Kathmandu University
Journal of Science, Engineering and Technology

Comparing OCR Engines for Nepal Lipi Extraction

Pragyan Shrestha, Samriddha Lal Shrestha, Pratik Sharma, Ranjita Dhakal, Prabal Lamichhane, and Rajani Chulyadyo*

Department of Computer Science and Engineering, Kathmandu University,Nepal.

Abstract Nepal Lipi, previously referred to as Prachalit Lipi, the historical script of the Kathmandu Valley, is the carrier of invaluable cultural and historical knowledge stored in ancient manuscripts and artifacts. However, this heritage is largely inaccessible and faces a significant risk of being lost due to the lack of effective digital preservation tools.
Our primary objective was to address lack of a system to systematically evaluate, compare and recognize Lipi text using modern OCR engines and assist in the digitization/preservation of these scripts.
To achieve our goal we decided on using four prominent OCR engines : EasyOCR, PaddleOCR, TesseractOCR, CalamariOCR. To train and validate the models we developed a custom dataset consisting of about 4,625 line level images sourced from historical dataset.
After evaluation there were significant performance differences between the models. CalamariOCR emerged as the top performing model achieving a Character Error Rate (CER) of just 3.755 percent and PaddleOCR as the second most effective model with a CER of 9.06 percent. This study establishes one of the few benchmarks of OCR engines for Nepali Lipi. To ensure practical impact and ease of use the models were integrated into a simple web-app. This provided a tool for preservation and comparison of Prachalit Lipi which serves as a critical tool in this area.

Keywords: OCR, Nepal Lipi, Deep Learning, Character Error Rate, Digitization

1. Introduction

Nepal Lipi (Nepal Script), traditionally used to write Nepal Bhasa, was replaced by Devanagari in modern times. With declining use and the deteriorating condition of old documents, our heritage is at risk. Digitizing such old manuscripts is of historical and cultural importance. Developing a specialized Optical Character Recognition (OCR) model for Nepal Lipi is a step toward its preservation. However, being a low-resource scripts, it is challenging to develop such an OCR model.

Optical Character Recognition is the technique that converts printed or handwritten text to editable text formats. Around the world, OCR is widely used for the purpose of digitizing books, documents, and archival materials. It allows the possibility to search, edit, and analyze texts that would otherwise be inaccessible. In the context of Nepal Lipi, OCR plays a significant role in digitizing documents, allowing better access and study of historical Nepal Bhasa texts.

Several OCR tools are available that claim to support a wide range of scripts. In this research work, we assessed 4 such tools for recognizing Nepal Lipi texts in handwritten manuscripts. We trained four OCR models - EasyOCR [1], PaddleOCR [2], TesseractOCR [3], and CalamariOCR [4] - on a dataset collected from Patan Campus. Among them, CalamariOCR had the best performace with an average Character Error Rate (CER) of 3.755%. We also deployed a web application for text extraction from images of documents written in Nepal Lipi. With the aim to contribute to the preservation of Nepal's cultural heritage, we believe this work can open opportunities in the field of Nepal Lipi OCR.

The paper is organized as follows: Section 1 provides an introduction to the problem domain and outlines the motivation behind the study. Section 2 reviews relevant literature and existing methods related to Nepal Lipi and OCR engines. Section 3 describes the methodology, including data collection and the techniques used for processing and analysis. Section 4 presents the results obtained from experiments and discusses their implications. Section 5 concludes the study with a summary of key findings and outlines potential directions for future work.

*Corresponding author. Email: rajani.chulyadyo@ku.edu.np

2. Related Works

Optical Character Recognition (OCR) has now become a critical application area of machine learning, especially for document digitization, translation, and preservation. The development of deep learning has had a remarkable impact on the improved performance of OCR, especially in low-resource scripts such as Nepal Lipi.

[5] deliver an overview of modern OCR systems, highlighting the evolution from traditional methods to deep learning architectures like Convolutional Neural Networks (CNN), ecurrent Neural Networks (RNN), onvolutional Recurrent Neural Networks (CRNN), and Transformers. These models incorporate components such as Connectionist Temporal Classification (CTC) decoding, facilitating strong text recognition and layout analysis across multiple document formats.

CNN-RNN hybrid, CRNNs have been proven efficient for sequence recognition in handwritten text, as emphasized by [6] and [7]. PaddleOCR has further characterized the efficiency of lightweight CRNN-based frameworks in handling multilingual texts with convolutional layers for feature extraction, BiLSTM (Bidirectional Long Short-Term Memory) for sequential understanding, and CTC for alignment-free decoding. On the other hand, Transformer-based models such as TrOCR [8] surpass traditional architectures at times by leveraging attention mechanisms and large-scale pretraining, often reducing the need for extensive preprocessing.

OCR for low-resource scripts is highly dependent upon high-quality datasets and extensive pre-processing. [9], and [10] pre-

sented the Ranjana Script Handwritten Character Datasets (RHCD), Rañjanā Lipi Dataset (RLD) respectively. Similarly, [11], and [12] published datasets on Nepal Lipi. These datasets were curated through community participation and accurate pre-processing. This helped achieve high accuracy and highlights the importance of script-specific datasets. Similarly, PreP-OCR [13] demonstrates that even advanced OCR systems perform significantly better with image enhancement techniques such as denoising, distortion correction, and basic operations like resizing and binarization.

Fine-tuning and transfer learning have arisen as fundamental strategies for adapting OCR models to low-resource languages. [14] demonstrate how TrOCR can be effectively fine-tuned for scripts like Nepali and Bengali using synthetic data and pre-trained weights.

These past studies highlight the importance of integrating deep architectures with dataset-focused strategies to build an effective OCR system for complex and low-resource scripts.

3. Theoretical Background

3.1. Convolutional Neural Networks (CNNs)

CNNs are neural networks that process input images and extract features like loops, curves, lines, and shapes of characters from the input image with the use of filters. Filters are used for a convolution operation that converts a raw image into a meaningful feature map. This is the first step in an OCR model.

3.2. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNN) RNNs are neural networks that process the data by remembering the past data and utilizing it to recognize future sequences. Long Short-Term Memory (LSTM) makes use of the forget gate and the keep gate to manage past data used in further processing. LSTM processes texts only in the forward direction, while Bidirectional LSTM(BiLSTM) processes texts in both forward and backward directions simultaneously. BiLSTM offers better context and accuracy for that reason.

3.3. Connectionist Temporal Classification (CTC)

CTC is a loss function used in OCR when the alignment between input and output is unknown. It allows models to be trained end-to-end without explicit character-level annotations by introducing a blank token and collapsing repeated predictions. Decoding is done either quickly with greedy search or more accurately with beam search.

3.4. Convolutional Recurrent Neural Networks (CRNNs)

CRNNs are a hybrid architecture that combines CNNs and RNNs. CNNs are used to extract features like loops, curves, lines, and shapes of characters from the input image, and RNNs are used to process the sequence of data. This makes CRNN perfect for tasks that require both spatial and temporal information. It consists of stacked convolutional layers, followed by recurrent layers, which help to achieve accuracy in the real world.

3.5. Character Region Awareness for Text Detection (CRAFT)

CRAFT is a deep learning model used to locate and outline text with images. CRAFT first identifies the individual characters and then determines how they connect to form a word or a sequence of texts. This can be achieved by creating two maps: a region score map that highlights where the characters might be present, and an affinity score map that links the characters that might belong to the same sequence. With the help of these two maps, CRAFT accurately detects texts for the text detection stage in the OCR pipeline.

3.6. Data Augmentation

Data augmentation refers to using pre-processing techniques such as rotation, distortion, and noise. These techniques can be used to manipulate the thickness, height, and strokes of the origins; images to create a diverse dataset that simulates the variations found in real-world images.

3.7. Decoding Strategies

OCR output probabilities must be decoded into character sequences. Greedy decoding selects the most probable character at each timestep and is computationally efficient but less accurate. Beam search decoding considers multiple candidate sequences to find the most likely output at the cost of higher computation.

3.8. Voting Ensembles

Voting Ensemble combines the predictions from multiple models and uses them to make the final prediction. This can be achieved using majority voting or probability scores. This method improves accuracy and reduces error even when handling uncertain or noisy inputs.

3.9. OCR Pipelines and Architectures

EasyOCR: EasyOCR implements a modular OCR pipeline beginning with image preprocessing, like noise reduction, binarization, and skew correction, to enhance image quality. It utilizes the CRAFT model for detecting character regions by generating character region and affinity score maps. For text recognition, EasyOCR employs a CRNN architecture composed of convolutional layers for feature extraction and BiLSTM layers for sequence modeling. The output is decoded using CTC with options for greedy or beam search decoding. Finally, post-processing techniques such as spell correction and text normalization improve the output quality.

TesseractOCR: TesseractOCR version 4 and above uses an LSTM-based deep learning model. Its pipeline includes preprocessing with grayscale conversion, thresholding, and noise correction, followed by layout analysis to segment text blocks, lines, and words. The character recognition stage uses a CNN combined with LSTM layers, decoded with CTC loss. Language modeling with dictionaries and n-grams assists in refining predictions.

CalamariOCR: CalamariOCR is designed for line-level OCR, operating on pre-segmented text lines. It preprocesses images by converting them to grayscale and resizing them to a fixed height. It consists of CNN layers for spatial feature extraction, followed by BiLSTM layers for context modeling, and a CTC decoding layer. During training, it uses the Adam optimizer with gradient clipping for stability. At inference, CalamariOCR supports voting ensembles across multiple trained models to improve prediction reliability and confidence.

PaddleOCR: PaddleOCR's pipeline begins with preprocessing and augmentation using decoding, random augmentations, and resizing. Its core recognition model is a CRNN variant using MobileNetV3 as the CNN backbone for efficient feature extraction, Bi-RNN layers for sequence modeling, and a linear softmax head with CTC loss for training. Optimization uses AdamW with a cosine learning rate scheduler. Decoding employs greedy methods with CTC label decoding. This architecture is well-suited for resource-constrained environments and supports multiple languages, including Nepal Bhasa.

4. Methodology

With the aim of assessing the performance of various OCR models for recognizing texts from documents written in Nepal Lipi, we followed a structured approach starting from data acquisition and

augmentation to choosing and deploying the OCR model. Fig. 1 illustrates the methodology we followed.

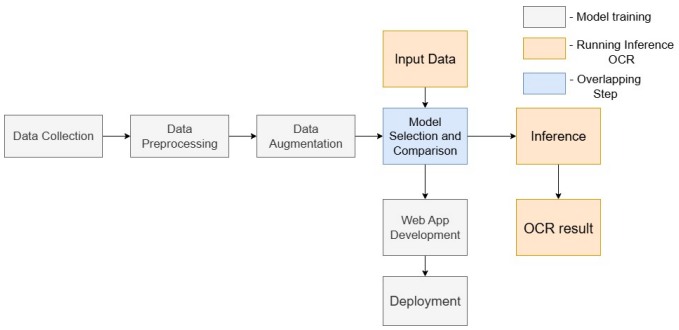

Figure 1: Block diagram illustrating workflow of OCR system.

4.1. Data Acquisition

A data set of 1,156 line segmented images of handwritten manuscripts was sourced from the Department of Nepal Bhasa, Patan Campus. The obtained images had already gone through data preprocessing steps that include binarization, line segmentation, noise reduction, and orientation correction. An example of an image is shown in Fig. 2.

Figure 2: Original image

4.2. Data Preprocessing and Augmentation

In order to enhance model accuracy and generalization as well as to avoid overfitting, data augmentation techniques like erosion, resizing, and dilation were applied, resulting in an increase of the dataset to 4,625 images. Fig. 3 and Fig. 4 show the images after applying erosion, and dilation techniques respectively.

Figure 3: Eroded image

Figure 4: Dilated image

4.3. Model Comparison

The OCR models evaluated in this project were PaddleOCR, CalamariOCR, EasyOCR, and TesseractOCR. Pre-trained models were used for PaddleOCR, CalamariOCR, and TesseractOCR, while EasyOCR was trained from scratch. All of these models were trained on line segmented imgages.

The data set was divided into three sets of train, test and validation with an 80/10/10 ratio. The training set of 3,700 images, testing set of 464 images, and validation set with 463 images were prepared. Two sets with the same split were created to ensure an unbiased treatment of model performance.

The models were trained using the following parameters:

- EasyOCR: 70,000 iterations, Batch Size = 16
- TesseractOCR: 300,000 iterations, Batch Size = 1
- PaddleOCR: 200 epochs, Batch Size = 32
- CalamariOCR: 107 epochs, Batch Size = 16

Although the models were intended to be trained under similar conditions, variations in default batch sizes, training parameters (iterations/epochs), and early stopping resulted in differences.

The input image sizes were:

- EasyOCR, TesseractOCR, CalamariOCR: 32×variable width
- PaddleOCR: 32×360

These models were evaluated based on their average CER performance on the testing set.

4.4. Web App Development

A web app was developed using ReactJS for front-end and Django for back-end. It allows users to upload images, processes them through the OCR model, and returns recognized text, supporting modularity and future scalability.

5. Results and Discussions

In this section we present the results and evaluation metrics for the models. Models were evaluated based on their setup process and results after running inference on the test set data.

5.1. Model Architecture Comparison

The models were initially evaluated on the basis of their ease of setup,drawbacks, models, documentation and support as well as their preferred/supported device. Tesseract only supported training on CPU which made the training lengthy and taxing on the hardware, EasyOCR had lack of proper support for other languages and documentation as well as requiring LMDB dataset to train, while PaddleOCR had its limitation on the recognition of curved text and CalamariOCR being sensitive to noise needing cleaner data.

Table 1: Comparison of OCR Models

Model	Device	Setup	Drawback	Architecture
PaddleOCR	CPU/GPU	Easy, well-documented	Limited with long/curved text	CRNN
TesseractOCR	CPU	Requires external tools	Sensitive to image quality	CNN + LSTM
EasyOCR	CPU/GPU	Needs LMDB; long setup	Few pretrained models	CRNN + CTC
CalamariOCR	CPU/GPU	Simple, but GPU setup tricky	Weak on noisy inputs	CNN + BiLSTM + CTC

5.2. Model Performance

Character Error Rate (CER) was used to compare the model performance. Table 2 and Figure 5 presents the CER observed on the test data set.

Table 2: Average CER (%) of OCR Models

Model	Average CER (%)
CalamariOCR	3.755
PaddleOCR	9.060
EasyOCR	11.039
TesseractOCR	25.516

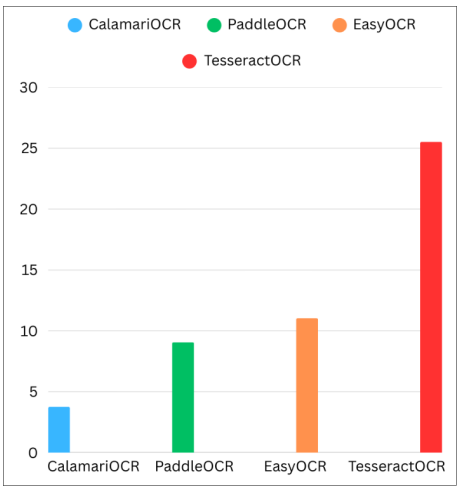

Figure 5: Average CER(%) of OCR Models

CalamariOCR performed the best with an average CER of 3.755%, owing to its CNN + BiLSTM + CTC architecture and ensemble prediction strategy. PaddleOCR followed with a CER of 9.06%, aided by a robust CRNN pipeline. EasyOCR, though capable of flexible input sizes, suffered due to its training complexity and limited pretrained support. TesseractOCR performed worst, struggling with noise and variability in inputs.

PaddleOCR obtained the second-best performance with an average CER of 9.06% over two datasets. Its architecture which is based on CRNN and the comprehensive documentation made it efficient for us to train and deploy. Its fixed input size could have limited it to adapt to variable text lengths causing a slightly higher error rate.

EasyOCR obtained a moderate performance with an average CER of 11.039%. Its flexible input size allowed proper handling of variable text but the requirement of LMDB input format and limited number of pretrained models affected its ease of use and performance. Its architecture combines CRNN, LSTM and CTC which sounds strong in theory but the lack of proper pre-training has impacted its recognition accuracy, especially for augmented samples.

TesseractOCR obtained the highest CER of 25.516% which represents the significant recognition challenges it faced. The model had a hard time recognizing images with noise, inconsistent formatting or even lower contrast. Due to it being dependent on clean, well formatted inputs, it adapted poorly to real world document conditions. TesseractOCR produced blank outputs and incorrect characters when it was dealing with distorted samples leading to a large number of substitution and deletion errors.

5.3. Output Comparsion

Figure 6 is a sample image from a test data on all of the four models as a view point of comparison.

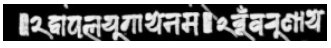

Figure 6: Sample from test data as a view point of comparison

5.3.1. EasyOCR

CER: 11.039%

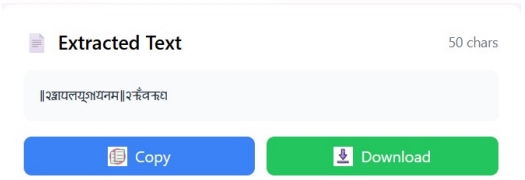

Figure 7: Output from EasyOCR

5.3.2. PaddleOCR

CER: 9.06%

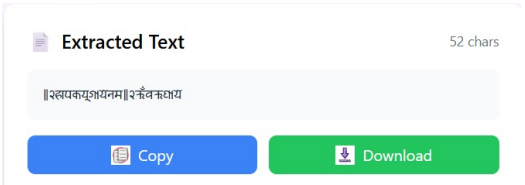

Figure 8: Output from PaddleOCR

5.3.3. CalamariOCR

CER: 3.755%

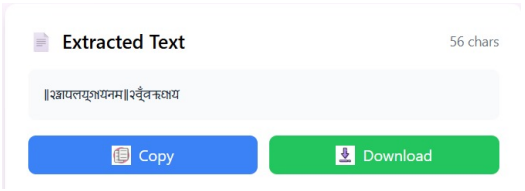

Figure 9: Output from CalamariOCR

5.3.4. TesseractOCR

CER:25.51%

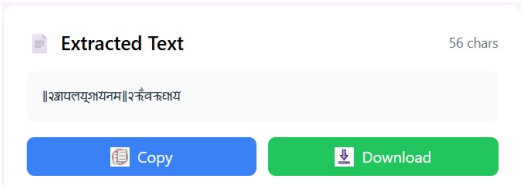

Figure 10: Output from TesseractOCR

5.4. Demonstration

Figure 11 presents the web app interface for our project. We have the ability to choose among the 4 trained models, get an input image preview as shown in Figure 12. Figure 13 illustrates the extracted text from the selected model. We can either download the text file or copy from the webapp

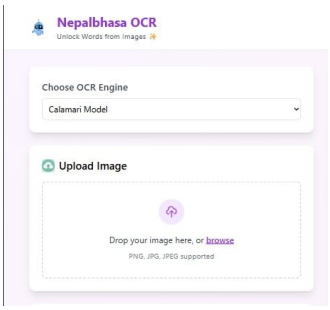

Figure 11: Webapp interface

Figure 12: Image upload portal

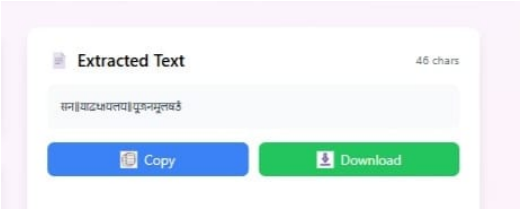

Figure 13: Extracted text displayed in webapp

6. Conclusion

This paper presented the development and evaluation of four OCR engines for the digitization of Nepal Lipi, utilizing a custom dataset. The dataset consisted of line-segmented images of handwritten manuscripts. Among the four engines, CalamariOCR and PaddleOCR demonstrated superior accuracy, achieving Character Error Rate (CER) of 3.755% and 9.06%, respectively.

This project contributed to the digitization of ancient documents written in Nepal Lipi. Furthermore, it lays a strong foundation for future research on Nepal Lipi digitization and script conservation.

Due to the lack of availability of resources, and time. The primary fine tuning was using either pretrained models with weights or by training the model from scratch on the dataset.Further enhancement were based on adjusting the batch size and training iterations or epochs for each model. Some of the model utilized early stopping to stop training beyond a specified threshold. The models were also trained on 3 sets of randomized selection of data from the dataset to ensure variety and variability in the training.

The results of this project were successfully integrated into a web app, allowing users to upload images and receive digitized output in real time.

6.1. Future Work and Enhancement

Although a great deal of progress has been made in this project for Lipi detection, there are still some limitations and further enhancements that can be made.

6.1.1. Short Term Future Work

Some short term goals that we have realized we can achieve in this projects are :

- **Model Refinement and Error Analysis:** Instead of training new architectures, focus on fine-tuning the hyperparameters of your best-performing models (Calamari and PaddleOCR).
- **Enhanced Data Augmentation:** Go beyond the current augmentation techniques. Experiment with more advanced methods like elastic distortions, perspective transforms, and generating synthetic noise that mimics real-world manuscript degradation.

6.1.2. Long Term Future Work

Although the project may seem sufficient for now, there are several ways we can enhance this by introducing larger goals such as these :

- **Custom Model Development:** As mentioned in the literature review, Transformer-based models like TrOCR show great promise. A long-term goal would be to gather a much larger dataset and fine-tune or train a Transformer architecture specifically for Nepal Script.
- **Creating a Public API and Collaborative Platform:** Evolve your web app into a fully-fledged public platform. Develop a public API so that other researchers, digital archives, and libraries in Nepal and around the world can integrate your OCR capabilities into their own systems.
- **Full Document Layout Analysis:** Current models work on pre-segmented lines. A major next step is to develop a system that can take a raw image of an entire manuscript page and automatically perform layout analysis.

References

[1] AI J. EasyOCR. URL https://github.com/JaidedAI/EasyOCR.

[2] Cui C, Sun T, Lin M, Gao T, Zhang Y, Liu J, Wang X, Zhang Z, Zhou C, Liu H et al., Paddleocr 3.0 technical report, *arXiv preprint arXiv:2507.05595*.

[3] Tessearct-OCR. TesseractOCR. URL https://github.com/tesseract-ocr/tesseract.

[4] Wick C, Reul C & Puppe F, Calamari-a high-performance tensorflow-based deep learning package for optical character recognition, *arXiv preprint arXiv:1807.02004*.

[5] Subramani N, Matton A, Greaves M & Lam A. A survey of deep learning approaches for ocr and document understanding (2020). https://doi.org/10.48550/arXiv.2011.13534. URL https://arxiv.org/abs/2011.13534, submitted Nov 27, 2020; revised Feb 4, 2021 (v2); accepted at ML-RSA Workshop, NeurIPS 2020.

[6] Du Y, Li C, Guo R, Yin X, Liu W, Zhou J, Bai Y, Yu Z, Yang Y, Dang Q & Wang H. Pp-ocr: A practical ultra lightweight ocr system (2020). https://doi.org/10.48550/arXiv.2009.09941. URL https://arxiv.org/abs/2009.09941, submitted Sep 21, 2020; revised Oct 15, 2020 (v3).

[7] Acharya R, Building ocr for devanagari handwritten character, *Analytics Vidhya / Medium*. URL https://medium.com/analytics-vidhya/building-ocr-for-devanagari-handritten-character-af5c52de3d23, published Nov 16, 2019.

[8] Li M, Lv T, Chen J, Cui L, Lu Y, Florencio D, Zhang C, Li Z & Wei F. Trocr: Transformer-based optical character recognition with pre-trained models (2021). https://doi.org/10.48550/arXiv.2109.10282. URL https://arxiv.org/abs/2109.10282, last revised Sep 6, 2022.

[9] Bati J & Dawadi P R, Ranjana script handwritten character recognition using cnn, *JOIV: International Journal on Informatics Visualization*, 7(3) (2024) –. https://doi.org/10.30630/joiv.7.3.1725. URL https://joiv.org/index.php/joiv/article/view/1725, accessed June 30, 2025.

[10] Maharjan D. Ranjana lipi (2023). https://doi.org/10.34740/KAGGLE/DSV/4893723. URL https://www.kaggle.com/dsv/4893723.

[11] Nakarmi S, Sthapit S, Shakya A, Chulyadyo R & Bal B K. Nepal script text recognition using CRNN CTC architecture. In: Melero M, Sakti S & Soria C, eds., *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024*. ELRA and ICCL, Torino, Italia (2024), pp. 244–251. URL https://aclanthology.org/2024.sigul-1.29/.

[12] Shakya S, Nepal S, Neupane R, Gupta S R & Shakya H, Reviving indigenous script: Optical character recognition for the prachalit newa script.

[13] Guan S, Lin M, Xu C, Liu X, Zhao J, Fan J, Xu Q & Greene D. Prep-ocr: A complete pipeline for document image restoration and enhanced ocr accuracy (2025). https://doi.org/10.48550/arXiv.2505.20429. URL https://arxiv.org/abs/2505.20429, submitted May 26, 2025; also available as version 2, May 28, 2025.

[14] Hasan S M R, Dhakal A, Mehedi M H K & Rasel A A. Optical text recognition in nepali and bengali: A transformer-based approach (2024). https://doi.org/10.48550/arXiv.2404.02375. URL https://arxiv.org/abs/2404.02375, submitted Apr 3, 2024; Accepted and presented at ICAECC 2023, Bengaluru, India.