer: the output of recurrent 1D-LSTM blocks is fed to linear layer to predict the output label. Dropout is implemented before the Linear layer to prevent overfitting (alsowith probability 0.5) [3].

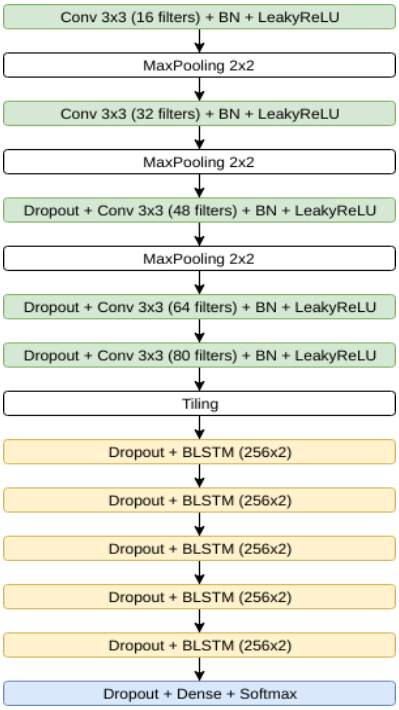

Fig. 2.3 Puigcerver HTR model

2.1.4 HTR FLOR Model

The offline handwritten text recognition system that is proposed in the HTR-Flor study is based on deep learning. This means that it can handle input sequences of varying length and recognise text from scanned documents or images [1,2].

The proposed model is inspired by Bluche architecture and Puigcerver architectures, aiming at: (i) to achieve better results than the Puigcerver model; and (ii) to keep a low number of parameters, such as the Bluche model [1,2,3,4].

The model is composed of three main components: a convolutional neural network (CNN) for feature extraction, a gated recurrent unit (GRU) for sequence modeling, and a fully connected layer for classification [8,24]. The CNN is used to extract features from the input image, which are then passed to the GRU to model the sequential information of the text [8]. The output of the GRU is then passed through a fully connected layer for classification [1,2,24].

A CRNN based architecture which was proposed in the HTR FLOR paper is used. The detailed work flow of the architecture is shown below.

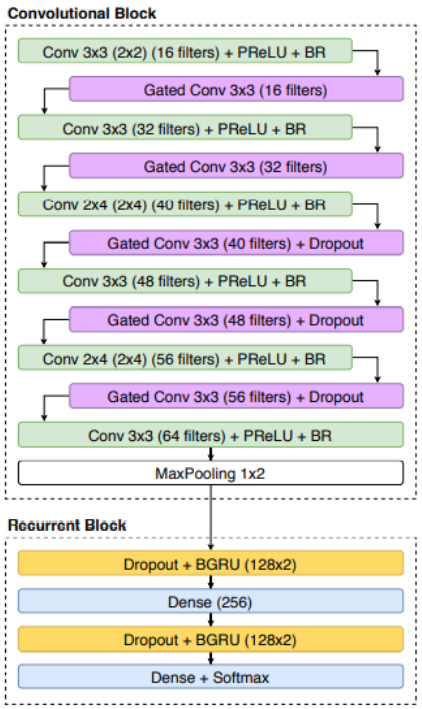

Fig. 2.4 HTR FLOR Model

The model's convolution block consists of six convolutional layers, with the last layer being followed by max pooling and every other layer being followed by a gated convolutional layer. In every layer, the PReLU activation function is employed [1,25]. The maximum results

from this activation function are x and 0.01*x. The required features are extracted from the preceding layer using the filters in each layer [8,20].

The recurrent block of the model consists of two GRU layers followed by dense layer. At last, the output is given to softmax activation layer. The GRU means gated recurrent unit that intends to use connections through a sequence of nodes to perform machine learning tasks associated with memory [24].

2.1.4.1 Bidirectional Gated Recurrent Units (BGRU)

Bidirectional Gated Recurrent Units (BGRU) is a variant of Recurrent Neural Networks (RNNs) that includes a "gating" mechanism to selectively allow or block information flow, improving the model's ability to capture long-term dependencies in sequences [26].

The "Bidirectional" aspect refers to the fact that BGRU uses two separate RNNs, one processing the input sequence from beginning to end (the "forward" RNN), and another processing the sequence in reverse (the "backward" RNN). The output of each RNN is concatenated and fed into a final layer that produces the final prediction [6,7,24,26].

2.2 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 [8,21,23].

- 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 [8,22,23].

CHAPTER-3

Proposed approach

Over the years there is no much research done on the Devanagari handwritten text recognition. So, we are using the model proposed in HTR FLOR paper (the detailed model is shown in figure 2.2) with the change in the filters used to extract features [1,2].

The photos in the dataset are binarized, or changed from colour 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 aforementioned 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 [25].

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 [24].

3.1 Dataset

The IIIT-HW-Dev dataset is a publicly available dataset of handwritten Devanagari text created by the Indian Institute of Information Technology, Hyderabad (IIIT-H). It contains over 92,000 images of handwritten Devanagari text in various forms, including printed and cursive text, as well as text written with different writing instruments and on different surfaces [19]. The dataset is divided into two parts: the training set and the test set. The training set contains over 63,000 images, while the test set contains over 29,000 images. The dataset includes text from various languages written in the Devanagari script, including Hindi and Sanskrit. 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 .

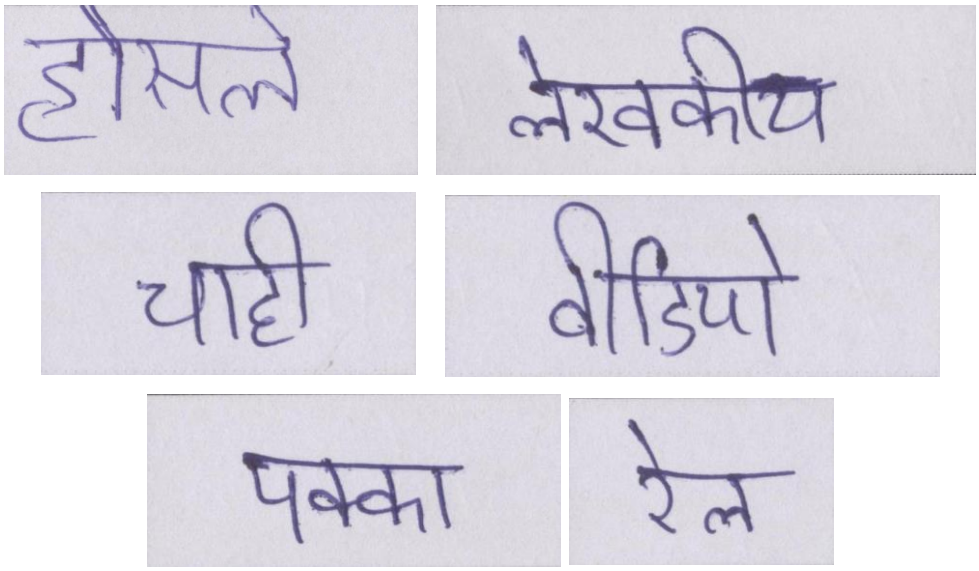

Fig. 3.1 Images from the dataset

3.2 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 [17,18].

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) [17].

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 [27].

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 [28].

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.

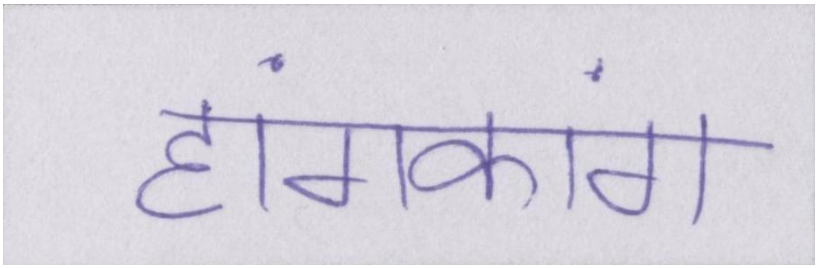

Fig. 3.2 Image before Pre-processing

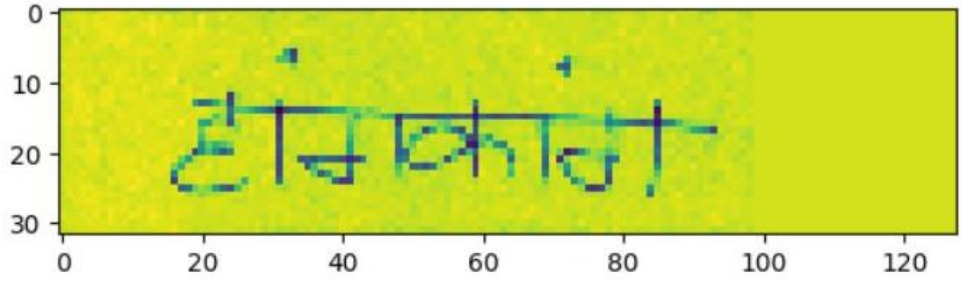

Fig.3.3 before adaptive threshold

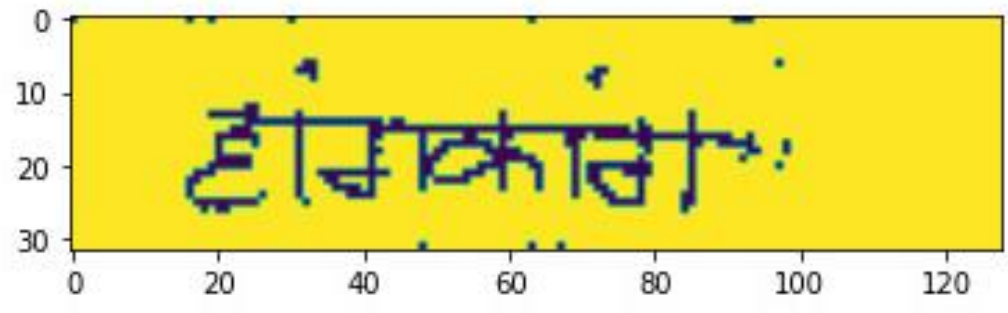

Fig.3.4 Image after Pre-processing

3.3 Methodology

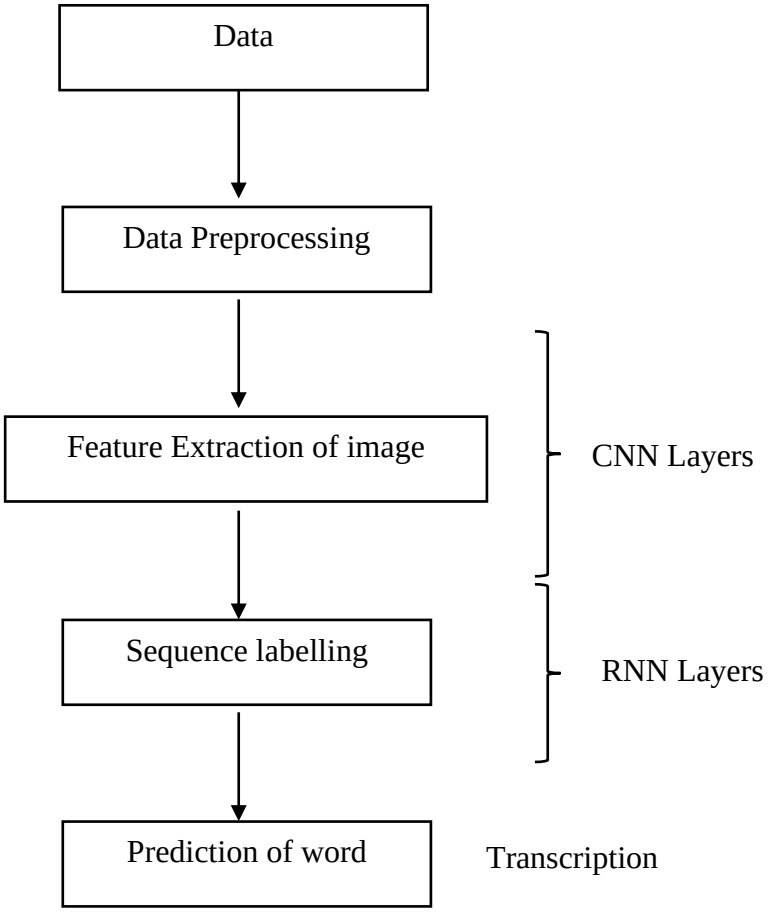

Fig.3.5 Architecture of Devanagari text recognition model

3.3.1 Feature Sequence Extraction

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) [8,22]. An input picture is utilized to extract a sequential feature representation using such a component. All of 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. In particular, 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 [25].

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 generalise to new data [21].

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 [1,2,8].

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 [22].

In the initial layers horizontal and vertical lines are extracted. while in the deeper layers curves and circles are detected. So, at last complete feature of character is extracted.

3.3.2 Sequence labelling

The recurrent block contains 2 BGRU with dropout (probability 0.5) in the GRU cells alternated by a dense layer [24,26]. A thick layer with a size equal to the charset size + 1 (the CTC blank symbol) exists in the model [5]. The final sequence labelling operation is carried out by a group of fully connected layers after being fed the output feature sequence. A vector of 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.

3.3.3 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 [1,2,3].

3.3.3.1 Probability of label sequence

We use the conditional probability defined in the Connectionist Temporal Classification (CTC) layer [5].

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.

CHAPTER-4

Experimental Procedure

4.1 Dataset Description

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.

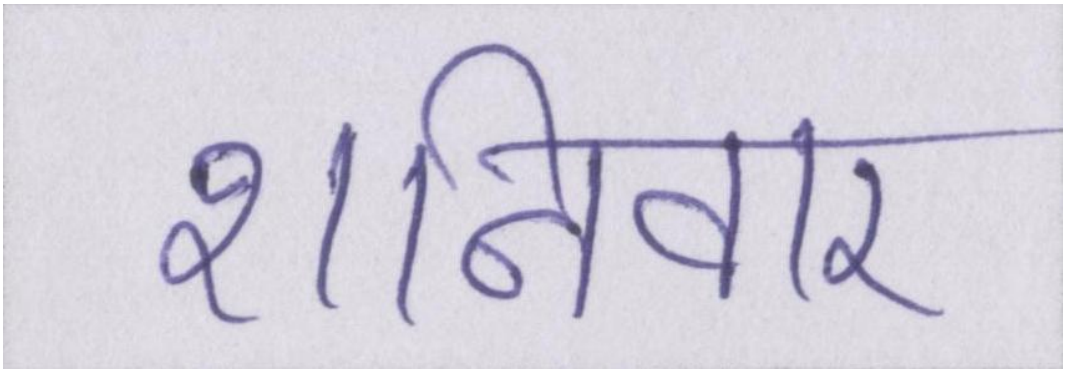

Fig.4.1 An image from the dataset

Table 4.1 Division of dataset

Training Data	69803
Validation Data	12713
Testing Data	10000

4.2 Implementation

Table 4.2 Detailed description of Flor model.

Model: " Flor model"

Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 128, 32, 1)]	0
conv2d (Conv2D)	(None, 64, 16, 16)	160
p_re_lu (PReLU)	(None, 64, 16, 16)	16
batch_normalization (BatchNormalization)	(None, 64, 16, 16)	112
full_gated_conv2d (FullGatedConv2D)	(None, 64, 16, 16)	4640
conv2d_1 (Conv2D)	(None, 64, 16, 32)	4640
p_re_lu_1 (PReLU)	(None, 64, 16, 32)	32
batch_normalization_1 (BatchNormalization)	(None, 64, 16, 32)	224
full_gated_conv2d_1 (FullGatedConv2D)	(None, 64, 16, 32)	18496
conv2d_2 (Conv2D)	(None, 32, 4, 40)	10280
p_re_lu_2 (PReLU)	(None, 32, 4, 40)	40
batch_normalization_2 (BatchNormalization)	(None, 32, 4, 40)	280
full_gated_conv2d_2 (FullGatedConv2D)	(None, 32, 4, 40)	28880

dropout (Dropout)	(None, 32, 4, 40)	0
conv2d_3 (Conv2D)	(None, 32, 4, 48)	17328
p_re_lu_3 (PReLU)	(None, 32, 4, 48)	48
batch_normalization_3 (BatchNormalization)	(None, 32, 4, 48)	336
full_gated_conv2d_3 (FullGatedConv2D)	(None, 32, 4, 48)	41568
dropout_1 (Dropout)	(None, 32, 4, 48)	0
conv2d_4 (Conv2D)	(None, 16, 1, 56)	21560
p_re_lu_4 (PReLU)	(None, 16, 1, 56)	56
batch_normalization_4 (BatchNormalization)	(None, 16, 1, 56)	392
full_gated_conv2d_4 (FullGatedConv2D)	(None, 16, 1, 56)	56560
dropout_2 (Dropout)	(None, 16, 1, 56)	0
conv2d_5 (Conv2D)	(None, 16, 1, 64)	32320
p_re_lu_5 (PReLU)	(None, 16, 1, 64)	64
batch_normalization_5 (BatchNormalization)	(None, 16, 1, 64)	448
reshape (Reshape)	(None, 16, 64)	0
bidirectional (Bidirectional)	(None, 16, 512)	494592
dense (Dense)	(None, 16, 256)	131328

bidirectional_1 (Bidirectional)	(None, 16, 512)	789504
dense_1 (Dense)	(None, 16, 111)	56943

Total params: 1,710,847
Trainable params: 1,709,567
Non-trainable params: 1,280

The convolutional block consists of mini-blocks with traditional and gated convolutions, so: (i) has a 3x3 convolution, Parametric Rectified Linear Unit (PReLU), batch normalization is applied, followed by dropout (probability 0.2) and a 3x3 gated convolution (16 features) [25]. This first block has total of 4928 parameters. (ii) a 3x3 convolution, PReLU, batch normalization is applied and a 3x3 gated convolution (32 features). This second block has total of 23,392 parameters. (iii) a 2x4 convolution, PReLU, batch normalization is applied and a 3x3 gated convolution (40 features) followed by dropout (probability 0.2) [25,29]. This third block has total of 39480 parameters. (iv) a 3x3 convolution, PReLU, batch normalization is applied and a 3x3 gated convolution (48 features) followed by dropout (probability 0.2). This 4th block has total of 59,280 parameters. (v) a 2x4 convolution, PReLU, batch normalization is applied and a 3x3 gated convolution (56 features) followed by dropout (probability 0.2). This fifth block has total of 78,568 and (vi) a 3x3 convolution (64 features), PReLU, batch normalization is applied. This layer has total of 32,832 parameters. The recurrent block contains 2 BGRU with dropout (probability 0.5) [26]. The number of hidden units in GRUs is set to 128.The first recurrent layer has a total of 625,920. The second recurrent layer has a total of 846,447. The total parameters in Flor model are 1,710,847.

Table 4.3 Detailed description of Bluche model.

Model: "Bluche model"

Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 128, 32, 1)]	0
reshape (Reshape)	(None, 64, 16, 4)	0
conv2d (Conv2D)	(None, 64, 16, 8)	296

conv2d_1 (Conv2D)	(None, 32, 4, 16)	1040
gated_conv2d (GatedConv2D)	(None, 32, 4, 16)	2320
conv2d_2 (Conv2D)	(None, 32, 4, 32)	4640
gated_conv2d_1 (GatedConv2D)	(None, 32, 4, 32)	9248
conv2d_3 (Conv2D)	(None, 16, 1, 64)	16448
gated_conv2d_2 (GatedConv2D)	(None, 16, 1, 64)	36928
conv2d_4 (Conv2D)	(None, 16, 1, 128)	73856
reshape_1 (Reshape)	(None, 16, 128)	0
bidirectional (Bidirectional)	(None, 16, 256)	263168
dense (Dense)	(None, 16, 128)	32896
bidirectional_1 (Bidirectional)	(None, 16, 256)	263168
dense_1 (Dense)	(None, 16, 111)	28527

Total params: 732,535
Trainable params: 732,535
Non-trainable params: 0

The convolutional block consists of mini-blocks with traditional and gated convolutions, so: (i) has a 3x3 convolution (8 features), Hyperbolic Tangent function (tanh). This first block has total of 296 parameters. (ii) a 2x4 convolution, tanh and a 3x3 gated convolution (16 features). This second block has total of 3,360 parameters. (iii) a 3x3 convolution, tanh and a 3x3 gated convolution (32 features). This third block has total of 13,888 parameters. (iv) a 2x4convolution, tanh and a 3x3 gated convolution (64 features). This 4th block has total of 53,376 parameters. (v) a 3x3 convolution and tanh is applied. This fifth block has total of 73,856 parameters. The recurrent block contains 2 BLSTM followed by dense layer (tanh activation). The first recurrent layer has a total of 296,064 parameters. The second recurrent layer has a total of 291,695 parameters. The total parameters in Bluche model are 732,535.

Table 4.4 Detailed description of Puigcerver model.

Model: " Puigcerver model"

Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 128, 32, 1)]	0
conv2d_5 (Conv2D)	(None, 128, 32, 16)	160
batch_normalization_5 (BatchNormalization)	(None, 128, 32, 16)	64
leaky_re_lu_5 (LeakyReLU)	(None, 128, 32, 16)	0
max_pooling2d_3 (MaxPooling2D)	(None, 64, 16, 16)	0
conv2d_6 (Conv2D)	(None, 64, 16, 32)	4640
batch_normalization_6 (BatchNormalization)	(None, 64, 16, 32)	128
leaky_re_lu_6 (LeakyReLU)	(None, 64, 16, 32)	0
max_pooling2d_4 (MaxPooling2D)	(None, 32, 8, 32)	0
dropout_4 (Dropout)	(None, 32, 8, 32)	0
conv2d_7 (Conv2D)	(None, 32, 8, 48)	13872
batch_normalization_7 (BatchNormalization)	(None, 32, 8, 48)	192
leaky_re_lu_7 (LeakyReLU)	(None, 32, 8, 48)	0

max_pooling2d_5 (MaxPooling2D)	(None, 16, 4, 48)	0
dropout_5 (Dropout)	(None, 16, 4, 48)	0
conv2d_8 (Conv2D)	(None, 16, 4, 64)	27712
batch_normalization_8 (BatchNormalization)	(None, 16, 4, 64)	256
leaky_re_lu_8 (LeakyReLU)	(None, 16, 4, 64)	0
dropout_6 (Dropout)	(None, 16, 4, 64)	0
conv2d_9 (Conv2D)	(None, 16, 4, 80)	46160
batch_normalization_9 (BatchNormalization)	(None, 16, 4, 80)	320
leaky_re_lu_9 (LeakyReLU)	(None, 16, 4, 80)	0
reshape_1 (Reshape)	(None, 16, 320)	0
bidirectional_5 (Bidirectional)	(None, 16, 512)	1181696
bidirectional_6 (Bidirectional)	(None, 16, 512)	1574912
bidirectional_7 (Bidirectional)	(None, 16, 512)	1574912
bidirectional_8 (Bidirectional)	(None, 16, 512)	1574912
bidirectional_9 (Bidirectional)	(None, 16, 512)	1574912
dropout_7 (Dropout)	(None, 16, 512)	0
dense_1 (Dense)	(None, 16, 111)	56943

Total params: 7,631,791

Trainable params: 7,631,311

Non-trainable params: 480

The architecture presented by Puigcerver uses a traditional CRNN approach, where it has a high level of recognition rate and many parameters. The workflow has the 5 convolutional and 5 BLSTM layers of the architecture. The first convolution layer has a 3x3 convolution (16 features), Leaky Rectifier Linear Units (LeakyReLU), batch normalization and MaxPooling with 2x2 kernel is applied. This first layer has a total of 224 parameters. The second convolution layer has a 3x3 convolution (32 features), Leaky Rectifier Linear Units (LeakyReLU), batch normalization and MaxPooling with 2x2 kernel is applied. This second layer has a total of 4,768 parameters. The third convolution layer has a 3x3 convolution (48 features), Leaky Rectifier Linear Units (LeakyReLU), batch normalization and MaxPooling with 2x2 kernel is applied. This third layer has a total of 14,064 parameters. The fourth convolution layer has a 3x3 convolution (64 features), Leaky Rectifier Linear Units (LeakyReLU), batch normalization and drop out is applied. This fourth layer has a total of 27,968 parameters. The fifth convolution layer has a 3x3 convolution (80 features), Leaky Rectifier Linear Units (LeakyReLU), batch normalization and drop out is applied. This fifth layer has a total of 46,480 parameters. The recurrent block consists of 5 BLSTM with dropout (probability 0.5) in the LSTM cells. The number of hidden units in all LSTMs is set to 256.The first recurrent block has total of 1,181,696 parameters. The second recurrent block has total of 1,574,912 parameters. The third recurrent block has total of 1,574,912 parameters. The fourth recurrent block has total of 1,574,912 parameters. The fifth recurrent block has total of 1,574,912 parameters. Finally, a dense layer is applied which as 56,943 parameters. Puigcerver model contains total of 7,631,791.

Initially a preprocessed image of 128*32 is provided to a CNN network, which extract features from the image by using filters followed by p-relu activation function. The extracted features are then given to the RNN network, which consists of 2 BGRU each followed by a dense layer. The size last dense layer is length of charset+1. The output of each fully connected layer is a vector of scores for each possible label. These scores are then used to compute the probability of each label, and the label with the highest probability is selected as the predicted output for that element in the input sequence. This output sequence is then given to ctc decode function, which gives the final output label vector.

4.2.2 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.

Word accuracy: no of words correctly predicted by total no of words.

4.3.1 Software specifications

Software specifications that are used in this project are briefly listed below:

1. Programming language used: Python3
2. Framework for deep learning: Tensorflow, keras, cv2
3. Visualization tools: Matplotlib
4. Other library and packages: numpy, pandas etc.

4.3.2 Hardware specifications

Table 4.5 Specifications

Component	Name
GPU	NVIDIA GeForce RTX 2080 TI 12 GB GPU
RAM	32 GB memory

Hardware specifications that are used in this project are briefly listed below:

1. GPUs are used as they are faster than CPU and much needed for this project as it involves image processing work.
2. RAM of decent size is used to store the training samples in batches and train our deep learning network architecture

CHAPTER-5

Results and Discussions

5.1 Results

Table 5.1 Metrics

	Training data	Testing data	Character Error Rate (CER)	Word accuracy
Flor	69803	10000	0.18801408	0.4160
Bluche	69803	10000	0.40745865	0.1143
Puigcerver	69803	10000	0.238264	0.3111

Each model is trained on training data on 40 epochs with batch size of 8 and got the above results. The results are showing that Flor model gives the highest accuracy with CER of 0.206149 as compared to Bluche and Puigcerver models with CER 0.40745865 and 0.238264 respectively

Table 5.2 Top 5 substituted characters in Flor model

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

Table 5.2 Top 5 substituted characters in Bluche model

Original character	Replaced character	No of times
न	ल	490
म	स	297
ा	्	166
त	ल	164
ो	ी	156

Table 5.2 Top 5 substituted characters in Puigcerver model

Original character	Replaced character	No of times
ल	त	134
व	ब	111
ब	व	103
स	म	101
ड़	ड	99

CHAPTER - 6

Summary and Conclusion

The project is to detect the Devanagari handwritten text using a CRNN model.

Chapter 2 discusses various works in the literature related to this project. Firstly, we discuss about the CRNN architecture which extracts the features from images and predict the word (character by character). Next, we discussed about the 3 CRNN based models namely Bluche, Puigcerver, Flor model architecture. Finally, we discuss about the filters for feature extraction.

Chapter 3 introduces about the proposed approach for Devanagari handwritten text recognition and the dataset used for training and testing. Next, we discussed about the data preprocessing used on the dataset and detailed methodology of the project.

Chapter 4 gives information about the implementation details and evaluation metrics used to evaluate the models. Chapter 5 contains all the results of all the models achieved on handwritten text recognition. The Flor model achieved a character error rate of 0.206 and word accuracy of 36.9 % which is better than Bluche model and Puigcerver model with CER 0.40745865 and 0.238264 respectively. The Flor model has less parameters compared to Puigcerver and has better accuracy than the Bluche and Puigcerver models.

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 Fern´andez, Faustino Gomez, J¨urgen 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).

[11] Panchal, Brijeshkumar & Chauhan, Gaurang & Panchal, Sandipkumar & Chaudhari, Urvashi. "An investigation on feature and text extraction from images using image recognition in Android. Materials Today: Proceedings",(2022).

[12] G.R. Hemanth, M. Jayasree, S. Keerthi Venii, P. Akshaya, and R. Saranya, "CNN-RNN BASED HANDWRITTEN TEXT RECOGNITION", ictact journal on soft computing, 2021.

[13] A. Sampath and N. Gomathi, "Handwritten Optical Character Recognition by Hybrid Neural Network Training Algorithm", The Imaging Science Journal, Vol. 67, No. 7, pp. 359-373, 2019.

[14] J. Sueiras, V. Ruiz, A. Sanchez and J.F. Velez, "Offline Continuous Handwriting Recognition using Sequence to Sequence Neural Networks", Neurocomputing, Vol. 289, pp. 119-128, 2018.

[15] Yuan, Aiquan, Gang Bai, Lijing Jiao, and Yajie Liu, "Offline handwritten English character recognition based on convolutional neural network", In 2012 10th IAPR International Workshop on Document Analysis Systems, pp. 125-129. IEEE, 2012.

[16] Ingle, R. Reeve, Yasuhisa Fujii, Thomas Deselaers, Jonathan Baccash, and Ashok C. Popat, "A scalable handwritten text recognition system", In 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 17-24. IEEE, 2019.

[17] Kumar, Gaurav, Pradeep Kumar Bhatia, and Indu Banger, "Analytical review of preprocessing techniques for offline handwritten character recognition", International Journal of Advances in Engineering Sciences 3, no. 3 (2013): 14-22.

[18] Kadhim, Ammar Ismael, "An evaluation of preprocessing techniques for text classification", International Journal of Computer Science and Information Security (IJCSIS) 16, no. 6 (2018): 22-32.

[19] Dutta, Kartik, Praveen Krishnan, Minesh Mathew and C. V. Jawahar, "Offline Handwriting Recognition on Devanagari Using a New Benchmark Dataset", 2018 13th IAPR International Workshop on Document Analysis Systems (DAS) (2018): 25-30.

[20] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks", in ECCV, 2014.

[21] Yu, Dingjun, Hanli Wang, Peiqiu Chen, and Zhihua Wei, "Mixed pooling for convolutional neural networks", In Rough Sets and Knowledge Technology: 9th International Conference, RSKT 2014, Shanghai, China, October 24-26, 2014, Proceedings 9, pp. 364-375. Springer International Publishing, 2014.

[22] Wiatowski, Thomas, and Helmut Bölcskei, "A mathematical theory of deep convolutional neural networks for feature extraction", IEEE Transactions on Information Theory 64, no. 3 (2017): 1845-1866.

[23] Chang, Shuo-Yiin, and Nelson Morgan, "Robust CNN-based speech recognition with Gabor filter kernels", In Fifteenth annual conference of the international speech communication association. 2014.

[24] Dey, Rahul, and Fathi M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks", In 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), pp. 1597-1600. IEEE, 2017.

[25] Dureja, Aman, and Payal Pahwa, "Analysis of non-linear activation functions for classification tasks using convolutional neural networks", Recent Patents on Computer Science 12, no. 3 (2019): 156-161.

[26] She, Daoming, and Minping Jia. "A BiGRU method for remaining useful life prediction of machinery." Measurement 167 (2021): 108277.

[27] Singh, Chandan, Nitin Bhatia, and Amandeep Kaur, "Hough transform based fast skew detection and accurate skew correction methods", Pattern Recognition 41, no. 12 (2008): 3528-3546.

[28] Roy, Payel, Saurab Dutta, Nilanjan Dey, Goutami Dey, Sayan Chakraborty, and Ruben Ray, "Adaptive thresholding: A comparative study", In 2014 International conference on control, Instrumentation, communication and Computational Technologies (ICCICCT), pp. 1182-1186. IEEE, 2014.

[29] Bjorck, Nils, Carla P. Gomes, Bart Selman, and Kilian Q. Weinberger. "Understanding batch normalization." Advances in neural information processing systems 31 (2018).

[30] Luce, Bryan R., Michael Drummond, Bengt Jönsson, Peter J. Neumann, J. Sanford Schwartz, U. W. E. Siebert, and Sean D. Sullivan, "EBM, HTA, and CER: clearing the confusion", The Milbank Quarterly 88, no. 2 (2010): 256-276.