# MAJOR PROJECT REPORT
# ON
# Nepali OCR

Submitted for the partial fulfillment of requirement of **Major Project** for Bachelors in Computer Engineering

**Submitted By**

Abish Bhusal(075BCT008)
Gopal Baidawar Chhetri(075BCT039)
Kiran Bhattarai(075BCT042)
Manjeet Pandey(075BCT048)

**Submitted to**

Department of Electronics and Computer Engineering
Institue of Engineering, Pulchowk Campus
Pulchowk, Lalitpur

**May 3, 2023**

MAJOR PROJECT REPORT

ON

# Nepali OCR

**Department of Electronics and Computer Engineering**

Institue of Engineering, Pulchowk Campus

Pulchowk, Lalitpur

**May 3, 2023**

## Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **”Nepali OCR”** submitted by **Abish Bhusal**, **Gopal Baidawar Chhetri**, **Kiran Bhattarai**, **Manjeet Pandey** in partial fulfillment of the requirements for the Bachelor’s degree in Electronics & Computer Engineering.

.............................
Supervisor
**Asst. Prof. Daya Sagar Baral**
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.............................
Internal examiner

Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.............................
External examiner

Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

# Copyright

# Acknowledgement

We, the authors of the Major Report on **"Nepali OCR"** , would like to express our heartfelt gratitude to the Department of Computer and Electronics Engineering of Central Campus, Pulchowk, for providing us with the opportunity to work on this minor project, which is an integral part of our final-year syllabus as per the course designed by IOE, Tribhuvan University.

We would like to thank our supervisors **Asst. Prof. Daya Sagar Baral** and **Mr. Anil Verma** for their unwavering encouragement, constant support, and valuable guidance, which have been instrumental in helping us successfully complete our project work. Their inputs and suggestions have immensely helped us in improving the quality of our work.

We would also like to extend our sincere thanks to **Rara Labs** for providing this opportunity to work on a real-world project as well for their assistance and cooperation throughout the duration of this project. Their contribution has been invaluable, and we are grateful for their support.

Lastly, we would like to express our appreciation to all the individuals who have helped us in one way or another, directly or indirectly, in completing this project. We are indebted to them for their assistance and support.

**Authors**
Abish Bhusal
Gopal Baidawar Chhetri
Kiran Bhattarai
Manjeet Pandey

# Abstract

The Nepali OCR project aims to develop a system that can detect and recognize handwritten Nepali texts. Our system's architecture consists of a Convolutional Neural Network (CNN) for feature extraction and a Recurrent Neural Network (RNN) for sequence recognition. The training dataset consists of around 80,000 Nepali handwritten images, which are preprocessed and augmented to increase the system's robustness. NepaliOCR can be used to convert printed or handwritten Nepali text into editable digital formats, making it useful for a range of applications such as document digitization, language learning, and natural language processing. This paper presents an overview of the NepaliOCR system, including its architecture, training methodology, and performance evaluation.

The lack of a reliable tool for Nepali handwriting recognition motivated us to develop this system. The system is developed as a part of the Bachelor in Computer Engineering Major Project. Machine learning has been used in the system to overcome the limitations of traditional computer systems. Our attempt to develop a tool for Nepali Handwriting Recognition using Machine Learning is discussed in this report. With the recent advancement in machine learning, our system shows great potential for practical applications in the digitization of Nepali handwriting.
**Keywords**: OCR, CRNN, Preprocessing, Data Collection, Augmentation, CNN, RNN

# Contents

**1 Introduction** **1**

1.1 Problem Statement . . . 1

1.2 Background . . . 1

1.3 Objectives . . . 2

1.4 Scope of the Project . . . 2

1.5 Application of the Project . . . 3

**2 Literature Review** **4**

2.1 Related work in traditional text recognition . . . 4

2.2 Related work in modern text recognition using RNNs . . . 5

2.3 Existing Systems . . . 6

**3 Theoretical Background** **7**

3.1 Optical Character Recognition (OCR) . . . 7

3.2 Convolutional Recurrent Neural Network . . . 8

**4 Methodology** **16**

4.1 System Design . . . 16

4.2 Data Collection . . . 18

4.2.1 Data Cleaning . . . 18

4.2.2 Data Annotation . . . 19

4.2.3 Data Augmentation . . . 19

4.3 Image Preprocessing . . . 23

4.3.1 Sharpening . . . 23

4.3.2 GrayScale . . . 24

4.3.3 Binarization . . . 25

4.3.4 Skewness Correction . . . 25

4.4 Training the Model . . . 26

4.4.1 Training . . . 26

4.5 Software Requirements . . . 31

**5 Result and Discussion** **34**

5.1 Result . . . 34

5.2 Discussion . . . 35

5.2.1 RecMetric . . . 36

5.2.2 Character Error Rate . . . . . . . . . . . . . . . . . . . . . . . . . 37
5.3 Comparison with existing systems i.e I2OCR . . . . . . . . . . . . . . . . . . 39

**6 Epilogue** **41**
6.1 Project Schedule . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
6.2 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
6.3 Limitations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.4 Future Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . . 42

**7 References** **43**

**8 Appendix** **45**
8.1 Appendix A . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
8.2 Appendix B . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
8.3 Appendix C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

# List of Figures

1 The network architecture . . . . . . . . 9
2 The receptive field . . . . . . . . 10
3 The structure of a basic LSTM unit . . . . . . . . 11
4 The structure of bi-directional LSTM used in the model . . . . . . . . 13
5 System Block Diagram . . . . . . . . 17
6 Original image . . . . . . . . 22
7 Deformed image . . . . . . . . 22
8 Rotated image . . . . . . . . 22
9 Translated image . . . . . . . . 22
10 Zoomed image . . . . . . . . 22
11 Steps of Image Preprocessing . . . . . . . . 23
12 Original Image . . . . . . . . 23
13 Sharpened Image . . . . . . . . 23
14 GrayScaled Image . . . . . . . . 24
15 Binarized Image . . . . . . . . 25
16 Skewed Image . . . . . . . . 26
17 MobilenetV3 . . . . . . . . 27
18 Output image . . . . . . . . 34
19 Training loss over global step . . . . . . . . 36
20 Training accuracy over global step . . . . . . . . 37
21 NepaliOCR vs i2OCR . . . . . . . . 39
22 NepaliOCR vs i2OCR . . . . . . . . 40
23 Gantt Chart . . . . . . . . 41
24 Word Images . . . . . . . . 45
25 MultiLine Images . . . . . . . . 45
26 Front Page . . . . . . . . 45
27 Text Generator Page . . . . . . . . 46
28 Annotation Page . . . . . . . . 46
29 Before OCR Page . . . . . . . . 47
30 After OCR Images . . . . . . . . 47
31 Output 1 . . . . . . . . 48
32 Output 2 . . . . . . . . 48

# List of Abbreviations

| | |
|---|---|
| **AI** | Artificial Intelligence |
| **CER** | Character Error Rate |
| **CNN** | Convolutional Neural Networks |
| **GT** | Ground Truth |
| **HMM** | Hidden Markov Models |
| **HP** | Hewlett Packard |
| **I/O** | Input/Output |
| **LSTM** | Long Short Term Memory |
| **ML** | Machine Learning |
| **OCR** | Optical Character Recognition |
| **OpenCV** | Open Source Computer Vision Library |
| **PNG** | Portable Network Graphics |
| **RGB** | Red Green Blue |
| **SPN** | Spatial Transformer Network |
| **TIFF** | Tagged Image File Format |
| **UML** | Unified Modeling Language |
| **UTF** | Unicode Transformation Format |
| OCR | Optical Character Recognition |

# 1 Introduction

## 1.1 Problem Statement

Nepali is the national language of Nepal and is spoken by millions of people worldwide. However, there is a lack of reliable tools for recognizing handwritten Nepali texts. The digitization of Nepali handwriting can have a significant impact on various domains such as banking, education, and government records. Currently, Google's Tesseract OCR is the most widely used OCR tool for Nepali text recognition. However, Tesseract OCR has limitations in recognizing handwritten Nepali texts accurately.

The lack of a reliable tool for Nepali handwriting recognition motivated us to develop a system that can recognize handwritten Nepali texts accurately. The objective of this project is to develop a system that can detect and recognize Nepali handwriting with high accuracy. The system's accuracy and robustness make it ideal for digitizing Nepali handwriting in various industries such as banking, education, and government records. Therefore, the problem statement of this major project is to develop a Nepali OCR system that can accurately recognize handwritten Nepali texts and can be applied in various practical domains.

## 1.2 Background

The background of the project highlights the lack of reliable tools for recognizing handwritten Nepali text and the potential benefits of having a reliable tool for Nepali handwriting recognition in various domains such as banking, education, and government records. Google's Tesseract OCR is currently the most widely used OCR tool for Nepali text recognition, but it has limitations in recognizing handwritten Nepali text accurately. Therefore, the project aims to develop a Nepali OCR system that can accurately recognize handwritten Nepali text using machine learning. The system uses a deep learning framework, and consists of a CNN for feature extraction and an RNN for sequence recognition. The system was trained using a dataset of around 80,000 Nepali handwritten character images, preprocessed, and augmented to increase the system's robustness.

## 1.3 Objectives

The primary objectives of our Nepali OCR project are as follows:

- Develop a reliable and accurate Nepali OCR system to recognize handwritten Nepali text.
- Utilize machine learning techniques, such as deep learning, to train the system and improve its accuracy.
- Create a user-friendly system that can be used by individuals and organizations to digitize handwritten Nepali documents and records.
- Promote digitalization in Nepal by bridging the gap between the digitally advanced world and Nepal in terms of language recognition and digitalization.
- Contribute to the academic and technological advancements in the field of OCR and machine learning.

## 1.4 Scope of the Project

The scope of the Nepali OCR project is quite broad and extensive. The project aims to develop a system that can recognize and digitize handwritten Nepali text. This can have numerous applications and benefits, such as:

- Digitizing historical and cultural documents written in Nepali script, which can aid in preserving Nepal's cultural heritage.
- Making it easier to store, search, and retrieve handwritten Nepali documents, such as legal documents, academic records, and medical records.
- Enabling the use of Nepali language in digital platforms and tools, such as text editors, search engines, and chatbots.
- Helping individuals and organizations to digitize their handwritten Nepali documents and records, which can save time and resources.
- Promoting digitalization in Nepal and contributing to the country's technological advancements.

Moreover, the use of machine learning techniques, such as deep learning, can improve the accuracy of the OCR system over time. This can lead to the development of more sophisticated systems that can recognize handwriting from various sources and contexts.

## 1.5 Application of the Project

The Nepali OCR project has a wide range of potential applications in various fields, including education, healthcare, legal, and cultural sectors. Some of the potential applications of the Nepali OCR project are:

- **Education:**
  The project can be used to digitize academic records, such as transcripts, certificates, and degrees, making it easier to store, search, and retrieve them. Additionally, the project can aid in developing digital libraries of Nepali literature, facilitating access to educational resources for students and researchers.
- **Healthcare:**
  The project can aid in digitizing medical records, such as patient histories, prescriptions, and test results. This can enable healthcare providers to retrieve patient information quickly and accurately, leading to better patient care.
- **Legal:**
  The project can be used to digitize legal documents, such as contracts, agreements, and deeds. This can make it easier to store and retrieve legal documents, reducing the time and resources required for legal procedures.
- **Cultural:**
  The project can aid in preserving Nepal's cultural heritage by digitizing historical documents, manuscripts, and inscriptions written in Nepali script. This can facilitate access to Nepal's cultural heritage for researchers and historians.
- **Business:**
  The project can be used by businesses to digitize their handwritten Nepali documents, such as invoices, receipts, and bills, making it easier to store and retrieve financial records.

# 2 Literature Review

## 2.1 Related work in traditional text recognition

The field of character recognition, which involves recognizing and classifying characters from images or scanned documents, has a rich history of research and development. In this literature review, we will highlight some of the most important papers in the field of character recognition.

- ”Pattern Classification and Scene Analysis” [13]: This classic text introduced many of the foundational concepts of pattern recognition, including the k-nearest neighbor algorithm, decision trees, and Bayesian classifiers. It remains a standard reference in the field of character recognition.
- ”Handwritten Character Recognition Using Neural Networks”[14]: This paper introduced the use of convolutional neural networks (CNNs) for handwritten character recognition, which is now a common approach used in many OCR systems. The paper presented a neural network architecture called the LeNet-5 that achieved state-of-the-art performance on the MNIST dataset.
- ”A Survey of Modern Optical Character Recognition Techniques”: This paper provided an overview of the most important techniques for OCR, including template matching, feature extraction, and classification. The paper highlighted the importance of combining multiple techniques to achieve high accuracy in OCR systems.
- ”Tesseract: A High-Performance OCR Engine” [15]: Tesseract is an open-source OCR engine that has become one of the most widely used OCR engines today. This paper described the Tesseract OCR engine and its architecture, including the use of adaptive recognition techniques and language models.
- ”Improving the Accuracy of Handwritten Character Recognition with a Split-and-Merge Approach” [16]: This paper introduced a split-and-merge approach for handwritten character recognition, which involves segmenting characters into smaller subcomponents and then reassembling them into the original characters. The approach achieved high accuracy on several benchmark datasets.
- ”End-to-End Text Recognition with Convolutional Neural Networks”[17]: This paper introduced an end-to-end approach for text recognition that combines a CNN for fea-

ture extraction with a recurrent neural network (RNN) for sequence recognition. The approach achieved state-of-the-art performance on several benchmark datasets.

- ”Robust Text Detection in Natural Images with Edge-Enhanced Maximally Stable Extremal Regions” [21]: This paper introduced an approach for text detection in natural scenes that uses maximally stable extremal regions (MSERs) and edge enhancement techniques. The approach achieved high accuracy on several benchmark datasets and has since become a popular approach in the field of scene text recognition.

These papers represent just a small sample of the many important contributions to the field of character recognition. Together, they demonstrate the importance of combining multiple techniques and approaches to achieve high accuracy in OCR systems.

## 2.2 Related work in modern text recognition using RNNs

Character recognition is a challenging task in the field of optical character recognition (OCR) that has been extensively researched over the past few decades. In recent years, deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have shown promising results in various OCR tasks, including character recognition. In this literature review, we will focus on papers related to CRNNs and sequence-based character recognition.

One important paper in this field is ”An Overview of Convolutional Neural Networks for Speech and Image Processing” by El Abbadi et al. (2019), which provides a comprehensive overview of CNNs and their applications in character recognition. The paper also discusses the use of RNNs for sequence-based recognition tasks. Another relevant paper is ”A Comprehensive Survey of Deep Learning for Image and Speech Processing” by Boussaid and Chetouani (2019), which provides a survey of deep learning techniques in OCR and handwriting recognition, including the use of CRNNs for sequence-based recognition.

In addition, ”A Survey on Recent Advances in Optical Character Recognition” by Manjunatha and Shivaprakash (2018) provides a survey of recent advances in OCR, including the use of deep learning techniques such as CNNs and RNNs. The paper also discusses the use of CRNNs for sequence-based recognition tasks. Similarly, ”A Survey of Convolutional Neural Networks for Character Recognition” by Alsheikh and Al-Ayyoub (2018) provides a survey of CNNs for character recognition, including their applications in OCR and handwriting recognition, as well as the use of CRNNs for sequence-based recognition tasks.

Finally, ”A Survey of Handwritten Character Recognition with MNIST and EMNIST” by Garsia and Igolnikov (2020) provides a survey of handwritten character recognition using the MNIST and EMNIST datasets, including the use of CNNs and RNNs for OCR tasks. The paper also discusses the use of CRNNs for sequence-based recognition tasks.

These papers demonstrate the importance of using deep learning techniques, particularly CRNNs, for sequence-based character recognition. By integrating these techniques into our CRNN-based OCR project, we aim to achieve high accuracy in recognizing and classifying characters from scanned documents and handwritten texts.

## 2.3 Existing Systems

Most existing systems for digitizing Nepali handwritten text are primarily based on pre-trained models like Tesseract OCR Engine, which are designed to recognize printed text rather than handwriting. These systems generally perform well on printed text but struggle with handwriting due to the variability and diversity in handwriting styles. A lot of currently available free online OCR tools are based on the Tesseract OCR Engine. Some such tools include i2ocr.com and easyocrconverter.com

Another prominent OCR system that is capable of recognizing handwritten and font texts is Google Cloud Vision OCR.Google Cloud Vision OCR is a cloud-based OCR service provided by Google that enables text recognition from images and documents in over 50 languages. The OCR service is available through a REST API that can be easily integrated into various applications.

Although these existing systems have contributed to the field of Nepali OCR, they have their limitations, including accuracy and usability issues. Therefore, the need for an improved and more accurate Nepali OCR system is still prevalent.

# 3 Theoretical Background

## 3.1 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a technology that recognizes and converts text into a machine-readable format by analyzing and comprehending its underlying patterns. It can identify both handwritten and printed text, as well as text that appears in real-life settings. In essence, OCR allows computers to read text. Deep learning and computer vision techniques are used to power OCR, with algorithms identifying text features and predicting corresponding outputs using neural networks. OCR can produce highly accurate outputs in a matter of milliseconds.

OCR is one of the earliest computer vision and deep learning problems that have undergone significant development. It is employed in various fields, including research and development, industrial applications, and personal use. Let us examine some practical applications and uses of OCR.

OCR is a highly versatile technology that has a wide range of applications due to its outstanding performance and ability to provide various solutions. One of the most significant areas where OCR can be utilized is information retrieval. By applying OCR to documents, receipts, and ID cards, it can convert the information into machine-readable text, making documents searchable and editable. Additionally, OCR can extract crucial information from documents and store it digitally. Another field where OCR is heavily used is Automatic License Plate Recognition (ALPR). ALPR relies on OCR to extract license plate numbers from vehicles, and it has become a core component in many places, such as offices, law enforcement agencies, and malls.

## 3.2 Convolutional Recurrent Neural Network

CRNN, or Convolutional Recurrent Neural Network, is a deep learning architecture that combines convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to process sequential data with variable-length inputs, such as text or images. The CNN component of CRNN is responsible for extracting spatial features from the input data, while the RNN component models the temporal dependencies between these features.

CRNN has been widely used for Optical Character Recognition (OCR), a task that involves recognizing text in images or scanned documents. OCR is a challenging task due to the variability of text appearance, such as different fonts, sizes, and orientations. CRNN is well-suited for OCR because it can handle variable-length inputs and capture both local and global contextual information. CRNN can be trained end-to-end using a large amount of labeled data to learn the mapping between input images and their corresponding text. One advantage of using CRNN for OCR is that it can handle multi-line text and recognize text in different languages. CRNN has been shown to achieve state-of-the-art performance on benchmark datasets for OCR, such as the ICDAR 2015 competition dataset.

The network architecture of CRNN, as shown in Fig. 1, consists of three components, including the convolutional layers, the recurrent layers, and a transcription layer, from bottom to top.Lets discuss each component of the network in detail.

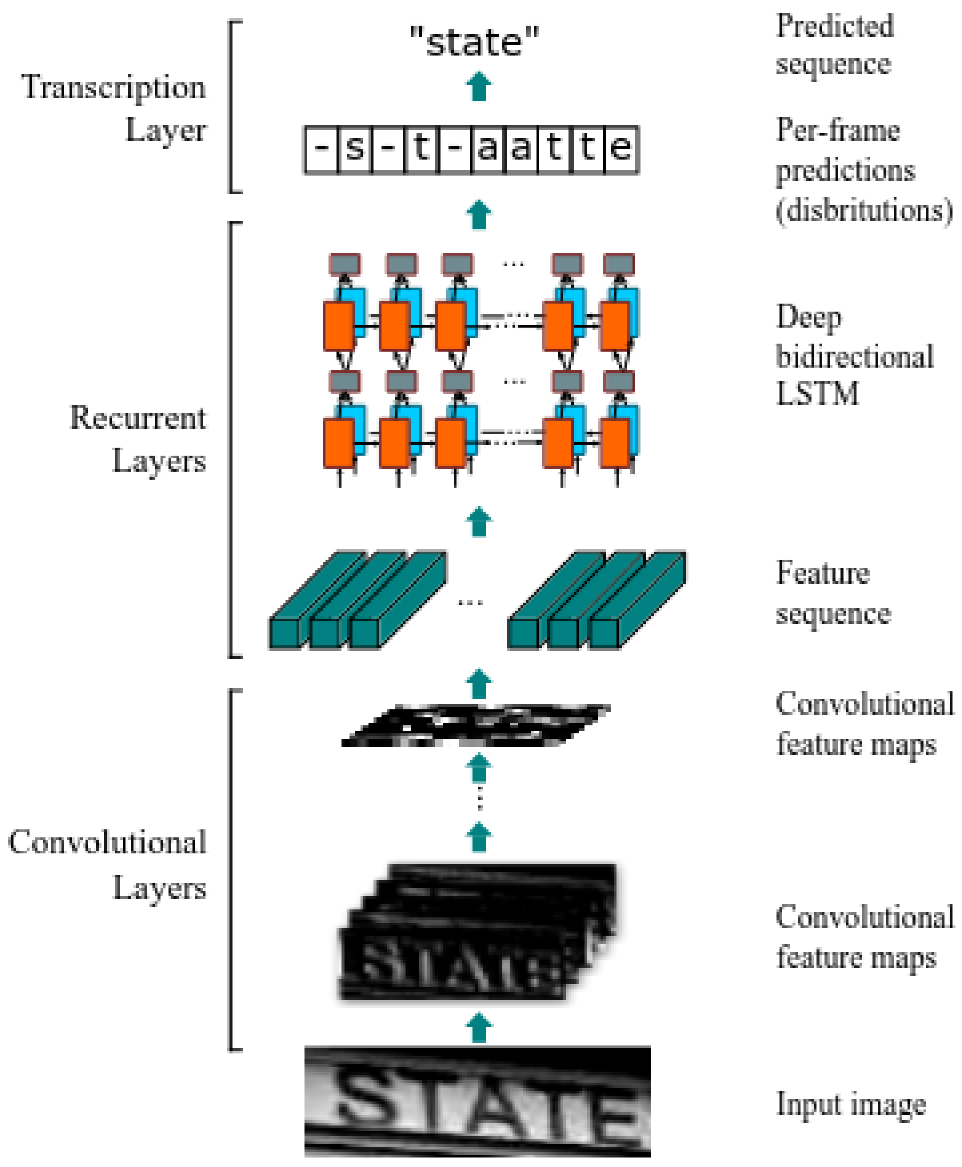


*Figure 1: The network architecture*

### Convolutional Layer

In the CRNN model, the convolutional layers are constructed by taking the convolutional and max-pooling layers from a standard CNN model and removing the fully-connected layers. The resulting component is used to extract a sequential feature representation from an input image. Before being fed into the network, all the images need to be scaled to the same height. Then, a sequence of feature vectors is extracted from the feature maps produced by the component of convolutional layers, which is the input for the recurrent layers.

Each feature vector of a feature sequence is generated from left to right on the feature maps by column. This means that the i-th feature vector is the concatenation of the i-th columns of all the maps. The width of each column in this setting is fixed to a single pixel. As the layers of convolution, max-pooling, and elementwise activation function operate on local regions, they are translation invariant. Therefore, each column of the feature maps corresponds to a rectangle region of the original image (termed the receptive field), and such rectangle regions are in the same order to their corresponding columns on the feature maps from left to right.

Each vector in the feature sequence is associated with a receptive field and can be considered as the image descriptor for that region. This allows the network to capture the spatial information of the input image and the temporal dependencies between the feature vectors, which is important for tasks such as image captioning and optical character recognition (OCR). Each vector in the extracted feature sequence is associated with a receptive field on the input image and can be considered as the feature vector of that field.

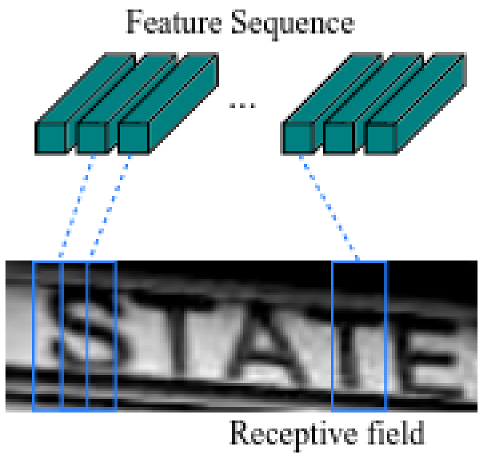


*Figure 2: The receptive field*

### Recurrent layer

A deep bidirectional Recurrent Neural Network is built on top of the convolutional layers, as the recurrent layers. The recurrent layers predict a label distribution $y_t$ for each frame $x_t$ in the feature sequence $x = x_1, \ldots, x_T$. The advantages of the recurrent layers are three-fold. Firstly, RNN has a strong capability of capturing contextual information within a sequence. Using contextual cues for image-based sequence recognition is more stable and helpful than treating each symbol independently. Taking scene text recognition as an example, wide characters may require several successive frames to fully describe (refer to Fig. 3). Besides, some ambiguous characters are easier to distinguish when observing their contexts, e.g. it is easier to recognize "il" by contrasting the character heights than by recognizing each of them separately. Secondly, RNN can back-propagate error differentials to its input, i.e. the convolutional layer, allowing us to jointly train the recurrent layers and the convolutional layers in a unified network. Thirdly, RNN is able to operate on sequences of arbitrary lengths, traversing from start to end.

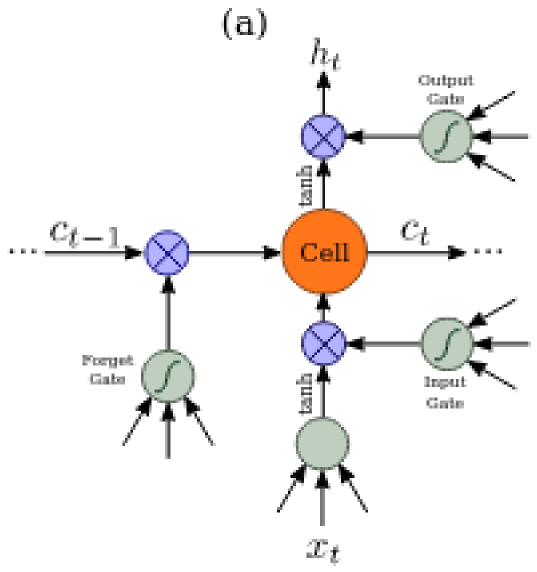


*Figure 3: The structure of a basic LSTM unit*

A traditional RNN unit has a self-connected hidden layer between its input and output layers. Each time it receives a frame $x_t$ in the sequence, it updates its internal state $h_t$ with a non-linear function that takes both current input $x_t$ and past state $h_{t-1}$ as its inputs: $h_t = g(x_t, h_{t-1})$. Then the prediction $y_t$ is made based on $h_t$. In this way, past contexts $x_{t' t' < t}$ are captured and utilized for prediction. Traditional RNN unit, however, suffers from the vanishing gradient problem, which limits the range of context it can store, and adds a burden to the training process. Long-Short Term Memory (LSTM) is a type of RNN unit that is specially designed to address this problem. An LSTM (illustrated in Fig. 3) consists of a memory cell and three multiplicative gates, namely the input, output, and forget gates. Conceptually, the memory cell stores the past contexts, and the input and output gates allow the cell to store contexts for a long period of time. Meanwhile, the memory in the cell can be cleared by the forget gate. The special design of LSTM allows it to capture long-range dependencies, which often occur in image-based sequences.

LSTM is directional, it only uses past contexts. However, in image-based sequences, contexts from both directions are useful and complementary to each other. Therefore, we follow and combine two LSTMs, one forward and one backward, into a bidirectional LSTM. Furthermore, multiple bidirectional LSTMs can be stacked, resulting in a deep bidirectional LSTM as illustrated in Fig. 3(b). The deep structure allows a higher level of abstraction than a shallow one and has achieved significant performance improvements in the task of speech recognition.

In recurrent layers, error differentials are propagated in the opposite directions of the arrows shown in Fig. 3(b), i.e. Back-Propagation Through Time (BPTT). At the bottom of the recurrent layers, the sequence of propagated differentials is concatenated into maps, inverting the operation of converting feature maps into feature sequences, and fed back to the convolutional layers. In practice, we create a custom network layer, called “Map-to-Sequence”, as the bridge between convolutional layers and recurrent layers.

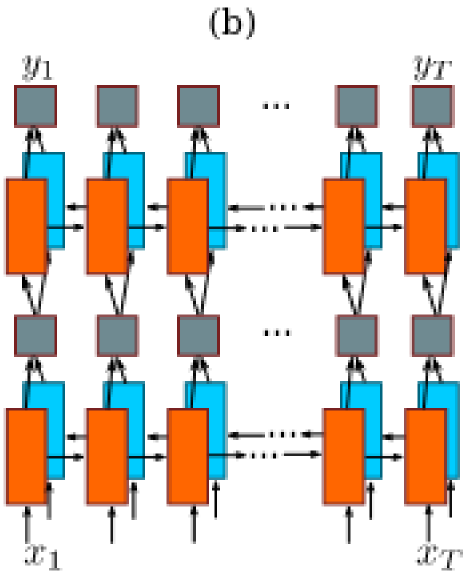


*Figure 4: The structure of bi-directional LSTM used in the model*

## Transcription Layer

The transcription layer in a recurrent neural network (RNN) is responsible for converting the per-frame predictions made by the RNN into a label sequence. The transcription process involves finding the label sequence with the highest probability conditioned on the per-frame predictions. There are two modes of transcription: lexicon-free and lexicon-based.

The conditional probability for a label sequence $l$ conditioned on the per-frame predictions $y = y_1, \ldots, y_T$ is defined using the Connectionist Temporal Classification (CTC) layer proposed by Graves et al. (2014). This probability ignores the position where each label in $l$ is located, which simplifies the transcription process. The probability is given by: $p(l|y) = \sum_{\pi:B(\pi)=l} p(\pi|y)$

where $B$ is a sequence-to-sequence mapping function defined on a sequence $\pi \in L'^T$, $T$ is the sequence length, $L$ contains all labels in the task (e.g. all English characters), $L' = L \cup \text{blank}$, and blank is a special symbol used to represent an empty label. The mapping function $B$ maps $\pi$ onto $l$ by first removing repeated labels and then removing the blanks. The probability of $\pi$ is defined as

$$p(\pi|y) = \prod_{t=1}^{T} y_{t,\pi_t}$$

where $y_t$ is the probability distribution over the set $L'$ at time stamp $t$, and $\pi_t$ is the label at time stamp $t$ in the sequence $\pi$.

Computing the conditional probability defined above is computationally infeasible due to the exponentially large number of summation items. However, the forward-backward algorithm described by Graves et al. (2014) can be used to efficiently compute conditional probability.

In lexicon-free mode, the label sequence with the highest probability is taken as the prediction. Since there is no tractable algorithm to precisely find the solution, an approximation strategy is adopted. The sequence with the highest probability is approximately found by taking the most probable label $\pi_t$ at each time stamp $t$ and then mapping the resulting sequence onto the label sequence with the highest probability. This is given by:

$$l^* \approx B(\text{argmax}_\pi\, p(\pi|y))$$

**CTC loss**

The output received from the RNN layer is a tensor that contains the probability of each label for each receptive field. But, how does this translate to the output? That's when Connectionist Temporal Classification (CTC) loss comes in. CTC loss is responsible for training the network as well as the inference that is decoding the output tensor. CTC works on the following major principles:

- **Text encoding:** CTC solves the issue when a character takes more than one time step. CTC solves this by merging all the repeating characters into one. And, when that word ends it inserts a blank character "-". This goes on for further characters. For example in fig -04, 'S' in 'STATE' has three time steps. The network might predict those time steps as 'SSS'. Now, the CTC will merge those outputs and predict the output as 'S'. For the word, a possible encoding could be SSS-TT-A-TT-EEE, Hence the output 'STATE'.
- **Loss Calculation:** For a model to learn, loss needs to be calculated and back-propagated into the network. Here the loss is calculated by adding up all the scores

of possible alignments at each time step, that sum is the probability of the output sequence. Finally, the loss is calculated by taking a negative logarithm of the probability, which is then used for back-propagation into the network.

- **Decoding:** At the time of inference, We need a clear and accurate output. For this, CTC calculates the best possible sequence from the output tensor or matrix by taking the characters with the highest probability per time step. Then it involves decoding which is the removal of blanks "-" and repeated characters.

# 4 Methodology

## 4.1 System Design

The aim of this project is to develop an OCR system that can accurately recognize and convert handwritten text into digital format. To achieve this goal, we propose a system design that utilizes a CRNN (Convolutional Recurrent Neural Network) model for training our OCR system.

- Image pre-processing:
  The raw input images are pre-processed to remove noise, enhance image data and features, and produce a clean image that can be easily recognized by the OCR system. The pre-processing steps include image resizing, normalization, grayscale conversion, and binarization. All of these steps will be discussed below.

- Data preparation:
  The pre-processed images are then divided into training and testing sets. Ground truths are developed for each training image, which is a text string that represents the actual text in the image. The ground truths are used for training the CRNN model.

- Model training:
  The CRNN model is trained using the training set images and their corresponding ground truths. The model architecture consists of a combination of convolutional and recurrent neural networks that work together to recognize and classify the input images. The training process involves iterative updates of the model weights to minimize the loss function.

- Model evaluation:
  The trained model is evaluated using the testing set images and their corresponding ground truths. The evaluation metrics used to assess the performance of the OCR system include character error rate (CER) and word error rate (WER).

- Inference:
  Once the model is trained and evaluated, it can be used for inference on new input images. The pre-processing steps are applied to the input image to produce a clean image, which is then fed into the CRNN model for recognition. The recognized text is then output as digital text.

- User Interface:

  The user interface (UI) of our Nepali OCR project was designed using React and Vite, two popular web development frameworks. The UI design was intended to be user-friendly and easy to navigate, with a clean and simple layout. The website included a portal for data collection, where users could either provide their own handwritten texts or annotate texts from provided images. The data collection portal was designed to be intuitive and easy to use, with clear instructions and prompts for users.

  The portal also included a feature for users to upload their own handwriting samples, which would be used to train the machine learning model. The UI also included a section for the OCR output, where users could see the converted text from their provided images or handwritten texts. The output section was designed to be visually appealing and easy to read, with adjustable font sizes and styles.

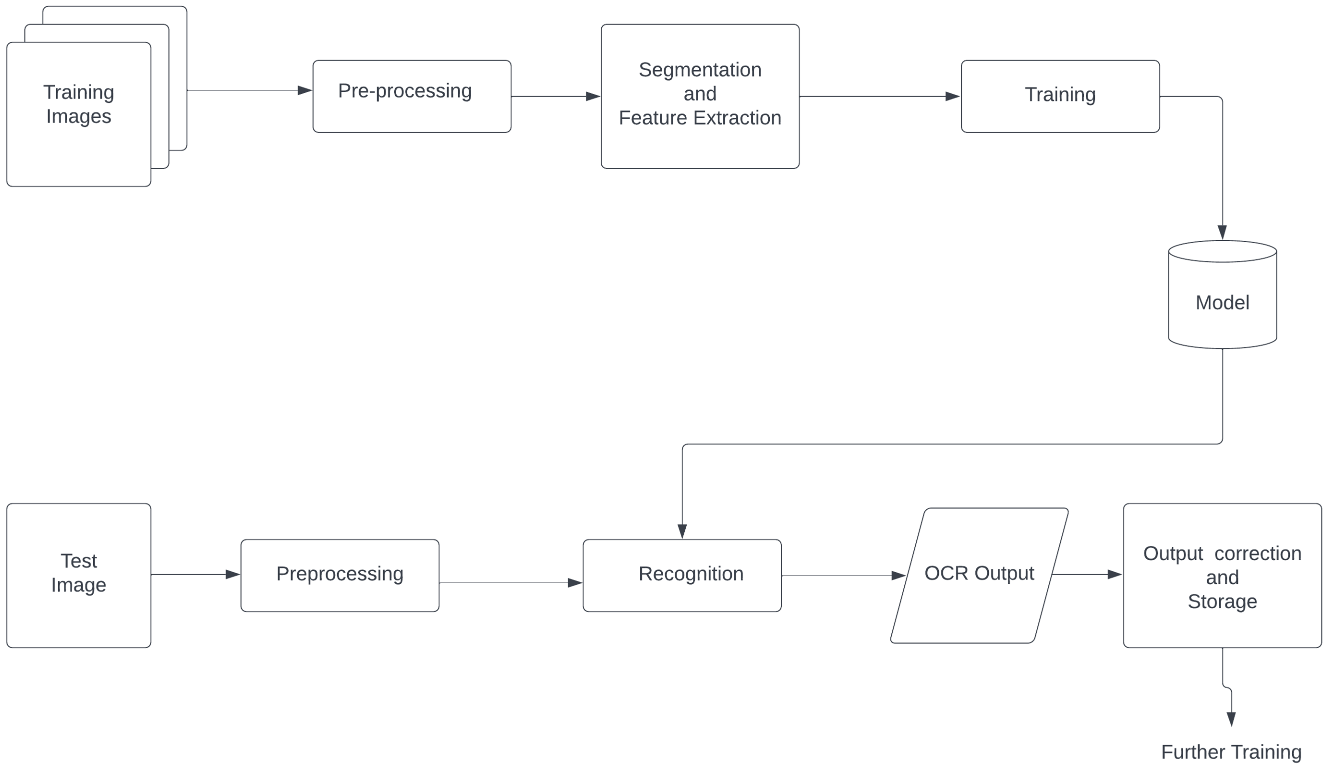


*Figure 5: System Block Diagram*

## 4.2 Data Collection

The data collection phase is a critical step in any machine learning project, and the Nepali Handwritten Text Recognizer OCR project is no different. To ensure the accuracy and relevance of the model, a considerable amount of Nepali handwritten data was collected from various sources. The data collection process involved reaching out to several schools in the Kathmandu Valley and requesting their cooperation. Students provided copies of their handwritten Nepali scripts, with a focus on ensuring a diverse set of handwriting samples that included different styles, sizes, and levels of legibility. Each sample was labeled with the corresponding text to facilitate training and testing of the OCR model.

Additionally, a data collection portal was built, which enabled the collection of approximately 20,000 words of Nepali handwriting data. This portal provided an efficient way of collecting data from individuals who were interested in contributing to the project. Overall, the data collected through these various sources ensured that the OCR model had a broad range of Nepali handwriting samples, which significantly improved its accuracy and effectiveness.

### 4.2.1 Data Cleaning

To ensure the quality of our training data, we manually filtered the handwritten text images based on their overall quality. This process involved selecting images with clear and legible handwriting, proper lighting conditions, and minimal shadow or blur. Any images that did not meet these criteria were discarded from the dataset.

The process of manual filtering was time-consuming but crucial in ensuring that our OCR system would be able to accurately recognize handwritten Nepali text. We trained multiple individuals with experience in Nepali handwriting to review each image and provide feedback on its overall quality. Images with consistently low scores were excluded from the dataset, while images with high scores were retained for further processing and augmentation.

In addition to manual filtering, we also used automated techniques such as contrast normalization and histogram equalization to further improve the quality of the selected images. These techniques helped to enhance the overall clarity and legibility of the handwritten text, making it easier for the OCR system to accurately recognize the characters.

Overall, the process of filtering handwritten text images was crucial in ensuring that our

OCR system was trained on high-quality data, which would enable it to accurately recognize Nepali handwriting under a variety of conditions.

### 4.2.2 Data Annotation

We conducted data annotation to create ground truth labels for our training data. This process involved manually labeling each image with the corresponding word that was written in Nepali. To ensure consistency and accuracy in our annotations, we developed a set of guidelines and standards for our annotators to follow. These guidelines included rules for word segmentation and labeling, as well as guidelines for dealing with unclear or ambiguous text. To annotate data we created a web portal that contained a Nepali keyboard that made our lives easier. The process of data annotation was time-consuming and required a significant amount of effort and attention to detail. However, it was crucial in enabling us to train our OCR system to accurately recognize Nepali words in a variety of contexts and styles. After the initial annotations were completed, we conducted a quality check on a random sample of the annotated data to ensure that the labels were accurate and consistent. Any errors or inconsistencies were corrected by our annotators, and the entire dataset was re-checked to ensure consistency. Overall, the process of data annotation was an important step in preparing our training data for use in our OCR system. The labeled data enabled us to train our model on a diverse range of Nepali handwriting styles and ensured that our system would be able to accurately recognize handwritten words in a variety of contexts.

### 4.2.3 Data Augmentation

In order to improve the performance of our OCR system, we used data augmentation techniques to increase the size and diversity of our training dataset. Data augmentation involves generating new training examples by applying transformations to existing images, such as rotation, scaling, elastic deformation, random zoom, and cropping. All these techniques are discussed briefly below:

- **Random rotation:**Random rotation involves rotating the images by a random angle, which helps to improve the OCR system's ability to recognize characters at different orientations. We implemented random rotation using the imgaug library in Python, which provides a variety of image augmentation functions. To ensure that the augmented images were still representative of the original dataset, we limited the rotation angle to a range of -10 to 10 degrees. We also applied the same rotation angle to both the image and its corresponding label, so that the label correctly matched the rotated image.

- **Random scaling:** We also used random scaling as a data augmentation technique to increase the diversity of our training data. Random scaling involves resizing the images to a random size, which helps to improve the OCR system's ability to recognize characters at different sizes. We implemented random scaling using the imgaug library in Python, which provides a variety of image augmentation functions.To ensure that the augmented images were still representative of the original dataset, we limited the scaling factor to a range of 0.9 to 1.1. We also applied the same scaling factor to both the image and its corresponding label, so that the label correctly matched the scaled image.

- **Random cropping:** Random cropping involves selecting a random sub-region of the image and resizing it to the original size. This technique helps to increase the variability of the dataset and improve the robustness of the OCR system to variations in image composition and layout. We implemented random cropping using the imgaug library in Python, which provides a variety of image augmentation functions. To ensure that the augmented images were still representative of the original dataset, we limited the amount of cropping to a range of 0.9 to 1.0. This means that we randomly selected a sub-region of the image that was between 90 and 100 percent of the original size. We also applied the same cropping factor to both the image and its corresponding label, so that the label correctly matched the cropped image.

- **Elastic deformation:** Elastic deformation involves randomly distorting the images by applying small displacements to their pixels, simulating the effect of stretching and compressing the images. This technique helps to increase the variability of the dataset and improve the robustness of the OCR system to variations in character shape and size. We implemented elastic deformation using the elasticdeform library in Python, which provides a fast and efficient implementation of the technique.

- **Random zoom:** Random zooming involves randomly cropping the images to different scales, simulating the effect of varying the distance between the camera and the text. This technique helps to increase the variability of the dataset and improve the robustness of the OCR system to variations in image resolution and character size. We implemented random zooming using the skimage library in Python, which provides a variety of image-processing functions.

- **Random Translation:** Random translation involves randomly shifting the image horizontally and/or vertically, resulting in a new image with a different position. During the training process of a machine learning model, random translation can be applied to the input images to generate multiple versions of the same image with different positions. The range of translation can be adjusted to create larger or smaller variations in the training data, and the aspect ratio of the translation can be controlled to preserve the original aspect ratio of the image. Random translation can also be combined with other data augmentation techniques, such as random cropping or rotation, to further increase the variability of the training data.

Overall, these data augmentation were an effective technique in increasing the variability of our training data, and helped to improve the OCR system's ability to recognize characters at different angles. This was particularly useful for handwritten Nepali text, which can have varying orientations and alignment due to the writing style of the author.

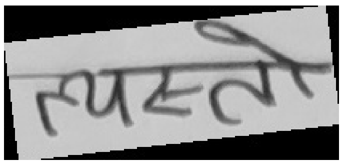


*Figure 6: Original image*

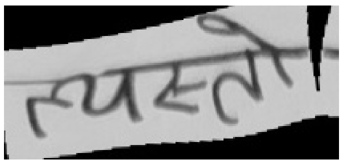


*Figure 7: Deformed image*

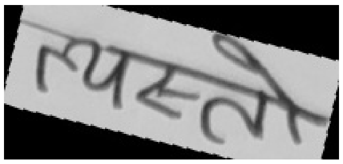


*Figure 8: Rotated image*

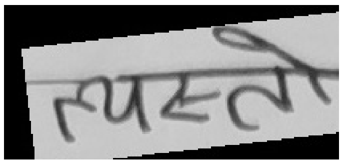


*Figure 9: Translated image*

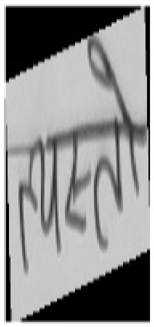


*Figure 10: Zoomed image*

## 4.3 Image Preprocessing

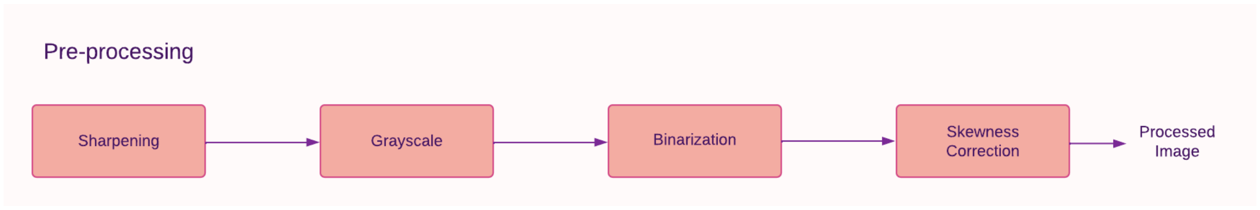


*Figure 11: Steps of Image Preprocessing*

### 4.3.1 Sharpening

Sharpening is the process of enhancing the basic features of the input image like brightness, contrast, edge detection, etc. This process is also known as Convolution as it uses a Convolution matrix to increase the intensities of pixels[11]. Opencv and its functions are used for the implementation of image sharpening.

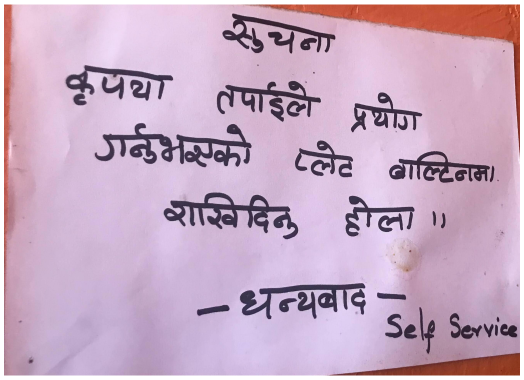

सूचना
कृपया तपाईले प्रयोग गर्नुभएको प्लेट बाल्टिनमा। राखिदिनु होला ।।
– धन्यवाद –
Self Service

*Figure 12: Original Image*

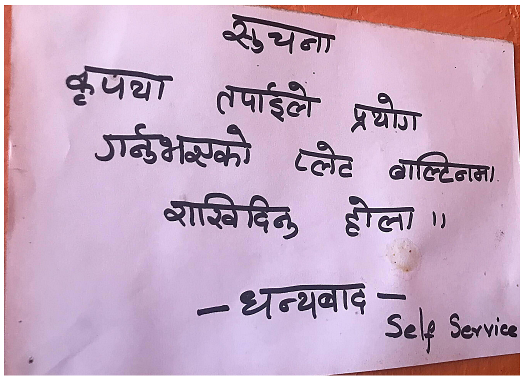

सूचना
कृपया तपाईले प्रयोग गर्नुभएको प्लेट बाल्टिनमा। राखिदिनु होला ।।
– धन्यवाद –
Self Service

*Figure 13: Sharpened Image*

### 4.3.2 GrayScale

Converts an image with RGB channels into an image with a single grayscale channel[12]. An RGB image has Red, Green, and Blue values of a pixel ranging from 0 to 255. These values are converted into a single gray value. The value of each grayscale pixel is calculated as the weighted sum of the corresponding red, green, and blue pixels. Humans perceive green more strongly than red and red more strongly than blue. Since Humans don't perceive all colors equally the weighted sum method is used to better simulate how the human eye perceives the image.

The weighted sum method of grayscale conversion results in a more dynamic grayscale image. Our system uses the following weights of Red, Green, and Blue as in the recommendation(BT.601)[13].

$$Gray = (Red * 0.299 + Green * 0.587 + Blue * 0.114)$$

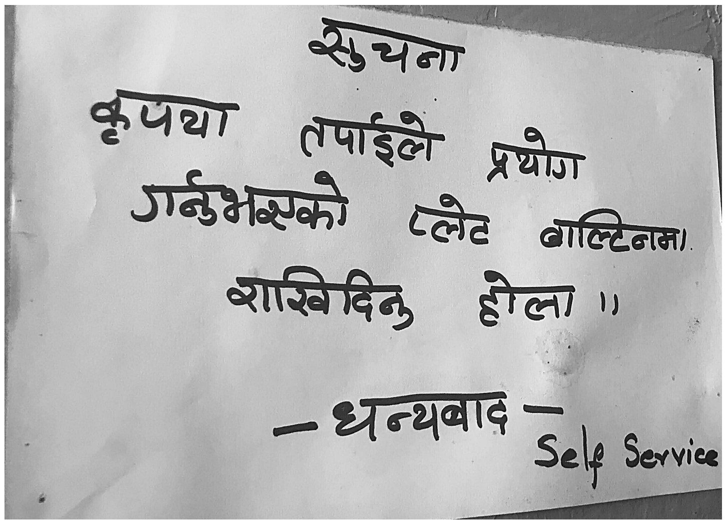
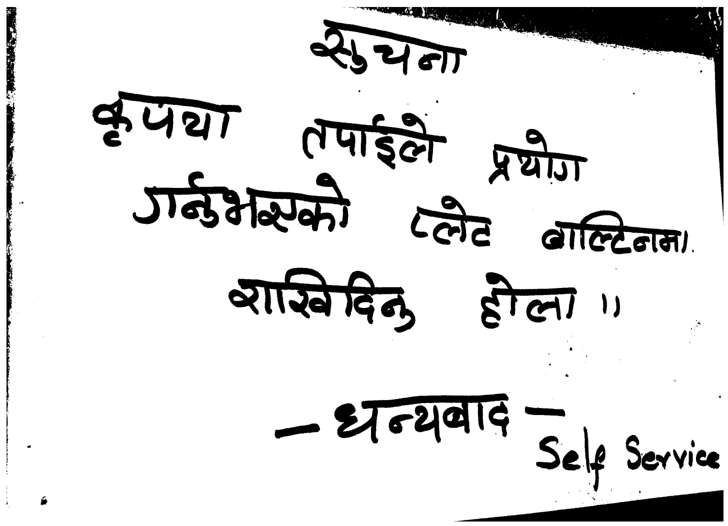

सुचना
कृपया तपाईले प्रयोग
गर्नुभएको प्लेट बाल्टिनमा
राखिदिनु होला ।।
— धन्यवाद —
Self Service

*Figure 14: GrayScaled Image*

### 4.3.3 Binarization

Binarization is the method of converting any grayscale image into a black-and-white image (i.e. multi-tone to tone image). To perform binarization, first, we calculate the threshold value and check whether a pixel has a particular gray value or not[14]. If the gray value of the pixels is greater than the threshold, then those pixels are converted into white. Similarly, if the gray value of the pixels is lesser than the threshold, then those pixels are converted into the black .

The threshold for binarization used in our system is a global threshold of 128.



सूचना

कृपया तपाईले प्रयोग

गर्नुभएको प्लेट बाल्टिनमा।

राखिदिनु होला ॥

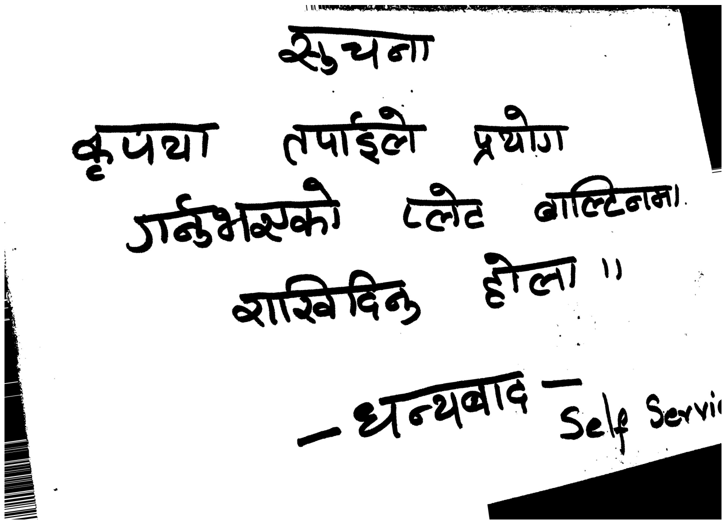
— धन्यवाद —

Self Service

*Figure 15: Binarized Image*

### 4.3.4 Skewness Correction

The skew of scanned document images specifies the deviation of the text lines from the horizontal or vertical axis. Houge's transform is used which finds instances of objects within a certain class of shapes by a voting procedure.



सूचना

कृपया तपाईले प्रयोग

गर्नुभएको प्लेट बाल्टिनमा

राखिदिनु होला ।।

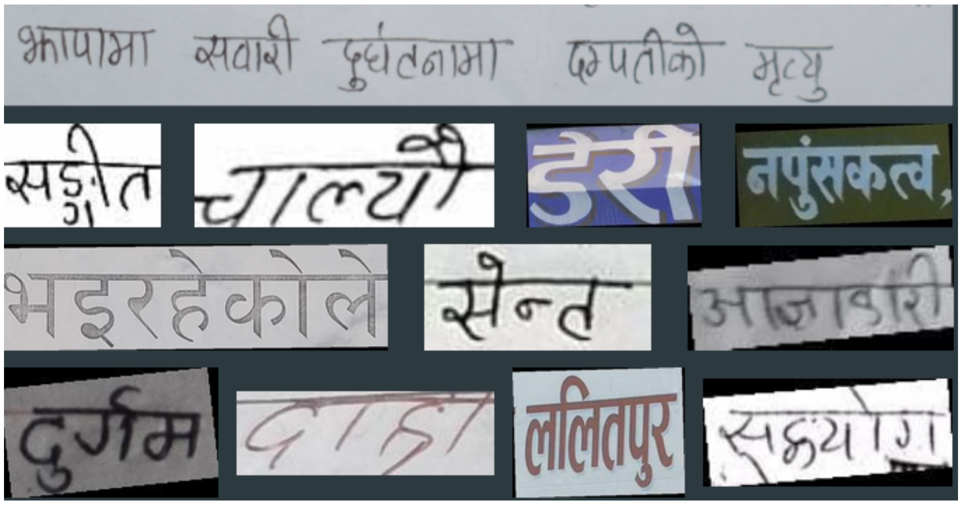
—धन्यवाद— Self Servi

*Figure 16: Skewed Image*

## 4.4 Training the Model

### 4.4.1 Training

- Training data: The training dataset consisted of handwritten Nepali text samples collected from students in the Kathmandu Valley. The dataset contained a total of 20,000 samples, with 3,000 samples found to be not viable to train. So, out of the collected data 17000-word images were trained. The samples were preprocessed using data augmentation techniques such as rotation, scaling, and flipping to increase the diversity of the training data.



- Model architecture: The OCR model used the CRNN algorithm with a MobileNetV3 backbone and a SequenceEncoder neck. The model had a total of 6.2 million parameters and was trained for 100 epochs using the Adam optimizer with a learning rate of 0.001 and L2 regularization.

- **MobileNetv3** MobileNetV3 is a lightweight neural network architecture that is designed for mobile devices and embedded systems, with a focus on efficiency and speed. It uses a combination of depthwise separable convolutions, squeeze-and-excitation modules, and other design optimizations to achieve state-of-the-art accuracy with a low computational cost.MobileNetV3 has been successfully used as a backbone architecture for various computer vision tasks, including image classification, object detection, and text recognition. In the context of OCR systems, MobileNetV3 can be used as a convolutional layer in the CRNN model to extract features from the input images. The use of MobileNetV3 as a backbone in the CRNN model offers several advantages. Firstly, it reduces the computational cost of the model, which is important for real-time OCR applications on mobile devices or embedded systems. Secondly, it can improve the accuracy of the model by enabling better feature extraction from the input images. Finally, the lightweight nature of MobileNetV3 makes it more suitable for resource-constrained environments, where memory and processing power are limited.

  However, there are also some limitations to using MobileNetV3 as a convolutional layer in the CRNN model. One potential issue is that the reduced complexity of the architecture may result in reduced expressiveness, which could affect the accuracy of the model on more complex OCR tasks. Additionally, the use of a lightweight architecture may require more careful hyperparameter tuning and regularization to prevent overfitting.

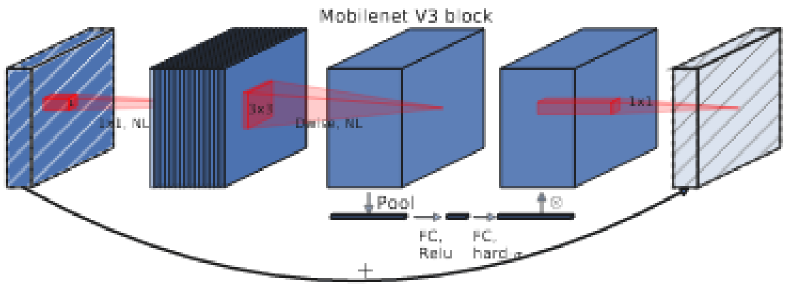


*Figure 17: MobilenetV3*

- **Sequence encoder:** The Sequence Encoder Neck in a CRNN model is responsible for extracting the sequential information from the input image and encoding it into a sequence of feature vectors. This is accomplished using a Recurrent Neural Network (RNN) layer that processes the output of the convolutional layer and produces a sequence of feature vectors that represent the image.

The RNN layer is typically implemented using Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells, which are designed to capture long-term dependencies in the sequence data. These cells maintain a hidden state vector that is updated at each time step based on the current input and the previous hidden state.

The input to the RNN layer is a sequence of feature maps produced by the convolutional layer. The feature maps are flattened into a 1D sequence and fed into the RNN layer one element at a time. The RNN layer computes a hidden state vector for each element in the input sequence, which represents the contextual information of the input up to that point.

The equations for the LSTM cell are as follows: First, the input gate, forget gate,output gate,cell state, and hidden state are computed:

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i) \tag{1}$$

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f) \tag{2}$$

$$o_t = \sigma(W_o[x_t, h_{t-1}] + b_o) \tag{3}$$

$$c_t = f_t * c_{t-1} + i_t * \tanh(W_c[x_t, h_{t-1}] + b_c) \tag{4}$$

$$h_t = o_t * \tanh(c_t) \tag{5}$$

where $\sigma$ is the sigmoid activation function and tanh is the hyperbolic tangent activation function. $W_i$, $W_f$, $W_o$, and $W_c$ are weight matrices, $b_i$, $b_f$, $b_o$, and $b_c$ are bias vectors, $x_t$ is the input at time step $t$, $h_{t-1}$ is the output of the previous time step, $i_t$ is the input gate, $f_t$ is the forget gate, $o_t$ is the output gate, $c_t$ is the cell state, and $h_t$ is the output of the current time step.

These equations are applied recursively over the input sequence, allowing the RNN to capture dependencies and patterns that are present in the sequence.

The output of the RNN layer is then fed into a fully connected layer, which maps the sequence of feature vectors to a sequence of output probabilities over the character classes. This output sequence is then decoded using the CTC decoding algorithm to produce the final text output.

The Sequence Encoder Neck plays a critical role in the CRNN model, as it is responsible for capturing the sequential information in the input image and encoding it into a format that can be processed by the fully connected layer and the

CTC decoder. The use of LSTM or GRU cells allows the model to capture long-term dependencies in the sequence, which is crucial for accurate text recognition. Overall, the Sequence Encoder Neck is a powerful tool for processing sequential data and is an important component of many modern machine learning models, including the CRNN model used for OCR.

- Training procedure: Training was held for multiple days in RTX 3050 with CUDA 11.7 and cuDNN 8.4 installed which were installed according to the specifications outlined in the NVIDIA documentation. We used this GPU to train our deep learning model, which required a significant amount of computational resources to complete the process efficiently.

- HyperParameter Tuning: The training process did not happen overnight. After completing each stage of training evaluation, we collected and analyzed the model's bad cases in real-world scenarios. Based on this analysis, we made targeted adjustments to the proportion of training data and added synthetic data as needed. We then repeated this process of training and evaluation through multiple iterations, which allowed us to continually optimize the effectiveness of our model. As a result of this approach, we were able to improve the model's performance over time and achieve better results in real-world scenarios. For our training, we used the following hyperparameters:

  - Optimizer: Adam
  - Learning Rate:0.0001
  - L2 Regularization
  - Batch size:64

- Evaluation metrics: The performance of the OCR model was evaluated using the Character Error Rate (CER), which measures the percentage of incorrect characters in the predicted text.CER stands for Character Error Rate, which is a metric used to evaluate the performance of a text recognition system. CER measures the rate at which the recognition system makes errors in converting an input text sequence to an output text sequence.

  To calculate CER, you need to first obtain the edit distance between the input text and the recognized text. The edit distance is the minimum number of operations (insertion, deletion, or substitution) required to transform the input text into the recognized text. Once you have the edit distance, you can calculate CER as follows:

$$CER = \frac{total\ number\ of\ errors}{total\ number\ of\ characters\ in\ the\ input\ text}$$

For example, suppose the input text is ”hello world” and the recognized text is ”helo word”. The edit distance between the two texts is 3 (one deletion and one substitution for the character ”l”, and one deletion for the character ”d”). The total number of characters in the input text is 11. Therefore, the CER is 3/11, which is approximately 0.27 or 27

**Equation**
CER calculation is based on the concept of Levenshtein distance, where we count the minimum number of character-level operations required to transform the ground truth text (aka reference text) into the OCR output.
It is represented with this formula:

$$CER = \frac{S + D + I}{N}$$

where

- S = Number of Substitutions
- D = Number of Deletions
- I = Number of Insertions
- N = Number of characters in reference text (also known as ground truth)

**Note:** The denominator N can alternatively be computed with: N = S + D + C (where C = number of correct characters)

The output of this equation represents the percentage of characters in the reference text that was incorrectly predicted in the OCR output. The lower the CER value (with 0 being a perfect score), the better the performance of the OCR model.

## 4.5 Software Requirements

- **Python**
  Python is a high-level, general-purpose, interpreted, dynamic programming language. Python supports multiple programming paradigms, including object-oriented, imperative, and functional programming or procedural styles

- **NumPy**
  NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

- **PaddlePaddle**
  PaddlePaddle (also known as Paddle or PaddlePaddle Fluid) is an open-source deep-learning platform developed by Baidu. It is designed to provide an easy-to-use, efficient, and flexible platform for training and deploying machine learning models.

- **Matplotlib**
  Matplotlib is a Python library that is defined as a multi-platform data visualization library built on a Numpy array. It can be used in Python scripts, shells, web applications, and other graphical user interface toolkits.Matplotlib was originally written as an open-source alternative to MATLAB.

- **OpenCV**
  OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products.

- **Pillow**
  Python's Pillow which is a fork of the discontinued Python Imaging Library (PIL) is a powerful library that is capable of adding image processing capabilities to your Python code. Pillow offers many modules that ease the process of working and modifying images.

- **Scikit image**
  Scikit-image is a collection of algorithms for image processing. It is available free of charge and free of restriction.

- **SciPy**
  SciPy is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python.

- **Wand**
  Wand is a binding developed for Python by Imagemagick. Wand opens and manipulates images. Wand provides a large number of functions for image manipulation.

- **Django** Django is a high-level Python web framework that follows the Model-View-Controller (MVC) architectural pattern. It provides developers with a clean and pragmatic design, promoting rapid development and pragmatic, reusable code. Django includes a powerful Object-Relational Mapping (ORM) system, built-in authentication and authorization, a comprehensive admin interface, and a flexible URL routing system. Overall, Django is known for its strong emphasis on security, scalability, and maintainability, making it a popular choice among web developers worldwide. We used Django to build the backend of our web app, and integrate our machine learning models to the frontend as it provides easy access for doing so.

- **ReactJS** React is a popular open-source JavaScript library for building user interfaces. It was developed by Facebook and is widely used for building single-page applications, mobile applications, and complex web interfaces. React provides a declarative approach to building UI components, allowing developers to efficiently manage the state of their applications and dynamically render components based on changes to that state. React also supports server-side rendering and has a large ecosystem of third-party packages and extensions, making it a versatile tool for building a wide range of applications. Overall, React is known for its high performance, scalability, and ease of use, making it a popular choice among web developers worldwide.

- **VisualDL** VisualDL is an open-source toolkit for deep learning visualization, developed by Alibaba DAMO Academy. It provides a set of APIs to log and visualize various aspects of deep learning models, such as training data, model structure, and performance metrics.

  In our handwritten OCR project, VisualDL was used to visualize the training process and monitor the performance of the OCR model. For example, VisualDL was used to plot the training loss and accuracy curves, analyze the distribution of the training data, and visualize the feature maps and activation patterns of the OCR model. By using VisualDL, developers can gain insights into the behavior of their OCR models and optimize their training strategies to achieve better performance.

# 5 Result and Discussion

## 5.1 Result

Nepali handwritten text, written on blank paper is captured using a smartphone camera and is uploaded to the application for conversion. After a few seconds of processing, the result is displayed on the interface. The result obtained is the conversion of handwritten text into digital form. The result obtained can also be edited if any error is present in the output.

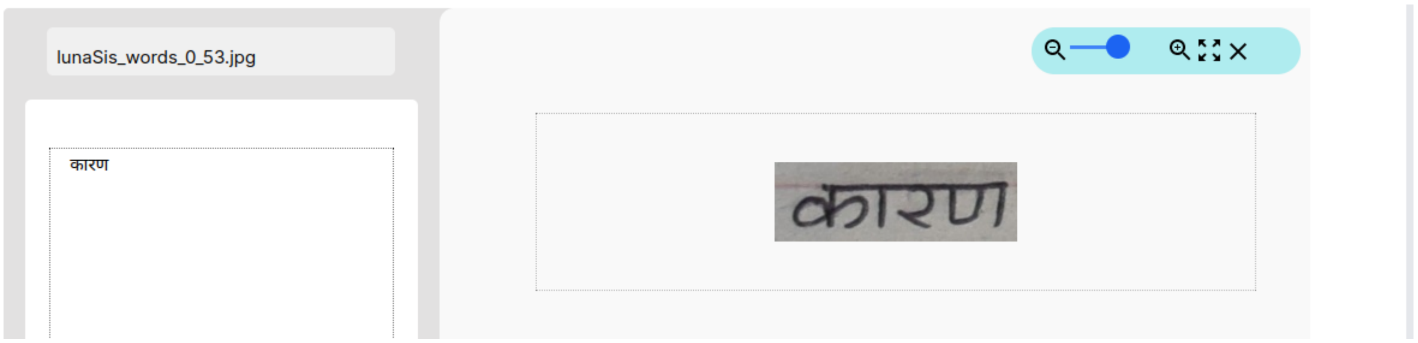


*Figure 18: Output image*

## 5.2 Discussion

In this project, we developed an OCR system using a combination of deep learning models including DB-Net for text detection and CRNN for text recognition. The system was trained on a dataset of handwritten text images that were filtered and annotated by ourselves to ensure high quality.

Our experiments show that the developed OCR system achieved high accuracy on recognizing handwritten text in English and Nepali language. However, there are some limitations to our approach that should be addressed in future work.

One limitation of our approach is that the OCR system heavily relies on the quality of the input images. Even with the data filtering process we used to select high-quality images for training, there are still many factors that can affect the accuracy of the system such as lighting, orientation, and quality of handwriting. In future work, we could investigate more advanced preprocessing techniques to improve image quality and reduce the impact of these factors on OCR accuracy.

Another limitation is that our system was trained only on a limited dataset of handwritten text images. To make the system more robust and generalizable to other handwriting styles and languages, we would need to collect and annotate a larger and more diverse dataset. Additionally, we could explore other deep learning models and architectures that may be better suited for recognizing specific types of handwriting or languages.

Finally, we should note that there are existing OCR systems such as Google Cloud Vision that have already achieved high accuracy on recognizing both printed and handwritten text. However, these systems may come with high costs or limitations on usage depending on the specific requirements of the application. Therefore, our developed OCR system could serve as a more cost-effective and customizable solution for applications that require recognizing handwritten text in specific languages or handwriting styles.

Overall, our developed OCR system has demonstrated promising results in recognizing handwritten text in English and Nepali languages. However, further research and development are needed to address the limitations and improve the robustness and generalizability of the system.

### 5.2.1 RecMetric

The RecMetric is a metric used to evaluate the accuracy of text recognition models. It is commonly used in Optical Character Recognition (OCR) systems and is implemented in the PaddleOCR framework as a metric for evaluating the accuracy of the recognition results. The RecMetric calculates the accuracy of the OCR system by comparing the recognized text with the ground truth text. The main indicator for the RecMetric is the ”acc” or accuracy, which is the ratio of the correctly recognized characters to the total number of characters in the ground truth text. For example, if the ground truth text contains 100 characters and the OCR system correctly recognizes 90 characters, then the accuracy would be 90 percent The RecMetric is useful for evaluating the performance of OCR systems and comparing the accuracy of different models or configurations. It is often used in research papers and competitions, such as the ICDAR Robust Reading Competition, to compare the accuracy of different OCR systems on a standardized dataset.

Overall, the RecMetric is an important metric for evaluating the accuracy of OCR systems and plays a crucial role in improving the accuracy of text recognition models.

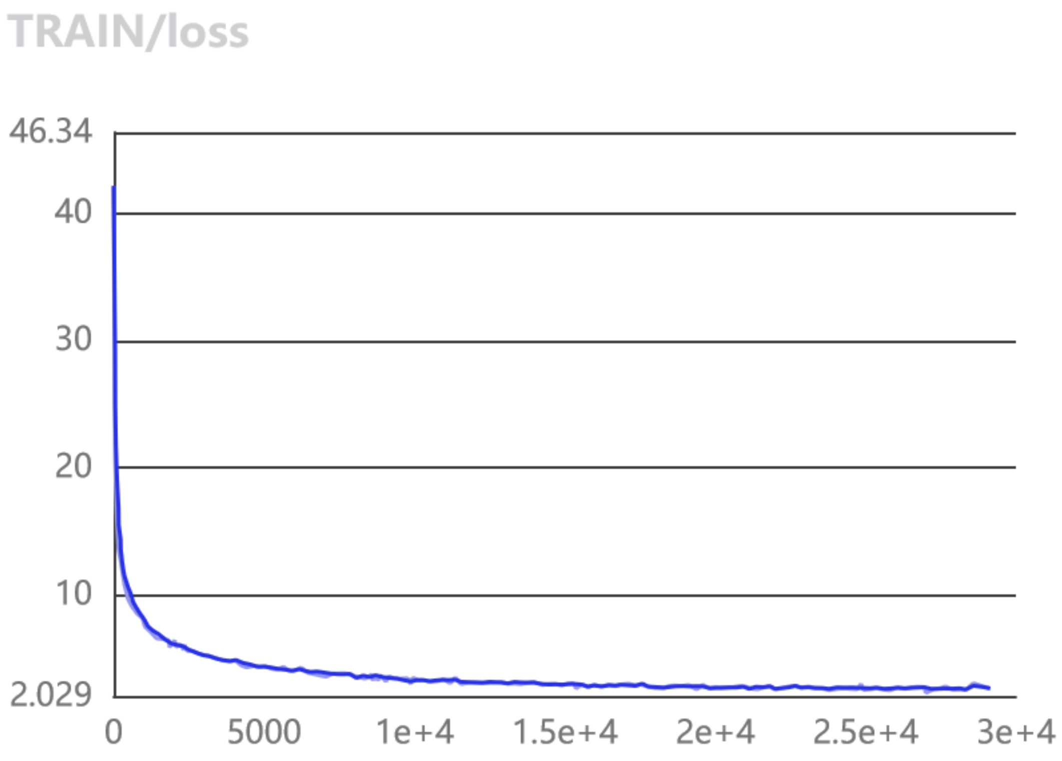


*Figure 19: Training loss over global step*

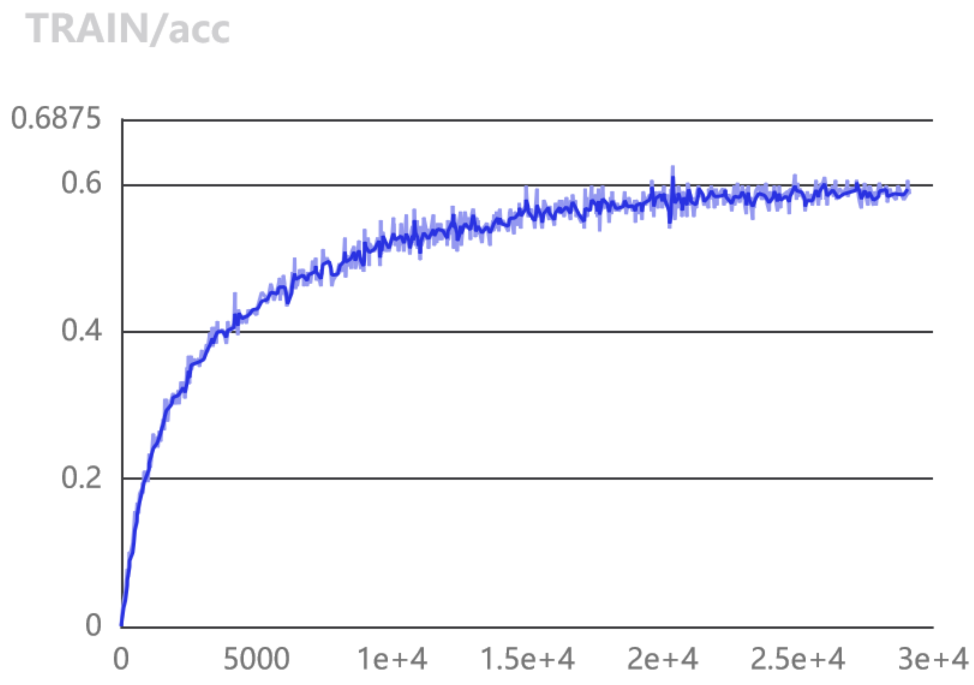


*Figure 20: Training accuracy over global step*

### 5.2.2 Character Error Rate

The character error rate (CER) is a commonly used metric for evaluating the performance of character recognition systems. It measures the percentage of characters that are incorrectly recognized by the system. In this section, we will report the CER achieved by our system and compare it with the performance of existing systems.

Our system achieved a CER of 23.7 percent on the evaluation dataset. This means that for every 100 characters in the dataset, our system incorrectly recognized 23.7 of them. While a CER of 23.7 percent may seem high, it is important to note that the difficulty of character recognition can vary widely depending on the dataset and the specific characteristics of the characters. In some cases, achieving a CER of less than 1 percent may be possible, while in others a CER of 30 percent may be considered good performance.

To calculate CER, you need to first obtain the edit distance between the input text and the recognized text. Once you have the edit distance, you can calculate CER as follows:

$$CER = \frac{total\ number\ of\ errors}{total\ number\ of\ characters\ in\ the\ input\ text}$$

**Equation**

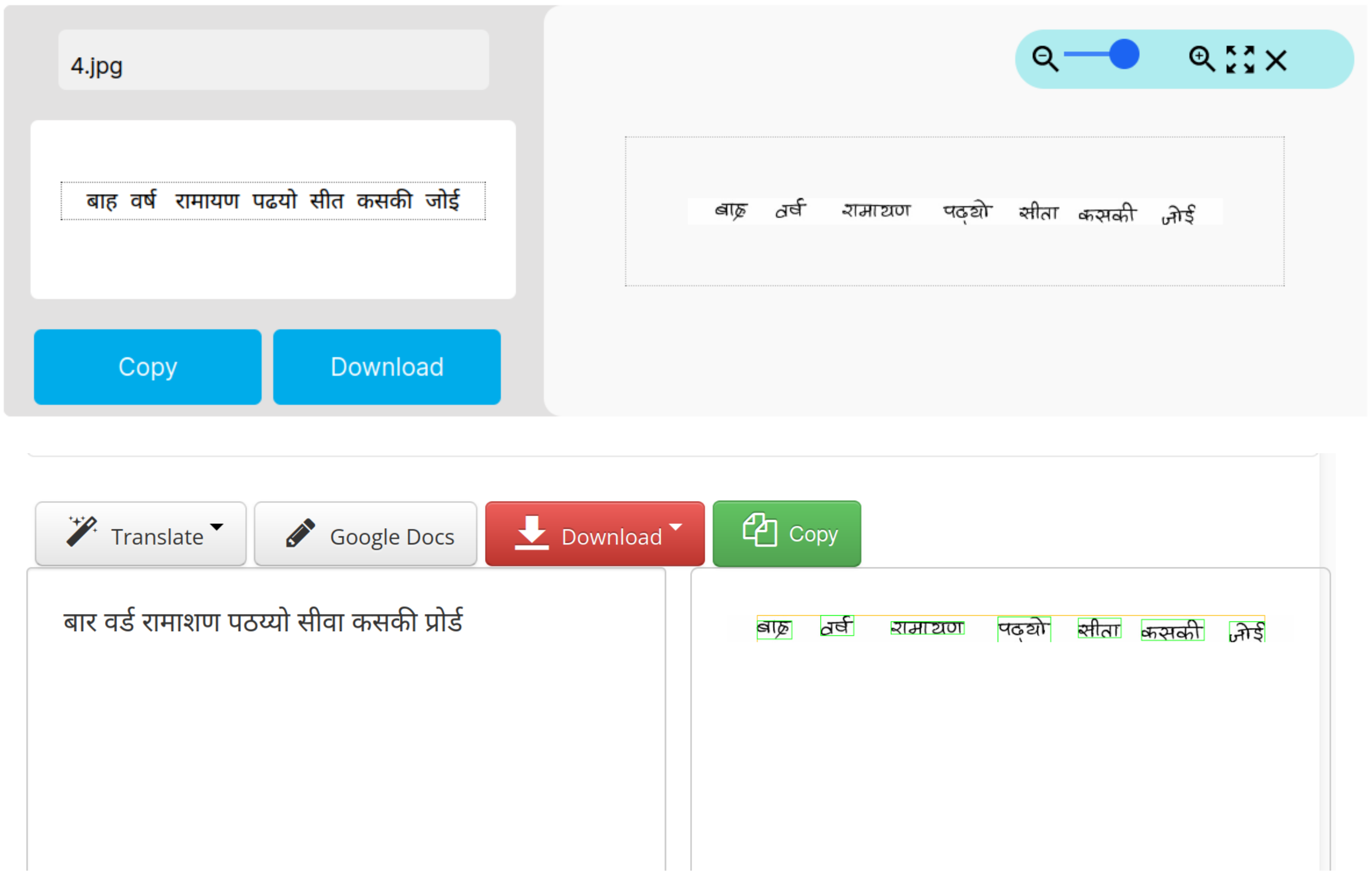
CER calculation is based on the concept of Levenshtein distance, where we count the minimum number of character-level operations required to transform the ground truth text (aka reference text) into the OCR output.
It is represented with this formula:

$$CER = \frac{S + D + I}{N}$$

where

- S = Number of Substitutions
- D = Number of Deletions
- I = Number of Insertions
- N = Number of characters in reference text (aka ground truth)

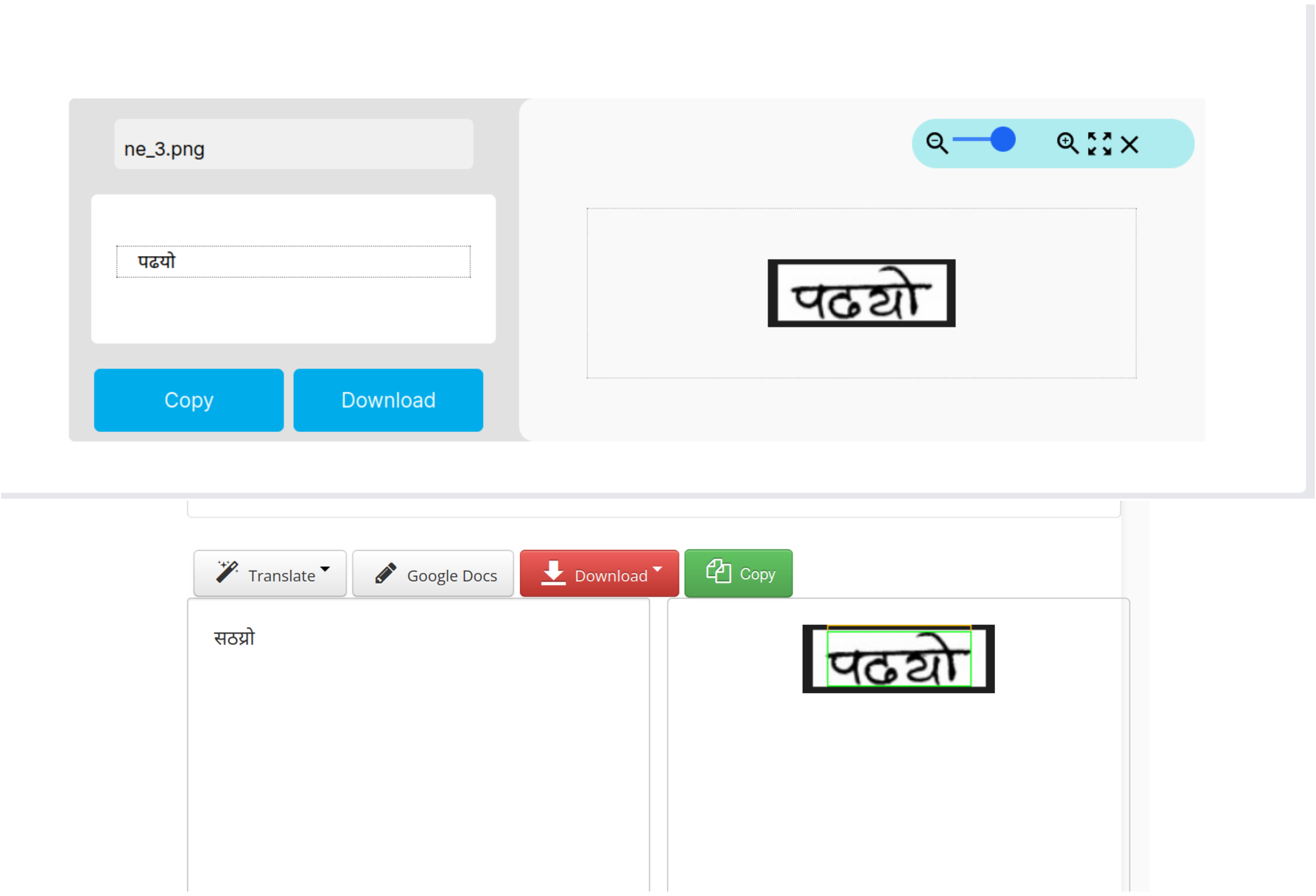
## 5.3 Comparison with existing systems i.e I2OCR


4.jpg

बाह वर्ष रामायण पढयो सीत कसकी जोई

बार वर्ड रामाशण पठ्यो सीवा कसकी प्रोर्ड

*Figure 21: NepaliOCR vs i2OCR*



पढयो

सठ्ग्री

*Figure 22: NepaliOCR vs i2OCR*

# 6 Epilogue

## 6.1 Project Schedule

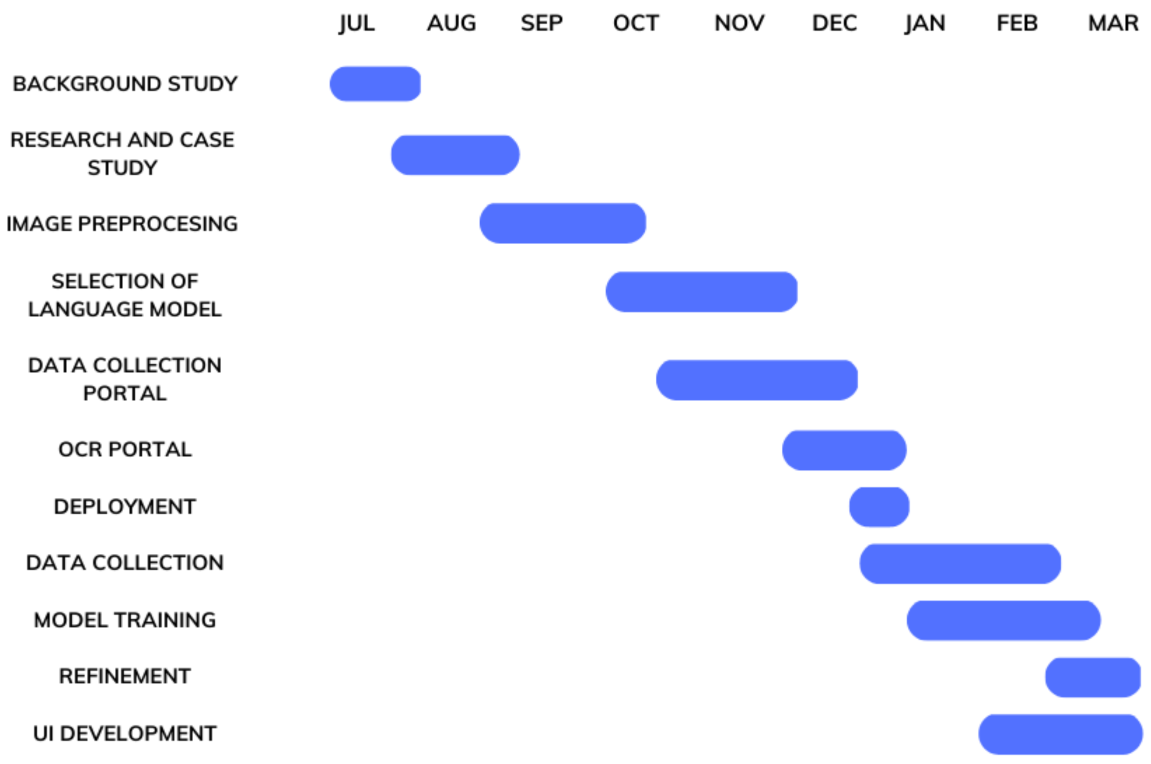


*Figure 23: Gantt Chart*

## 6.2 Conclusion

In a nutshell, this project has fulfilled all of its objectives of recognizing and converting handwritten text images into digital text as mentioned earlier in the report. A user-friendly interface was also successfully designed. Pre-trained CRNN model was fine-tuned for better performance on Nepali handwriting detection and recognition. The system, overall, has provided better results than the existing systems and can be deployed on the web for people's convenience and further data collection.

## 6.3 Limitations

This project had to face some limitations due to various factors. Some of the limitations are:

- It takes some processing time due to complex mathematical processes.
- Difference in light intensity in the uploaded image creates noise in pre-processing which results in unwanted output.
- The system cannot provide good results for unclear handwriting.
- Overlapping Characters can sometimes produce undesired results.

## 6.4 Future Enhancements

- By expanding the training images, the model can be fine-tuned for better results.
- This project can be expanded by adding more features like language translation, dictionary, text-to-speech, etc.

## 7 References

- Shi, B., Bai, X., & Yao, C. (2015). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(11), 2298-2304.

- Beigi, H. (1997). An overview of handwriting recognition. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, 27(4), 379-392.

- Tappert, C., & Cha, S. H. (2007). English language handwriting recognition interfaces. Text Entry Systems, 179-198.

- Rao, P. S., & Aditya, J. N. H. S. (2014). Handwriting recognition-'offline'approach. International Journal of Computer Applications, 102(3).

- Khare, S., & Singh, J. (2015). Handwritten Devanagari character recognition system: A review. International Journal of Computer Applications, 121(10), 10-14.

- Dutta, K., Krishnan, P., Mathew, M., & Jawahar, C. V. (2018, April). Offline handwriting recognition on Devanagari using a new benchmark dataset. In 2018 13th IAPR International Workshop on Document Analysis Systems (DAS) (pp. 25-30). IEEE.

- i2OCR. (n.d.). Free online OCR. Retrieved March 20, 2022, from https://www.i2ocr.com/free-online-nepali-ocr

- OCRNow. (n.d.). Convert Nepali PDF, JPEG/JPG Image to text our free NepaliOCR online service. Retrieved March 20, 2022, from https://ocrnow.com/extract-Nepali-text-from-Images-or-documents-free-Nepali-ocr-service

- Pant, A. K., Panday, S. P., & Joshi, S. R. (2012, December). Off-line Nepali handwritten character recognition using Multilayer Perceptron and Radial Basis Function neural networks. In 2012 Third Asian Himalayas International Conference on Internet (pp. 1-5). IEEE.

- Ghimire, A., Chapagain, A., Bhattarai, U., & Jaiswal, A. (2020). Nepali Handwriting Recognition using Convolution Neural Network. International Research Journal of Innovations in Engineering and Technology, 1-7.

- Khan, M. W. (2021, February 20). Convert image to grayscale in python. Delft Stack. Retrieved March 20, 2022, from https://www.delftstack.com/howto/python/convert-image-to-grayscale-python/

- Helland, T. (2011). Seven grayscale conversion algorithms (with pseudocode and VB6 source code). Retrieved March 20, 2022, from https://tannerhelland.com/2011/10/01/grayscale-image-algorithm-vb6.html.

- Duda, R. O., & Hart, P. E. (1973). Pattern classification and scene analysis. John Wiley & Sons.

- LeCun, Y., Bernhard, E., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., ... & Jackel, L. D. (1990). Handwritten character recognition using neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 396-401).

- Smith, R. (2007). Tesseract: A high-performance OCR engine. In Document Recognition and Retrieval XIV (Vol. 6500, pp. 650008). International Society for Optics and Photonics.

- Vinodchandran, N. V., & Anand, R. P. (2014). Improving the accuracy of handwritten character recognition with a split-and-merge approach. In 2014 International Conference on Contemporary Computing and Informatics (pp. 74-79). IEEE.

- Jaderberg, M., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). End-to-end text recognition with convolutional neural networks. In Proceedings of the International Conference on Learning Representations (ICLR).

- Boussaid, F., & Chetouani, A. (2019). A comprehensive survey of deep learning for image and speech processing. Journal of Machine Learning Research, 20(1), 1-87.

- Manjunatha, K. N., & Shivaprakash, K. (2018). A survey on recent advances in optical character recognition. International Journal of Advanced Research in Computer and Communication Engineering, 7(9), 482-486.

- Garsia, L. G., & Igolnikov, V. I. (2020). A survey of handwritten character recognition with MNIST and EMNIST. In 2020 International Conference on Artificial Intelligence and Information Technology (ICAIIT) (pp. 211-216). IEEE.

# 8 Appendix

## 8.1 Appendix A

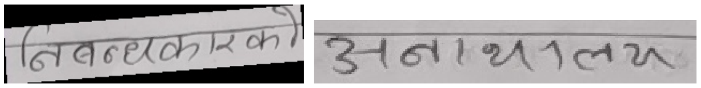


*Figure 24: Word Images*

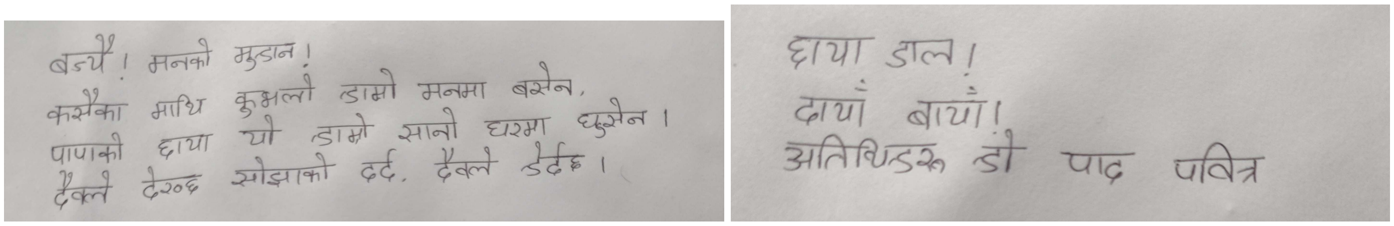
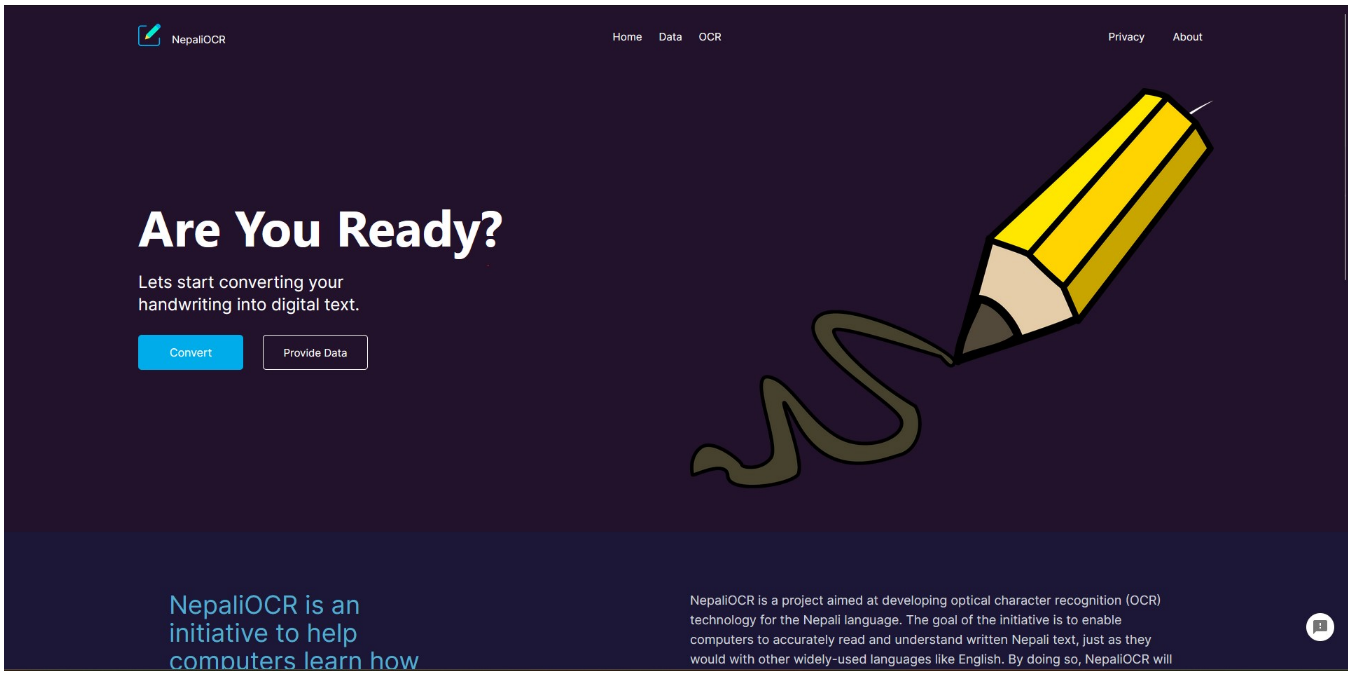

बन्यौं! मनको मुडान!
कसैका माथि कुभलो हाम्रो मनमा बसेन,
पापाको छाया यो हाम्रो सानो घरमा छुसेन।
ढैक्ले ढेरछ झोझाको दर्द. ढैक्ले डर्दछ।
छाया डाल!
दायाँ बायाँ!
अतिथिहरू डो पाद पवित्र

*Figure 25: MultiLine Images*

## 8.2 Appendix B



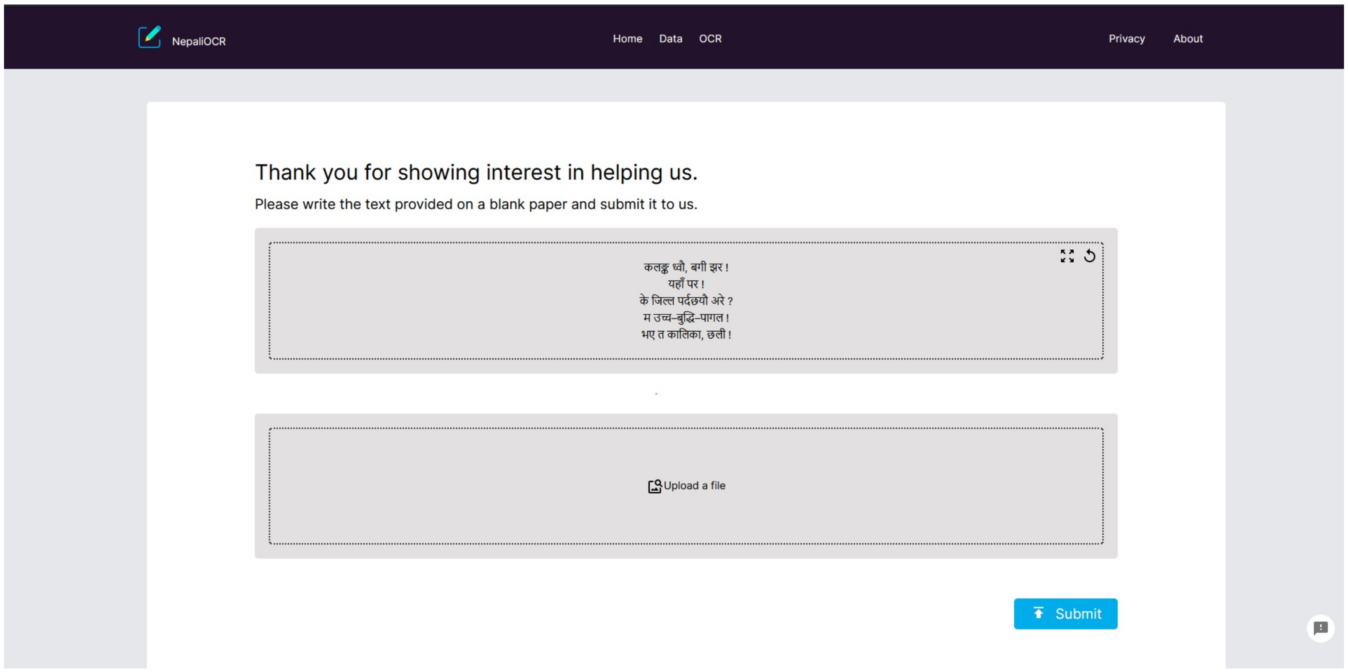
Are You Ready?

Lets start converting your handwriting into digital text.

NepaliOCR is an initiative to help computers learn how

NepaliOCR is a project aimed at developing optical character recognition (OCR) technology for the Nepali language. The goal of the initiative is to enable computers to accurately read and understand written Nepali text, just as they would with other widely-used languages like English. By doing so, NepaliOCR will

*Figure 26: Front Page*

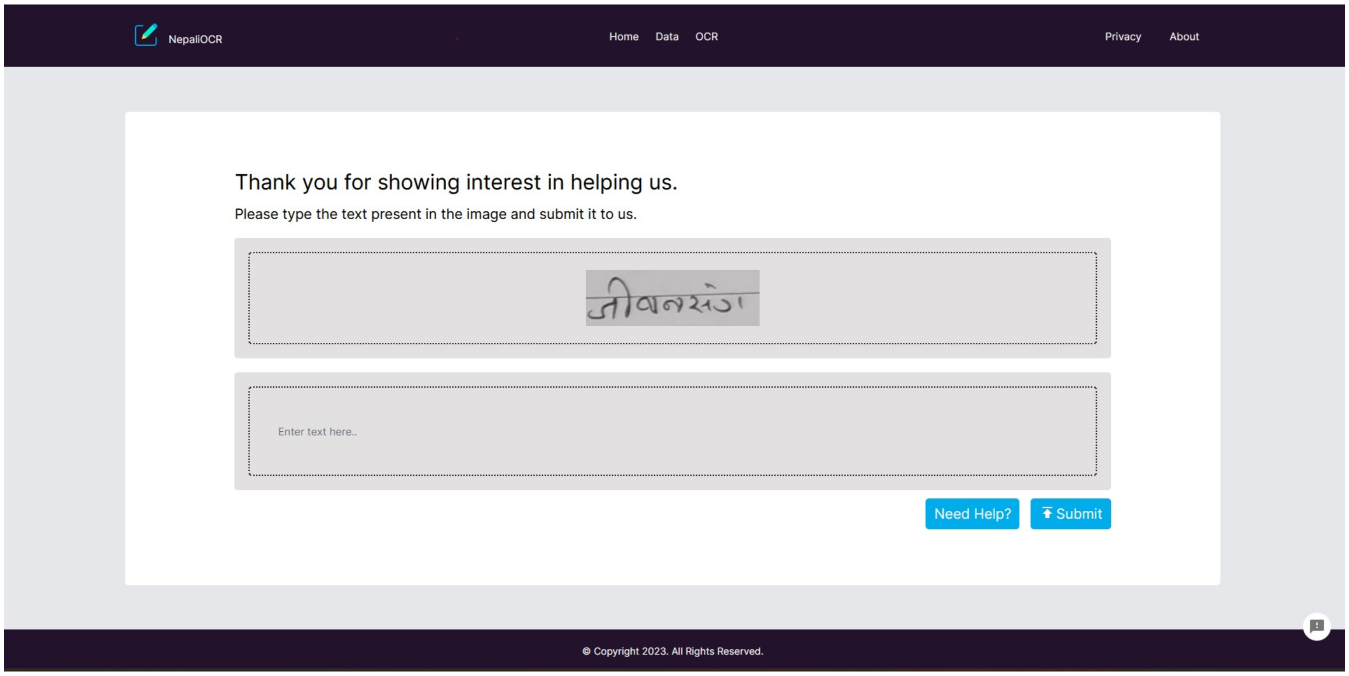


Thank you for showing interest in helping us.

Please write the text provided on a blank paper and submit it to us.

कलङ्क ध्वौ, बगी झर !
यहाँ पर !
के जिल्ल पर्दछ्यौ अरे ?
म उच्च–बुद्धि–पागल !
भए त कालिका, छली !

*Figure 27: Text Generator Page*



Thank you for showing interest in helping us.

Please type the text present in the image and submit it to us.

जीवनसंग।

*Figure 28: Annotation Page*

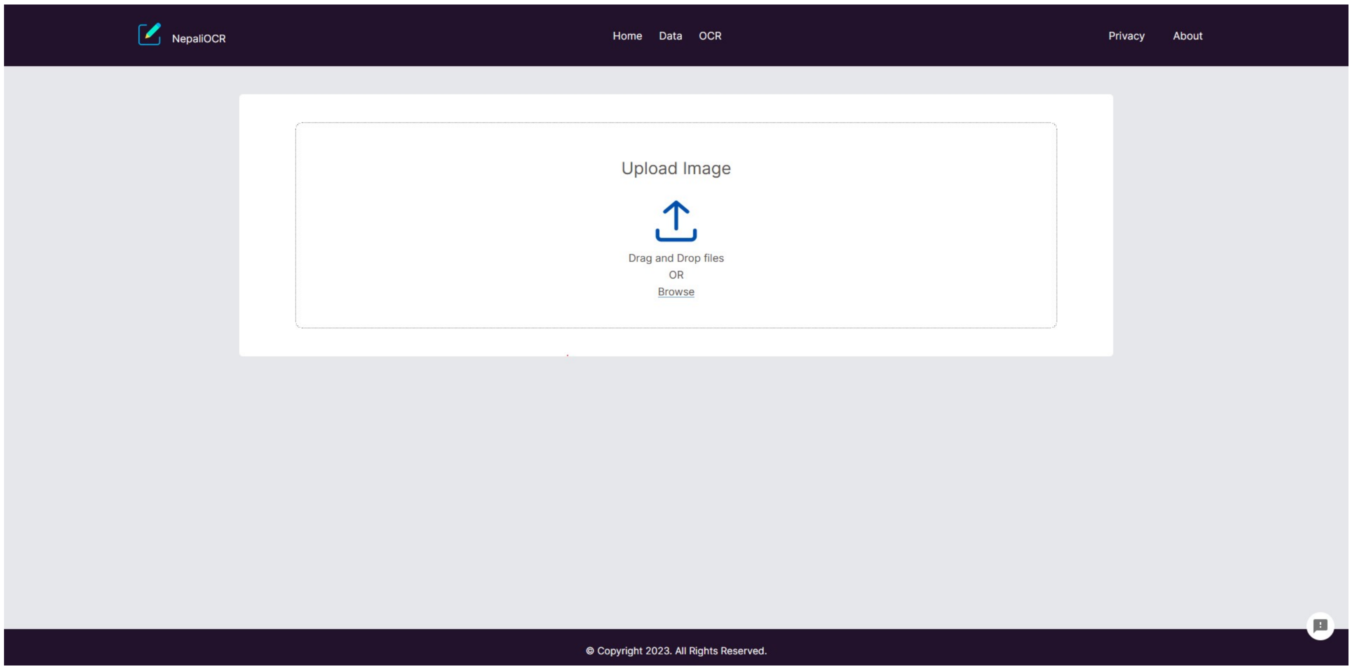


*Figure 29: Before OCR Page*

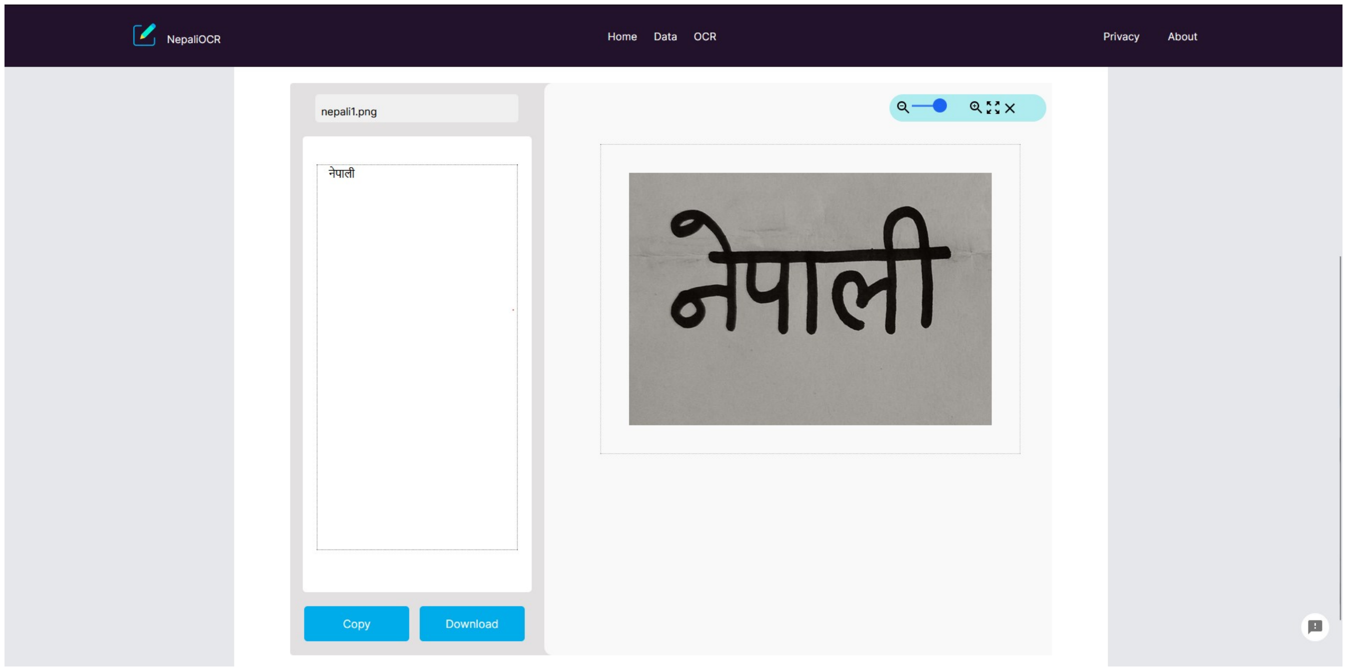
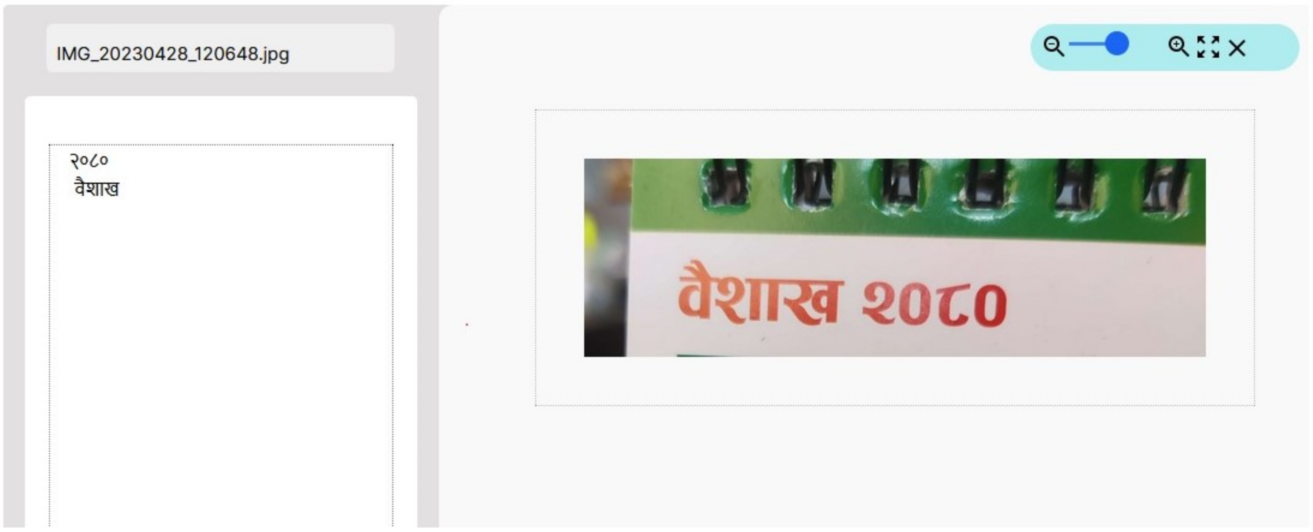


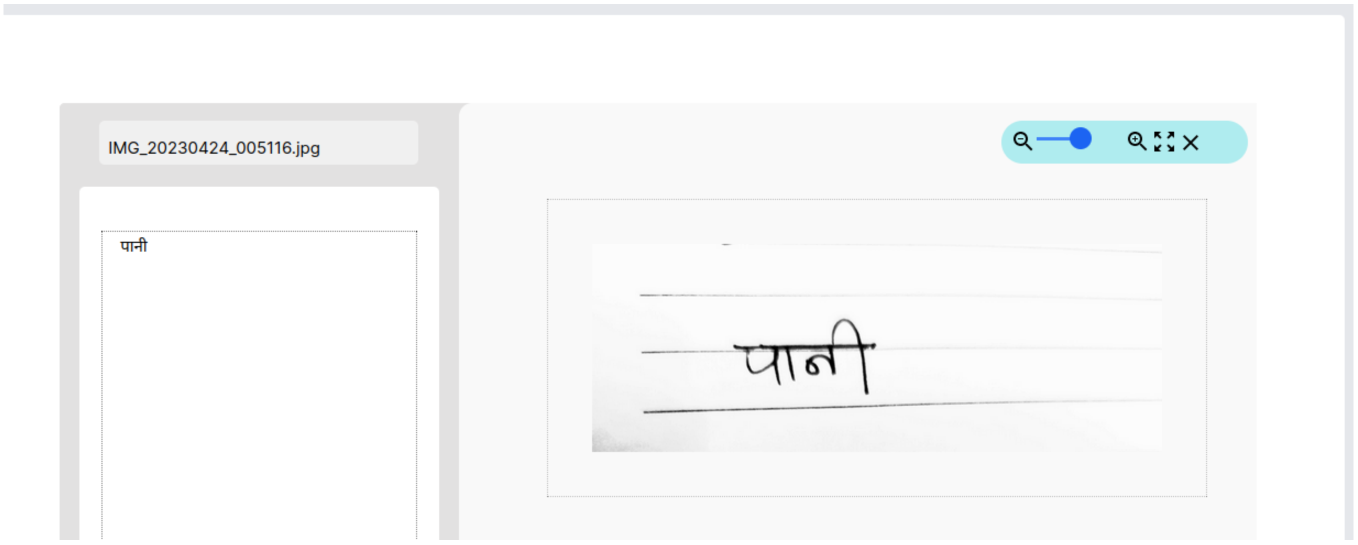
*Figure 30: After OCR Images*

## 8.3 Appendix C


IMG_20230428_120648.jpg

२०८०
वैशाख

Figure 31: Output 1


IMG_20230424_005116.jpg

पानी

Figure 32: Output 2