BTech Major Project

An Approach for Line Detection in Indic Manuscripts for OCR Applications

Instructor:

Dr. Anand Misra

Report Submitted by:

Prachi Katare (B21CS068)

Yogesh Jangir (B21CS083)

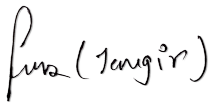
Indian Institute of Technology Jodhpur

Computer Science and Engineering

Dec, 2024

Declaration

I, Yogesh Jangir (B21CS083), at this moment, declare that the Project Report entitled "An Approach for Line Detection in Indic Manuscripts for OCR Applications" done by me under the guidance of Dr. Anand Misra, at the Indian Institute of Technology Jodhpur, is submitted in partial fulfillment of the requirements for the award of the Bachelor of Technology degree in Computer Science and Engineering.



Signature

Yogesh Jangir

B21CS083

Acknowledgments

I am writing to express my heartfelt gratitude to my supervisor **Dr. Anand Misra** for his invaluable guidance, constant support, and encouragement throughout this project. His expertise and insights have been pivotal to the successful completion of this work.

I extend my special thanks to **Katikapalli Lokesh**, Research Analyst, and **Anik De**, for their valuable suggestions, collaboration, and assistance during the course of this project. Their input has greatly enriched the quality of this work.

My deepest gratitude goes to my project members for their continuous support and camaraderie, which made this journey enjoyable and rewarding.

Table of Contents

1. Abstract

Line detection in historical Indic manuscripts presents significant challenges due to varying text sizes, complex layouts, and noisy backgrounds. Traditional segmentation methods often struggle to handle these issues effectively, particularly in noisy and unstructured manuscript images. This project introduces YOLOv11, a deep learning-based approach, for precise line detection in Indic manuscripts, focusing on improving text segmentation in complex scenarios.

The approach involves training YOLOv11 on manuscript datasets, including a small Sanskrit dataset with 28 images and larger datasets containing Khmer, Sundanese, and Balinese manuscripts. YOLOv11, known for its ability to detect features at multiple scales, was fine-tuned to accurately identify and segment text lines in these diverse manuscripts. The model was trained to handle challenges such as varying text formats, multi-line layouts, and noisy conditions that are common in historical manuscript images.

Experimental results demonstrate that YOLOv11 significantly outperforms traditional line detection methods, providing high accuracy and robustness in detecting text lines. The proposed approach shows great potential for enhancing the automatic digitization of Indic manuscripts, facilitating better transcription, preservation, and analysis of historical texts.

2. Introduction

Historical manuscripts hold immense cultural, linguistic, and scholarly value, but their digitization presents significant challenges. Ancient Indic manuscripts, in particular, are characterized by complex layouts, noisy backgrounds, and varied text formats. Preserving and analyzing these manuscripts requires efficient text segmentation methods, which can extract meaningful information from unstructured and degraded images.

Traditional Optical Character Recognition (OCR) techniques and line detection methods often struggle with such manuscripts due to issues like inconsistent text alignment, overlapping characters, and the presence of diacritics. These limitations underscore the need for advanced approaches that can handle the unique challenges of historical texts.

This project explores modern deep learning techniques, particularly YOLOv11, for line and word detection in Indic manuscripts. The goal is to enhance text segmentation, enabling better transcription and preservation of historical documents. By focusing on datasets of Sanskrit, Khmer, Sundanese, and Balinese manuscripts, this work aims to create scalable and robust solutions for digitizing and analyzing ancient texts.

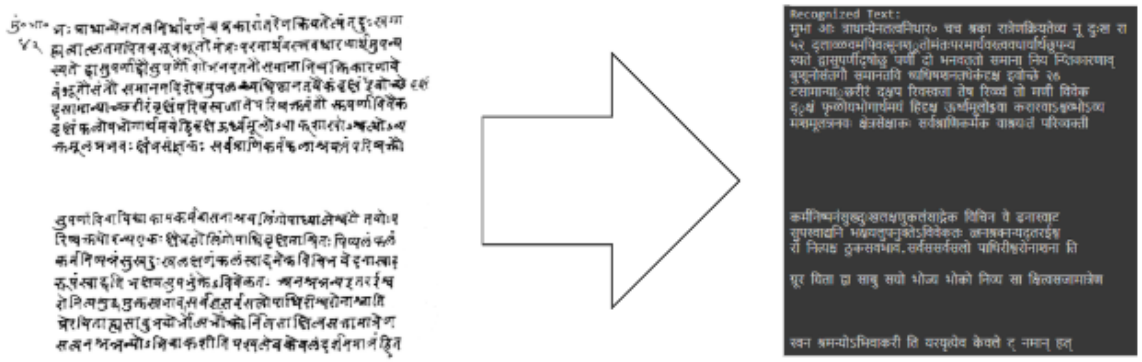


Figure 1: Extracting Text from Clean Indic Manuscript Using PyTesseract

2.1 Problem Statement

The digitization of historical Indic manuscripts poses significant challenges due to the unique characteristics of these texts. These manuscripts are often handwritten on palm leaves or other traditional materials, leading to natural aging and degradation over time. This results in noisy images with faded ink, uneven lighting, and damaged sections. Furthermore, the complex scripts

used in these manuscripts, such as those in Balinese and Khmer texts, include intricate diacritics and symbols, adding another layer of complexity. The manuscripts also exhibit irregular layouts, such as multi-line or double-columned text, with occasional illustrations interspersed. Traditional Optical Character Recognition (OCR) and line detection methods struggle to handle such variability, often failing to achieve the precision needed for accurate text recognition and segmentation. These challenges highlight the need for more advanced methods to ensure the effective digitization and preservation of these invaluable historical documents.

2.2 Objectives

This project aims to address the limitations of traditional approaches by introducing advanced methods for text segmentation in historical manuscripts. Specifically, it seeks to develop a robust solution for line detection in noisy and complex manuscript images, leveraging deep learning techniques. The objectives include improving word-level segmentation to facilitate precise transcription, experimenting with both small and large datasets for model training and validation, and benchmarking the performance of deep learning models like YOLOv11 against traditional line detection techniques such as OpenCV-based methods. By achieving these objectives, the project will contribute to the broader goal of creating scalable solutions for the digitization of diverse historical texts.

2.3 Scope

The scope of this project spans various tasks, datasets, and techniques, all aimed at enhancing the digitization of Indic manuscripts. It involves working with a small Sanskrit dataset of 28 images as a preliminary testbed and extending the approach to larger datasets, including Khmer, Sundanese, and Balinese manuscripts. These datasets offer significant diversity in terms of text formats, line arrangements, and page structures, making them ideal for evaluating the robustness of the proposed methods. The project employs a combination of traditional image processing techniques and modern deep learning methods, particularly YOLOv11, to tackle the challenges of line and word detection. While the primary focus is on Indic manuscripts, the methodologies developed could be applied to other historical texts worldwide, broadening the project's impact.

3. Background and Related Work

The digitization of historical Indic manuscripts is a challenging task due to the intricate scripts, degraded physical conditions, and complex layouts of these documents. Traditional segmentation methods often struggle to process the noisy, curved, or overlapping text lines typically found in palm-leaf manuscripts. This has driven researchers to explore a wide range of techniques, from classical image processing to modern deep learning approaches, to improve text line and word segmentation. While many methods have been applied to other historical documents, palm leaf manuscripts present unique challenges, such as frequent diacritics, irregular text layouts, and the need for high precision in segmenting closely spaced lines.

3.1 OCR Techniques

Optical Character Recognition (OCR) has been fundamental in the digitization of historical texts, including Indic manuscripts. Classical OCR tools such as PyTesseract and EasyOCR utilize rule-based algorithms for recognizing characters from scanned images. While effective for clean and structured documents, these tools struggle when applied to noisy, degraded, or complex manuscript layouts. To improve performance, preprocessing steps like noise removal, binarization, and contrast enhancement are commonly used. Template-based OCR methods further aid segmentation by matching known patterns to text templates. Keras OCR provides a deep learning-based solution that is more adept at handling complex text layouts. However, all these techniques face limitations when dealing with heavily degraded images or non-standard scripts.

As shown in **Figure 1,** we applied PyTesseract OCR to a clean Indic manuscript image. Despite the image's clarity, the output was poor, highlighting the limitations of inbuilt OCR methods for accurate text extraction from historical documents.

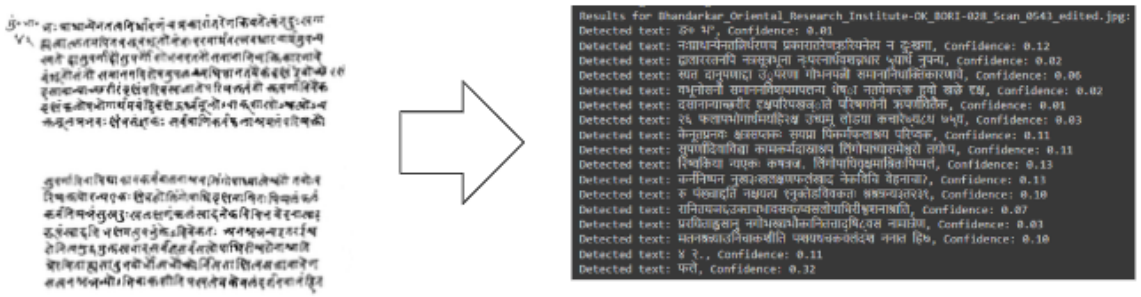


Figure 2: Extracting Text from Clean Indic Manuscript Using EasyOCR

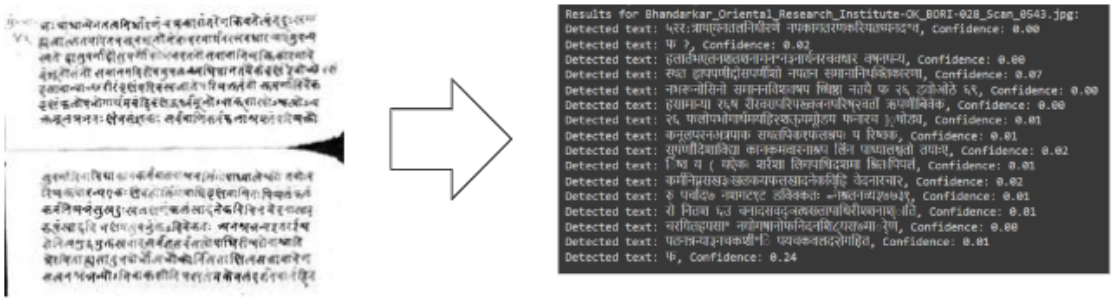


Figure 3: Extracting Text from Noisy Indic Manuscript Using EasyOCR

Figure 2 demonstrates text extraction using EasyOCR on a clean Indic manuscript, revealing improved performance compared to PyTesseract, but still showing signs of limitations in accurately recognizing all characters.

Figure 3 The confidence scores, as seen in the figures, indicate that despite preprocessing efforts, these inbuilt OCR tools often struggle with maintaining high accuracy across different types of manuscript quality. This highlights the need for more robust and specialized approaches for effective text detection and segmentation in diverse manuscript types.

3.2 Line Detection in Manuscripts

Line detection is a critical step in processing historical manuscripts. Early methods relied on classical image processing techniques, such as Canny edge detection and Hough transform, to identify line structures. These approaches, while effective for simple layouts, struggled with the curved and irregular lines found in palm-leaf manuscripts. Template matching, an extension of OCR, has been explored to detect predefined structures in these documents, offering some success in identifying text lines and regions.

More recently, deep learning-based approaches, such as YOLO (You Only Look Once), have demonstrated significant improvements in line and word detection by leveraging convolutional neural networks for feature extraction and multi-scale detection. In this project, YOLOv11 is employed for line detection. Initially, a small Sanskrit dataset is used for fine-tuning the model, followed by testing on larger and more diverse datasets, including Khmer, Sundanese, and Balinese manuscripts. The model's ability to handle noisy backgrounds, overlapping text, and varying layouts makes it particularly suitable for these challenging datasets.

4. Methodology

4.1 Proposed Approach

The project adopted a two-pronged approach to achieve precise line detection. First, traditional OpenCV image processing methods were employed to extract text lines based on edge and line detection algorithms. Although effective for clean images, these methods needed to work on noisy or complex layouts. To overcome these challenges, YOLO-based deep learning models were used. Specifically, YOLOvNAS and YOLOv11 were trained on annotated manuscript datasets using Roboflow, a platform that simplifies dataset preparation and model training. YOLOvNAS was selected for its computational efficiency, making it suitable for real-time applications. YOLOv11, on the other hand, was chosen for its advanced feature extraction capabilities, which allowed it to handle complex layouts and noisy backgrounds effectively. This dual-model approach ensured high precision and robustness across diverse manuscript datasets.

4.2 Tools and Dataset

Tools:

- **Roboflow:** Used for annotating, augmenting, and managing datasets, as well as for training YOLO models.
- **OpenCV:** Applied for traditional edge detection and Hough transform-based line segmentation.
- **YOLO Framework:** Integrated YOLOvNAS and YOLOv11 for object detection.
- **Google Colab:** Provided the computational environment for training and fine-tuning models.
- **LabelImg**: A tool for annotating manuscript images, and creating bounding boxes for text lines.

Dataset:

1. **Sanskrit Dataset**: The Sanskrit dataset consisted of 55 manuscript images, divided into 39 for training, 11 for validation, and 5 for testing. This dataset, characterized by complex layouts and varying text formats, was used to fine-tune and evaluate the proposed line detection models.
2. **Balinese Dataset**: The Balinese dataset consists of 393 pages of palm leaf manuscripts, extracted from the AMADI LontarSet. These manuscripts, primarily consisting of 4 text

lines per page, are often double-columned and contain occasional illustrations. One of the unique characteristics of these manuscripts is the variety of diacritics used in the text. For this project, the challenge dataset includes 96 pages, with 47 pages used for training and 49 for testing. The average size of each manuscript page in this dataset is 500 × 5000 pixels, showcasing the challenge posed by the large, complex layout of these texts.

3. **Khmer Dataset**: The Khmer dataset consists of 657 pages of palm leaf manuscripts, extracted from the SleukRith Set, representing Cambodian historical texts. These pages typically contain 5 text lines, and the challenge dataset includes 250 pages, with 50 pages for training and 200 pages for testing. The average size of each manuscript page in this dataset is 500 × 5500 pixels. Like the Balinese dataset, the Khmer manuscripts feature complex layouts, with varying text arrangements and noisy backgrounds, making them an ideal test case for the robustness of the YOLOv11 model.
4. **Sundanese Dataset**: The Sundanese dataset includes 66 pages of Lontar manuscripts, randomly selected from 27 collections. On average, each page contains 4 text lines. The challenge dataset consists of 61 pages, with 31 pages used for training and 30 for testing. The average size of each manuscript page in this dataset is 350 × 3000 pixels. This dataset, like the others, presents challenges such as irregular text layouts and diacritics, providing a valuable test set for evaluating the performance of line detection methods across different types of Indic manuscripts.

Figure 4: Sudanese Manuscript (Extreme Document Degradations, first), Khmer Manuscript (Poor Text Contrast - Low Ink , second, third), Balinese Manuscript (Poor Document Contrast, fourth)

4.3 Workflow in YOLO

The workflow for training a YOLO model for line detection in Indic manuscripts follows a structured pipeline to ensure that the data is properly collected, annotated, preprocessed, and then used to train the model. Each step is crucial for achieving high accuracy in detecting text lines in the noisy and complex layouts of the manuscripts.

- Data Collection: The first step is to gather high-quality manuscript images from datasets like Balinese, Khmer, and Sundanese. These images should represent diverse text layouts, including different line structures and diacritical marks.
- Data Annotation Using LabelImg: Next, data annotation is performed using LabelImg, a tool for drawing bounding boxes around text lines in the images. This helps YOLO learn where text lines are located. The annotations are saved in a format compatible with YOLO.

- Preprocessing: The collected and annotated images undergo preprocessing:
    - Auto-Orientation: Ensures the images are correctly aligned, especially for manuscripts with varying orientations.
    - Resize to 640x640: All images are resized to 640x640 pixels to meet YOLO's input size requirements.
- Training the YOLO Model: Once preprocessed, the dataset is used to train the YOLOv11 model. During training, the model learns to predict text lines by minimizing the error between predicted and actual bounding boxes.
- Evaluation and Testing: After training, the model is evaluated using a validation set and tested on new data to ensure it can generalize well to unseen manuscripts.
- Post-Processing: Finally, post-processing techniques refine the model's predictions, such as filtering out false positives and adjusting bounding boxes, before visualizing the detected text lines.

5 Implementation

5.1 Line Detection Using OpenCV

The first approach involves leveraging OpenCV's image processing capabilities to detect text lines in manuscript images. The process begins with the loading of manuscript images in grayscale, a step aimed at reducing computational complexity and focusing on textual features. Preprocessing techniques, such as Gaussian blurring and adaptive thresholding, enhance the text regions by minimizing noise and improving contrast. Edge detection is performed using the Canny edge detector, which highlights the contours of the text lines. To identify the structural arrangement of these lines, the Hough Transform is applied, mapping the detected edges into lines and curves in the Hough space. The centroids of these line segments are calculated and classified to group overlapping or closely spaced lines into coherent units. Finally, these grouped centroids are segmented into individual text lines for further processing. While the OpenCV-based method provides reasonable results for clean manuscript images, its limitations become apparent in noisy or degraded conditions. Figures illustrating these results demonstrate that the method struggles with overlapping characters and curved text lines, often resulting in inaccurate or incomplete segmentation.

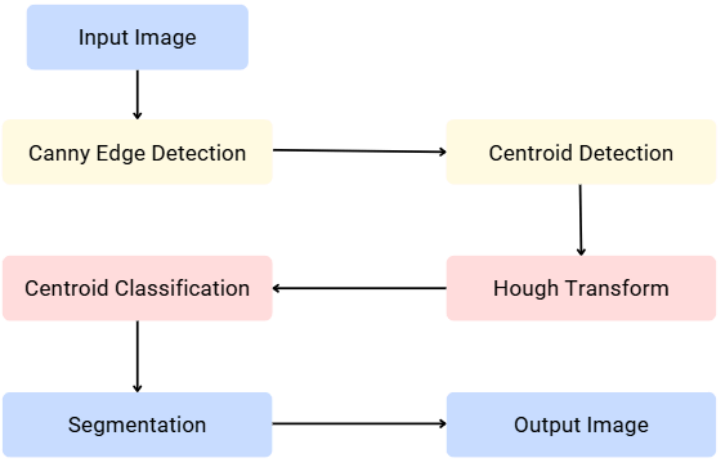


Figure 5: Architecture Follow for line detection using OpenCV

5.2 Line Detection Using Yolo

In this approach, both YOLOv11 and Neural Architecture Search (NAS) are employed to improve the performance of line detection in manuscript images. YOLOv11 is used as the primary object

detection framework, while NAS helps in optimizing the network architecture for improved accuracy and efficiency in detecting text lines.

YOLOv11 for Line Detection

As previously mentioned, YOLOv11 is an object detection algorithm capable of real-time detection of objects by framing the task as a classification problem. In this approach, YOLOv11 is used to detect text lines in the manuscript images. The process begins by manually annotating the dataset using tools like LabelImg, where bounding boxes are drawn around the text lines. The images are preprocessed by auto-orienting them and resizing them to 640x640 pixels, which ensures the consistency of input data for the model.

During training, YOLOv11 learns to predict the bounding box coordinates and the class of each detected object. For line detection, the object class is "text line," and the model learns to predict the position and size of these lines within the manuscript images. This approach is particularly robust in detecting text lines in challenging conditions, such as noisy images, overlapping characters, or curved text lines, where traditional methods like OpenCV struggle.

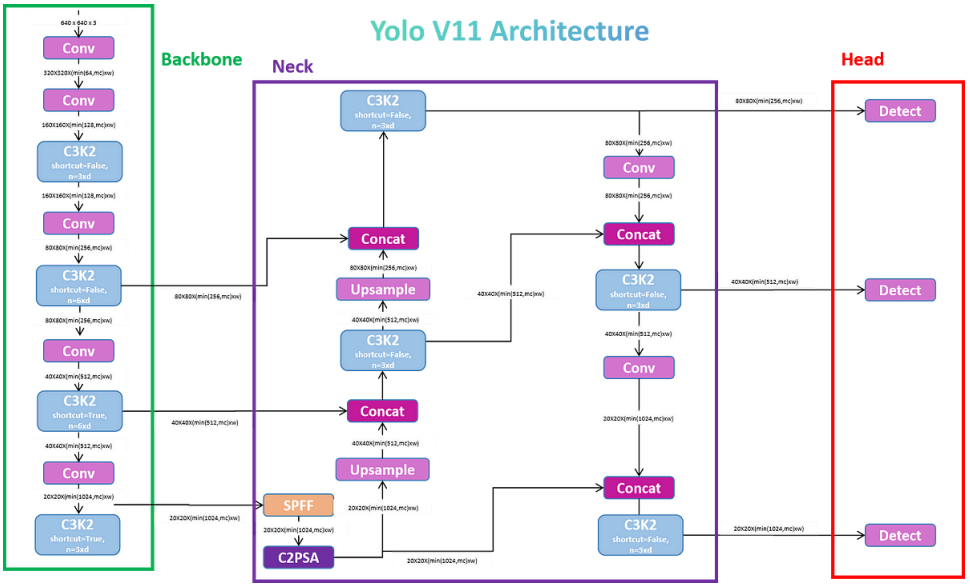


Figure 6: YoloV11 Architecture

NAS (Neural Architecture Search) for Model Optimization

Neural Architecture Search (NAS) is a powerful technique used to automate the design of deep learning models. In this work, NAS is applied to optimize the architecture of the YOLOv11 model

specifically for the task of text line detection. NAS aims to find the best combination of layers, activations, and hyperparameters that improve the performance of the model on the specific task. By leveraging NAS, the YOLOv11 model can be fine-tuned to achieve higher accuracy, faster convergence, and better generalization for line detection. The search process involves training multiple candidate models with different architectures and evaluating their performance on the training set. The best-performing architecture is then selected for final training and testing. NAS allows for a more tailored architecture that is better suited for the unique challenges of manuscript image processing, such as varying text line orientations, varying image quality, and the presence of background noise.

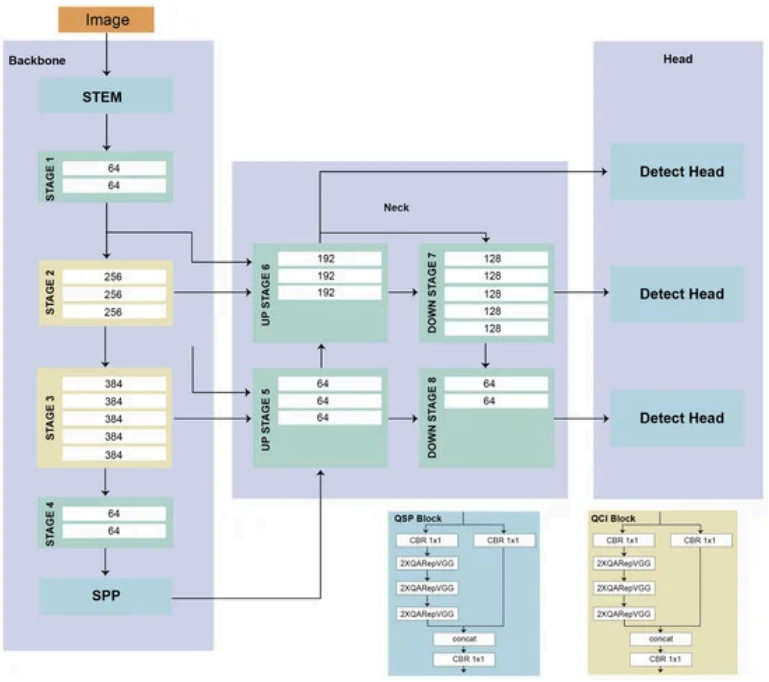


Figure 7: YoloNAS Architecture

6. Results and Discussion

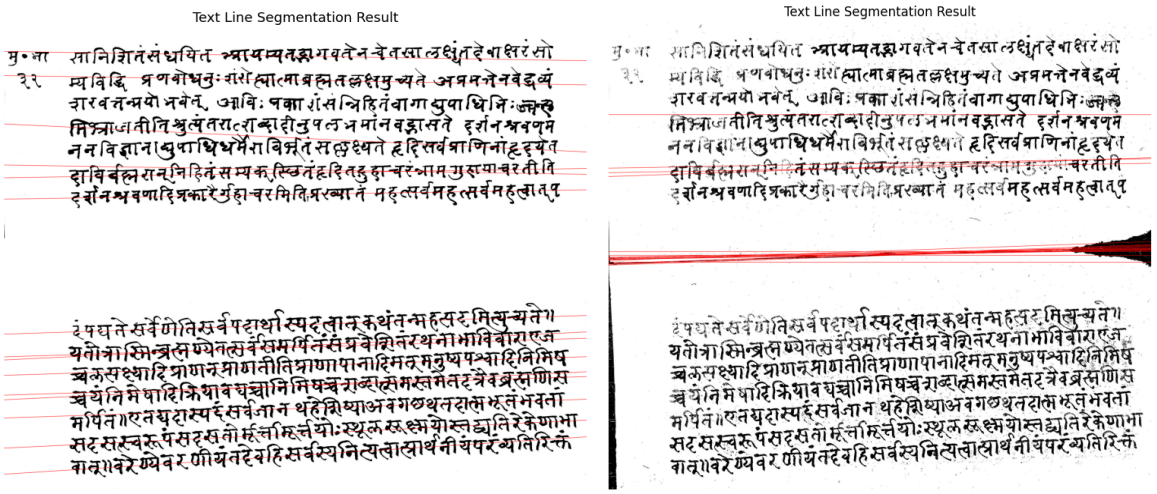


Figure 5: Line Detection using OpenCV on clear image v/s noisy image

The images compare text line segmentation on clear and noisy text images using OpenCV. Segmentation works well on the clear image, with accurate alignment to text rows, but degrades on the noisy image due to interference from artifacts. Preprocessing techniques like noise reduction and adaptive thresholding, or advanced OCR models, can significantly improve segmentation accuracy in noisy conditions.

	Precision	Recall
Khmer	0.979	0.974
Sundanese	0.986	0.957
Balinese	0.995	0.916

Table 1: Evaluation Matrix of YoloV11

	Precision	Recall
Khmer	0.992	0.975
Sundanese	0.958	1
Balinese	1	0.952

Table 1: Evaluation Matrix of YoloNAS

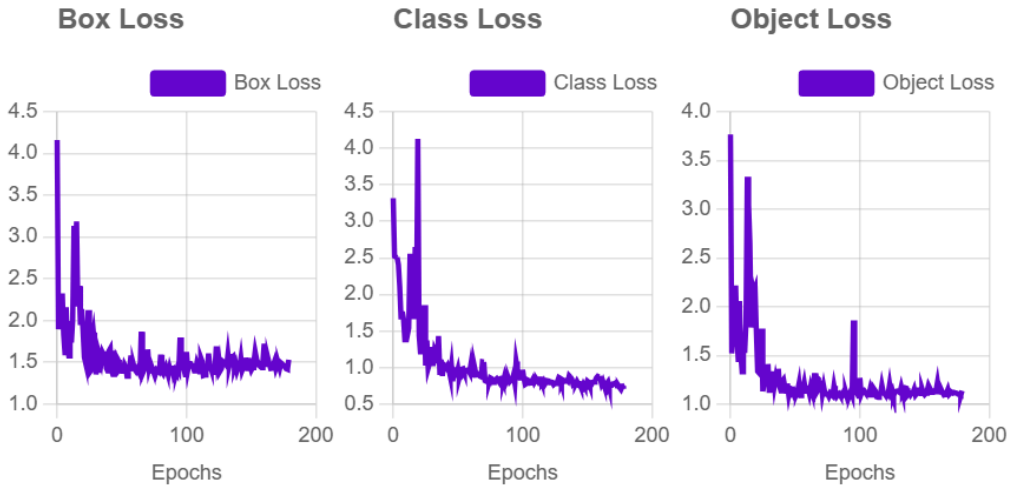


Figure 6: Training Graph of Khmer dataset using YoloV11

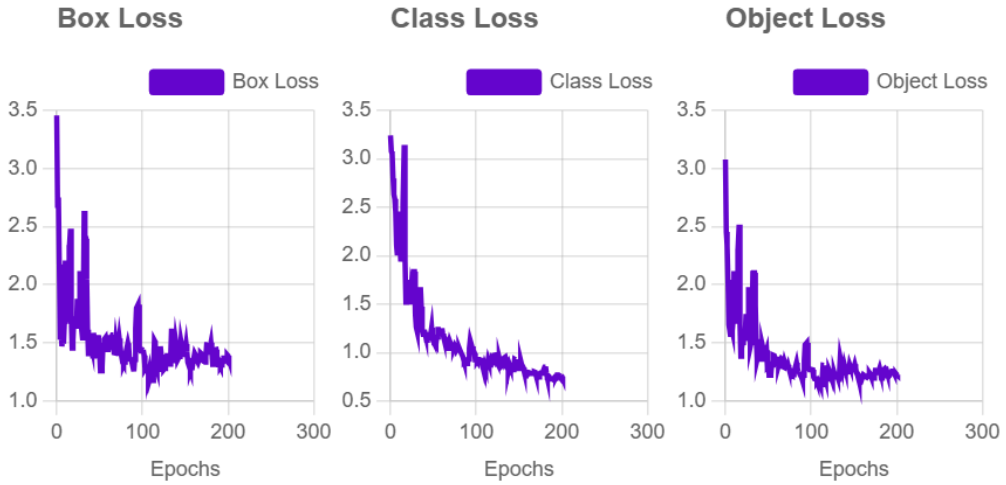


Figure 7: Training Graph of Balinese dataset using YoloV11

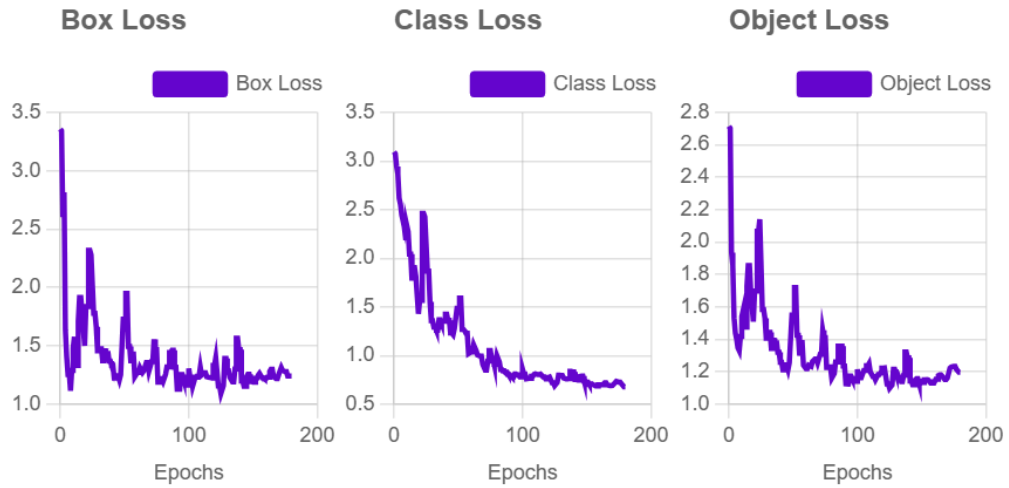


Figure 8: Training Graph of Khmer dataset using YoloV11

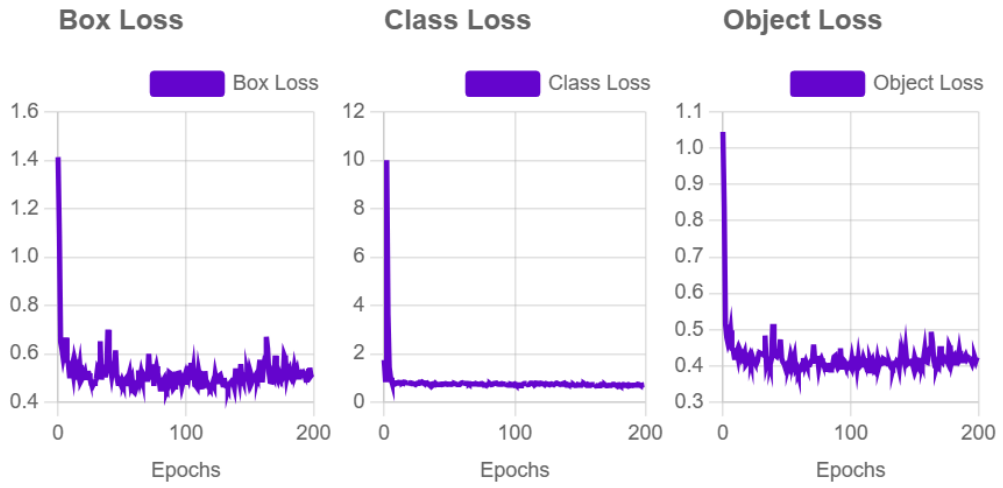


Figure 9: Training Graph of Khmer dataset using YoloNAS

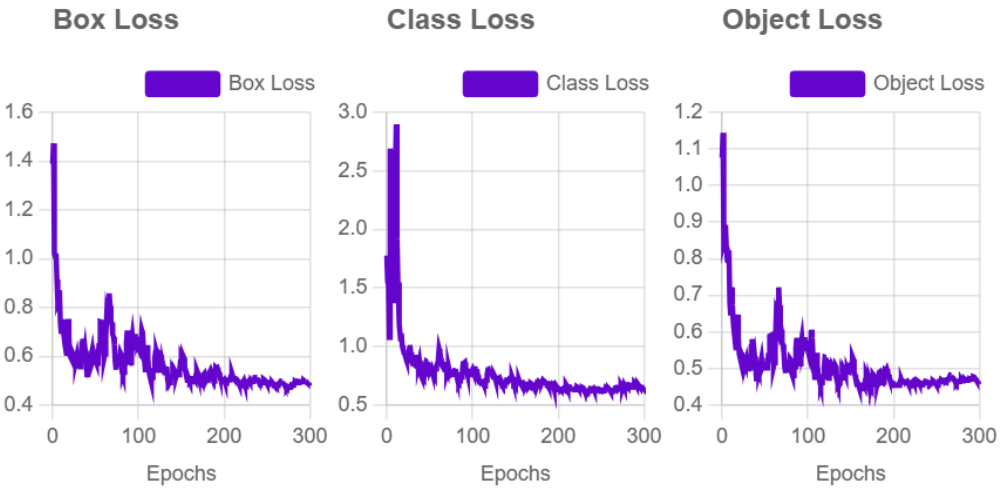


Figure 10: Training Graph of Balinese dataset using YoloNAS

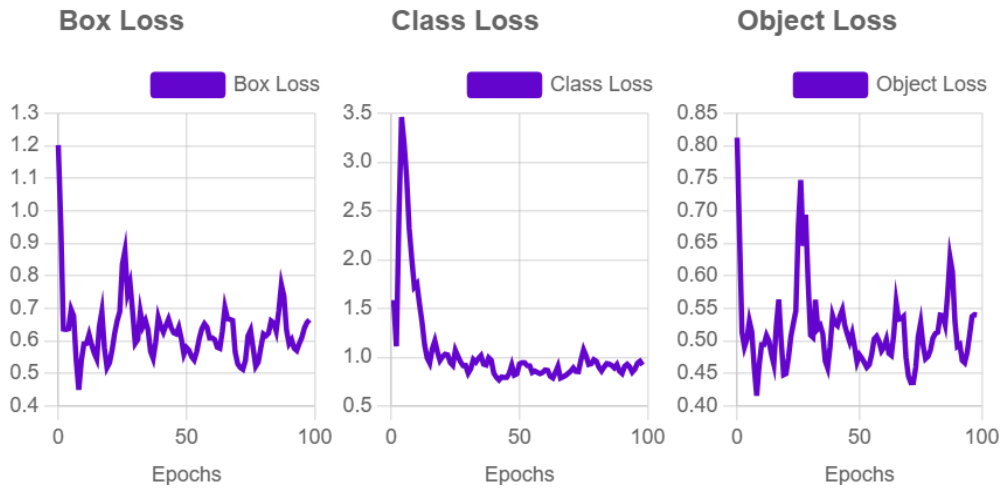


Figure 11: Training Graph of **Sundanese** dataset using YoloNAS

	IoU (↑)	HD (↓)	AvgHD (↓)	HD95 (↓)
Khmer	0.96	2.67	2.35	530.31
Sundanese	0.9667	2.66	2.35	617.86
Balinese	0.97	2.67	2.33	532.62

Table 3: YoloV11 Performance Metrics on Different Datasets

	IoU (↑)	HD (↓)	AvgHD (↓)	HD95 (↓)
Khmer	0.95	2.67	2.356	530.29
Sundanese	0.9667	2.66	2.35	617.86
Balinese	0.98	2.65	2.34	532.64

Table 4: YoloNAS Performance Metrics on Different Datasets

When evaluating the performance of YoloV11 and YoloNAS for line detection across datasets, YoloNAS consistently outperformed YoloV11. YoloNAS achieved higher precision and recall across Khmer, Sundanese, and Balinese datasets, with particularly strong results for Khmer and Balinese. For example, YoloNAS achieved perfect precision (1.0) for Balinese and Sundanese datasets, while YoloV11 exhibited slightly lower recall for these datasets, indicating its limitations in handling challenging text patterns. Metrics such as Intersection over Union (IoU) and Hausdorff distances (HD, AvgHD, and HD95) further validated YoloNAS's superior line localization capabilities. YoloNAS achieved higher IoU scores and consistently lower Hausdorff distances, highlighting its ability to detect and segment text lines with greater accuracy. The training graphs for both models revealed that YoloNAS had faster convergence and exhibited greater stability during training compared to YoloV11. This indicates that YoloNAS's architecture is better suited for learning diverse and complex text patterns across multilingual datasets. YoloV11, while effective for certain datasets, required more training epochs to stabilize and occasionally struggled with datasets that had noisy or overlapping lines. Despite its superior performance, YoloNAS has areas for potential improvement. Fine-tuning the model for specific datasets, especially to enhance recall for datasets with complex layouts like Khmer, could lead to even better results. Preprocessing techniques such as noise filtering, background normalization, and adaptive binarization could further enhance input quality and improve segmentation accuracy for

noisy images. Additionally, exploring ensemble methods or integrating attention mechanisms could help refine line detection for densely packed or overlapping text.

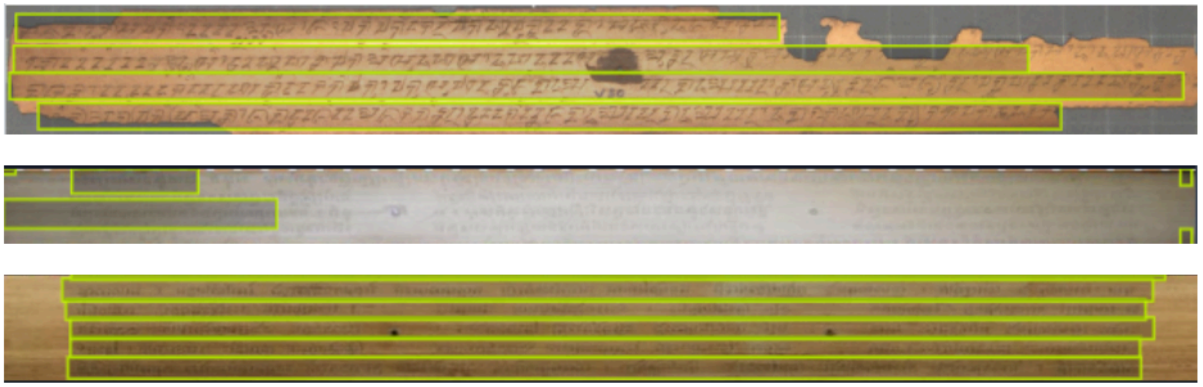


Figure 12: Results of (Sundanese, Balinese, and Khmer respectively)YoloV11 model

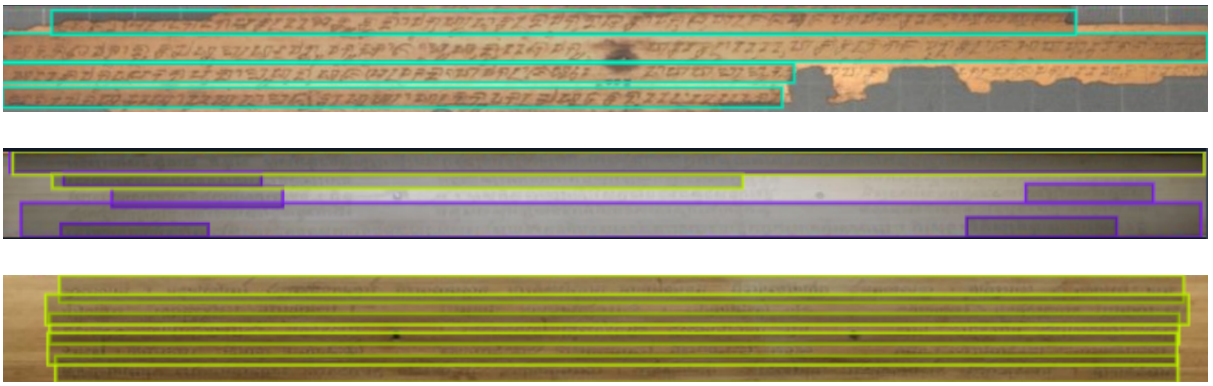


Figure 13: Results of (Sundanese, Balinese, and Khmer respectively) YoloNAS model

7. Conclusion

7.1 Key Finding

This project aimed to address the challenges of line detection in historical Indic manuscripts by leveraging advanced deep learning techniques, specifically YOLOv11. The key findings from this research are below:

- **Effectiveness of YOLOv11:** YOLOv11 demonstrated significant improvements in accurately detecting text lines in complex and noisy manuscript images. It outperformed traditional methods, particularly in handling overlapping characters, varied text formats, and degraded image quality.
- **Performance Across Diverse Datasets:** The model was tested on multiple datasets, including Sanskrit, Khmer, Sundanese, and Balinese manuscripts. YOLOv11 consistently

showed high accuracy and robustness across these datasets, effectively managing the unique challenges posed by each type of manuscript.

- **Comparison with Traditional Methods:** Traditional OCR and line detection methods struggled with the irregularities and noise inherent in historical manuscripts. YOLOv11, however, provided a more reliable and precise solution, significantly reducing errors and improving the overall quality of text segmentation.
- **Impact on Manuscript Digitization:** The successful application of YOLOv11 for line detection in historical manuscripts indicates its potential for enhancing the digitization process. Improved text segmentation facilitates better transcription, preservation, and analysis of these invaluable cultural and historical documents.

7.2 Future Work

While this project has improved line detection in historical Indic manuscripts, there are several areas where future work can build upon these results:

- **Improving Training Data**: We can collect more diverse manuscripts to train the model, which will help it work better with different kinds of texts.
- **Better Preprocessing:** Using advanced methods to clean and prepare the images before detection can make the model more accurate. This includes techniques like reducing noise and normalizing the background.
- **Skew Correction:** One of the challenges we faced was with lines that were tilted. Adding a step to correct the tilt of lines before detection will improve the model's performance.
- **Combining with OCR:** Future work can look into combining our line detection model with Optical Character Recognition (OCR) systems. This would help in reading and digitizing the manuscripts more effectively.

8. References

1. https://drive.google.com/file/d/1UU_4irR3m8IuuzuOXYl35eK6UbD2d3tH/view
2. https://ihdia.iiit.ac.in/seamformer/#materials
3. https://netraneupane.medium.com/text-skewness-correction-a51fd3a27157
4. https://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/
5. https://medium.com/@amit25173/opencv-line-detection-ccf6ee026c5c
6. https://www.tutorialspoint.com/line-detection-in-python-with-opencv

9. Appendix

Code & Dataset: https://github.com/yogeshjangir16/Line-Detection-Using-Yolo