Pokhara Engineering College Journal (PECJ)

Applied Sciences and Engineering Insights

ISSN: 3021-9795 (Print) / 3059-9628 (Online)
(Volume-3, Issue -I)

Design of a Hardware-Integrated OCR System for Devnagari Text in Nepalese Citizenships

Prashant Subedi[1*], Sandesh Bashyal[1], Pragati Basnet[1], Amrit Giri[1], Suraj Basant Tulachan[2], Smita Adhikari[3]

[1] *Department of Electronics and Computer Engineering, Pashchimanchal Campus, IoE, TU*

[2] *Department of Compute, Pokhara Engineering College, PU*

[3] *Department of Electronics and Computer Engineering, IoE, TU*

*Corresponding email: prashantsubedi35@gmail.com

Received:10- January-2026; Revised: 12- February-2026; Accepted: 27- March-2026

DOI: https://doi.org/10.3126/pecj.v3i1.93533

Abstract

In today's date banks to government institutions, most forms still require users to fill in details on the paper and then retype them into a database. Essential documents such as citizenship cards, national identification cards, driving licenses, and passports are only available in physical form. To digitize such information, an Optical Character Recognition (OCR) system is required. OCR systems are technologies that convert printed or handwritten text from the images into machine-readable format. A complete hardware-integrated machine learning framework is presented for automatically extracting Devanagari text from citizenship documents and storing it in a database without manual input. A document feeder mechanism equipped with a high-torque planetary gear motor was designed, in which the roller is rotated to replace documents from a stack and capture images sequentially. The captured image is processed using a YOLO-based model to detect the region of interest (ROI) of the document, which is then passed to Tesseract OCR for converting the printed Devanagari details into machine-readable text. Experimental results show that our model achieves a Character Error Rate (CER) of average 13% on previously unseen citizenship documents, showing the feasibility of our approach for large-scale document digitization.

Keywords: Character Error Rate (CER), Citizenship Document, Devanagari Script, Document Feeder Mechanism, Optical Character Recognition (OCR), Tesseract, YOLO.

1. Introduction

1.1 Background

Optical Character Recognition (OCR) is a technology that enables the automatic identification and conversion of printed or handwritten text into machine readable digital data. By analyzing the shapes and patterns of characters in images or scanned documents, OCR systems can extract textual information efficiently, eliminating the need for manual data entry by different bodies.

Previously OCR were used in technology involving telegraphy and creating reading devices for the bind. Later, the OCR was made available online as a service (WebOCR), in a cloud computing environment (Tafti *et al.*, 2016), and in mobile applications like real-time translation of foreign-language signs on a smartphone, with the advancement overtime they are used on the internet connected mobile devices to extract text from the images using the device's camera (Cutter and Manduchi, 2017).

Analyzing the characteristics of Nepalese citizenship documents (12.7*8.9 cm), designing a deep-learning OCR model tailored for Devanagari script, and integrating the model with dedicated hardware for real-time performance is what this paper is all about. The proposed system targets applications in financial institutions, government service centers, and digital archiving platforms where fast, reliable extraction of citizenship data is essential.

The recent advances in artificial intelligence and deep learning have transformed OCR capabilities by enabling models to automatically learn robust representations from large collections of annotated images. Convolutional Neural Networks (CNNs) improve text detection, while Recurrent Neural Networks (RNNs), Transformers (Hasan *et al.*, 2024), and Connectionist Temporal Classification (CTC) mechanisms enhance sequence prediction without the need for explicit character segmentation. Incorporating CNN and LSTM models in a single pipeline enables directly transcription of the text directly and this OCR engine is popularly known as Tesseract (Kumar *et al.*, 2024). These end-to-end learning systems have demonstrated strong performance in multilingual scene text recognition, making them suitable for languages such as Nepali (Mehta and Mehta, 2025) that require understanding of script-specific structural variations.

OCR systems understand contextual description of documents, multimodal and real-time-like processing of text in the video, enhanced adaptability, security and privacy. This impact on the application like automating data entry, invoice processing, financial operation, digitizing textbooks for easier research and study (Mittal and Garg, 2020).

1.2 Problem Statement

Nepal's administrative and financial systems still heavily rely on physical documentation. Most institutions require an individual person just to fill in paper forms; later, the employees manually transfer that information into digital databases. This workflow introduces multiple inefficiencies: Manual data entry is slow, Human errors, Large-scale digitization is impractical, Existing OCR solutions lack support for Devanagari.

Given these limitations, there is a need for a system that can automatically feed the physical documents, capture their images, detect relevant textual regions from the obtained image, and convert printed Devanagari text into structured machine-readable data. Such a solution must be accurate, reliable, and capable of operating without human intervention to support nationwide digitization efforts.

1.3 Objective

The objective of the research study is:

- To develop an accurate and efficient system that detects the region and extracts Devanagari text from citizenship documents for digitization in banks and government institutions

2. Literature Review

A segmentation-free LSTM-based recognition approach has been applied to significantly enhance printed Devanagari OCR performance. The outcomes were reported in the study by Karayil et al. (2015), where LSTM achieved a 9% error rate compared to Tesseract's 11.96%, highlighting the benefits of sequence-learning models for structured Indic scripts (Karayil, Ul-Hasan and Breuel, 2015).

Research conducted by Dessai and Patil in 2019 extended OCR research to handwritten Devanagari text by employing a convolutional neural network model (Dessai and Patil, 2019). Their system achieved an accuracy of approximately 89%, showing the effectiveness of deep-learning-based feature extraction in handling the irregularities and variations present in handwritten scripts.

Comparative studies between traditional OCR engines and artificial neural networks further demonstrate the progression toward the latter. Neural network models have achieved higher training and testing accuracies than Tesseract when applied to Nepali script, demonstrating better adaptability and generalization. This trend was observed in the work of Prajapati et al. (2018), where ANN-based OCR outperformed Tesseract across multiple evaluation premises (Prajapati *et al.*, 2018).

Document-level OCR automation has recently been enhanced through integrated deep-learning-powered text detection pipelines. Such systems combine detection and recognition to handle real-world identification documents with high precision. A notable example is the citizenship-document extractor developed by Dhakal et al. (2024), which achieved mean average precision values of 99.1% on the document front and 96.4% on the back, illustrating the maturity of end-to-end OCR frameworks for official document processing (Dhakal *et al.*, 2024).

In 2020, Luthra team demonstrated a low-cost hardware-assisted OCR workflow by developing an automated document scanning machine using Raspberry Pi for camera control, page turning, and flexible motor–sensor positioning (Luthra, Chauhan and Ahmad Ansari, 2020). Their system incorporated image preprocessing prior to Tesseract OCR, resulting in improved text extraction performance across documents of varying sizes and thicknesses.

A more advanced hardware–software co-design was proposed by Rybalkin et al. (2020) through the iDocChip architecture, a portable and low-power SoC-based accelerator designed for real-time historical document processing (Rybalkin *et al.*, 2018). By integrating segmentation, image enhancement, and neural-network-based OCR directly into a CPU–FPGA hybrid workflow, the system delivered significant improvements in throughput and energy efficiency, establishing one of the most cited hardware-accelerated OCR frameworks in recent literature.

A modular hardware OCR implementation was presented by Parween and Saha, who structured their system around dedicated preprocessing, segmentation, and feature-extraction modules (Parween and Saha, 2021). Character recognition was performed using an artificial neural network implemented in MATLAB, revealing the advantages of classical ANN models for printed-text OCR when paired with structured hardware support.

Assistive hardware applications have similarly benefited from embedded OCR pipelines. Team along with Saint-Vil introduced a headband-mounted document scanner incorporating a microcontroller, high-resolution camera, and wireless transmission module for hands-free text capture (Saint-Vil *et al.*, 2022). The system leveraged Tesseract OCR for recognition alongside a live video stream to a mobile application, providing an accessible solution for users with motor impairments and highlighting the role of lightweight embedded platforms in personalized OCR systems.

3. Methodology

3.1 Datasets

The dataset consists of images of Nepali citizenship documents containing both printed and handwritten components with text written in Nepali language. Along with these images, there exist ground truth annotations for structured components with a total of six prominent fields. The images were obtained from structured forms filled out by participants who were given a non-disclosure

and data-use agreement with mutual consent. The images have been carefully labeled with labels corresponding to these structured components named as Name, Citizenship Number, Issuing Office, Gender, Birthdate, and Address. Various preprocessing operations have been done on these images including denoising and transforming them into grayscale. To make these images more robust under differing imaging conditions, some more operations have also been conducted on these images. The brightness, saturation, and hue of these images were controlled. It should be noted that for consistency within the data set, images were obtained and then resized to 640x640 resolution. The data set itself consists of 85%, 7.5%, and 7.5%, partitioning into train, validate, and testing sets, respectively.

3.2 System Architecture

The Nepali citizenship documents were fed one by one into an automatic paper feeder mechanism controlled by a microcontroller and a relay switch. The document is loaded into a dedicated frame, where the top citizenship image is captured using a camera module. After that, the raw image is transmitted to the backend server, which acknowledges the receipt and triggers the feeder to process the next document.

The backend communicates with both the microcontroller and the machine learning model about the readiness of the document to capture and the captured one to process further, respectively. The machine learning model, after receiving the captured images, comes into play. The workflow of the model part is explained in detail in the section below. After receiving the output from the machine learning model, it is sent to Tesseract to generate the characters in the form of a result. All the results obtained out of it are gathered and stored for further usage.

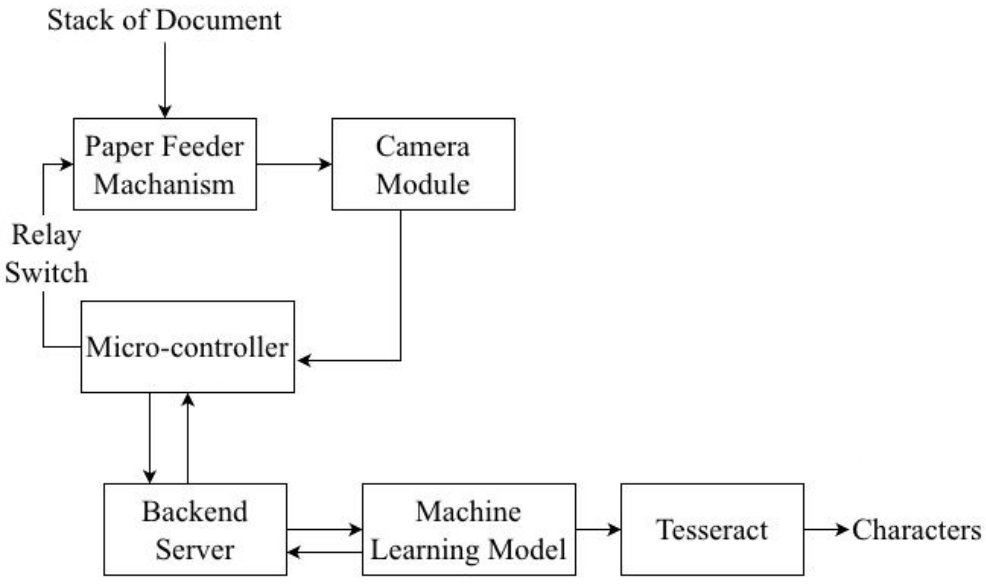

Figure 1: System Architecture

3.3 Hardware Architecture

The illustrated figure shows the mechanism of the automatic paper feeder and scanner. As shown in (1), the bundle of citizenship documents is placed. The top part of the document is captured using a camera module that is placed at (2). The captured image is then sent to the backend server. The server then passes it to the machine learning framework.

A binary handshaking protocol is implemented between the Arduino and the backend to ensure precise synchronization between document processing and mechanical actuation. The Arduino continuously checks for an acknowledgement from the backend server. The backend triggers the microcontroller to fetch the next paper by switching ON the relay module as soon as it acknowledges reception of the image. The microcontroller sends a high signal to turn on the relay as initially it's in the OFF state. The relay module is responsible for supplying power to a high-torque DC motor. This DC motor drives the primary roller and the roller attached to the primary roller via a rubber belt in a clockwise direction. Similarly, the secondary roller, which is in contact with the primary roller, moves as a reaction to the primary roller in an anticlockwise direction, thereby moving the paper forward as shown in (4). Then the whole process is repeated until all of the paper is passed through the motor.

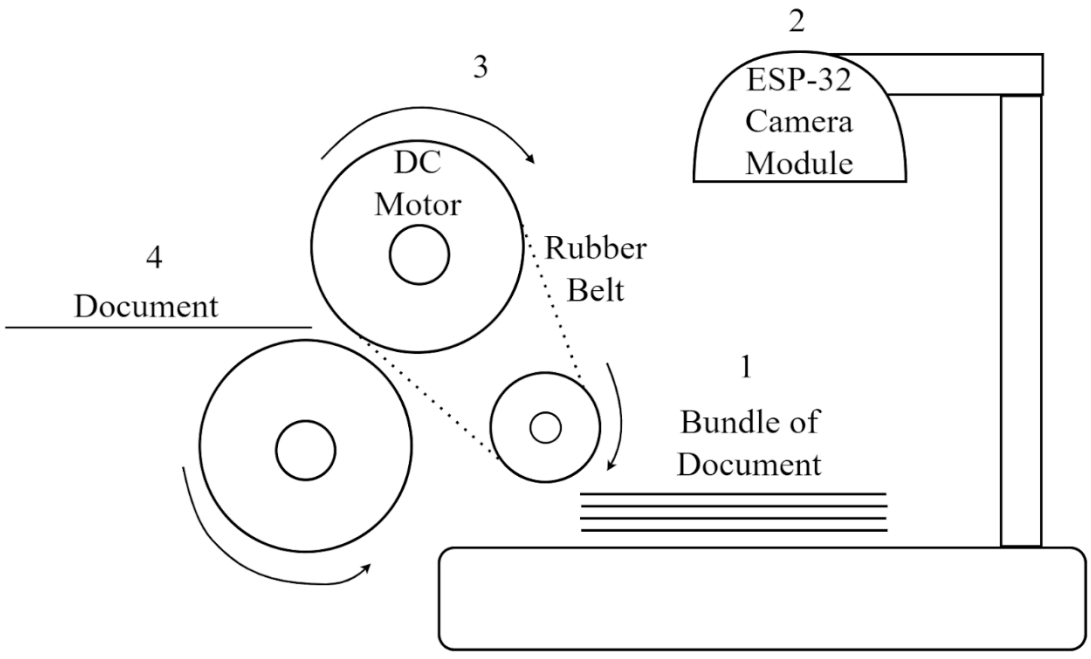

Figure 2: Document Feeder Mechanism

3.4 Machine Learning Workflow

The machine learning workflow as shown in diagram 3 begins with image acquisition, which is done with the help of the hardware setup described previously. After which, the image is passed to an ML-driven Region of Interest (ROI) detection algorithm. An ROI refers to any specific portion of the input image containing meaningful information that was annotated. The algorithm identifies these regions within scanned documents and isolates them through the cropping of ROIs to remove irrelevant details. After cropping, the cropped segments are passed into the character reader (such as Tesseract), from where characters are extracted from the segments. The raw OCR output then undergoes a cleaning process that removes noise and normalizes characters. Finally, all cleaned outputs are assembled into a structured format, producing a coherent digital representation of the original document.

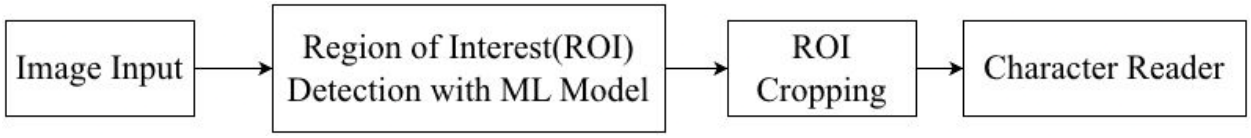

Figure 3: Machine Learning Workflow

3.5 Model Architecture

There are 2 model architectures used in the research. First one is YOLO and is followed by Tesseract which are explained below.

3.5.1 YOLO

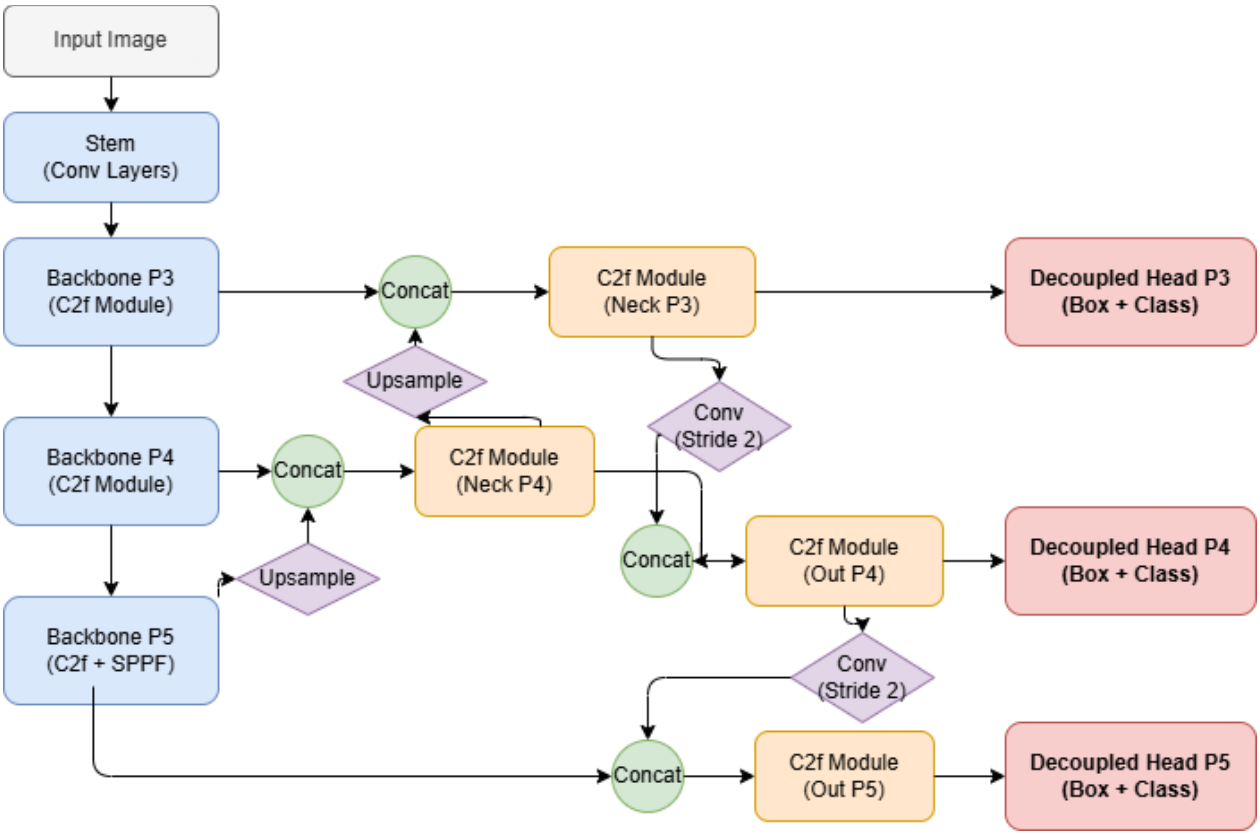

Figure 4: YOLO Architecture

YOLO architecture has always been popular on the object detection problem (Liu *et al.*, 2018). The architecture flow of figure 4 depicts a modern YOLO detector made up of a convolutional Stem layer (process images in first layer), a hierarchical (cross- stage partial with 2 fusion paths) C2f-based Backbone, and a multi-scale fusion Neck. Stem manages initial spatial reduction and low-level feature extraction, providing progressively richer representations to Backbone stages P3, P4, and P5. Each stage employs C2f modules to boost gradient flow, efficiency, and feature reuse; the deepest stage incorporates an SPPF block to gather multi-scale contextual information and enhance large-object detection. These components work together to form a hierarchy of semantically rich feature maps, which are then processed at various resolutions.

These features are fused in the Neck through up sampling, stride convolution, and feature concatenation, with additional C2f modules refining each merged representation to produce three output scales (Out P3, Out P4, and Out P5). Each scale is handled by a decoupled detection head that separates bounding-box regression and classification, reducing gradient interference while increasing accuracy. Architecture detects objects of varying sizes in a robust and computationally efficient manner by utilizing efficient feature extraction, cross-scale aggregation, and task-separated prediction.

3.5.2 Tesseract

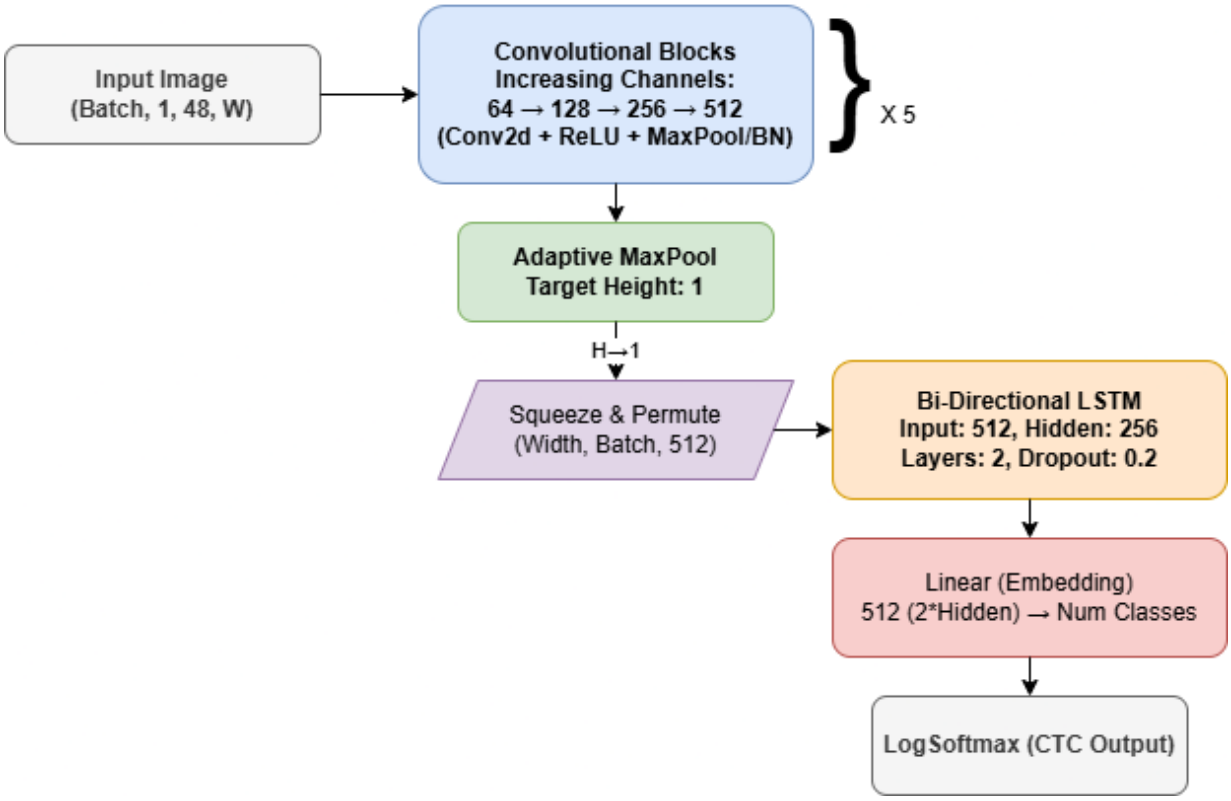

Figure 5: Tesseract Architecture

This architecture illustrated in figure 5 represents a deep learning version of the Tesseract OCR recognition system. It uses convolutional feature extraction alongside sequence modeling to ensure precise text retrieval. The input image, normalized to a fixed height and flexible width, passes through five blocks, which boosts channel capacity starting from 64 up, to 512. Every block contains convolution, ReLU activation and either max pooling or batch normalization, enabling the network to capture abstract visual features while minimizing spatial dimensions. This convolutional encoder generates a representation of nearby stroke configurations, letter forms, and text layout.

After extracting features, an adaptive max-pooling step compresses the height dimension to one row, transforming the 2D feature map into a sequence. Each line of text is handled as a sequence during a squeeze-and-permute operation. It changes the tensor into a (Width, Batch, 512) shape. The sequence is then processed by a two-layer bidirectional LSTM containing 256 hidden units in each direction. Doing so enables data from both past and future positions to affect every prediction. Afterwards, a linear embedding layer translates the 512- features into character categories. Finally, the Log SoftMax layer generates probabilities for Connectionist Temporal Classification (CTC). Combining all these allows the precise recognition of variable-length text without the need for any character segmentation.

3.6 Training

The citizenship document datasets were prepared with all the preprocessing steps. It was then trained using Roboflow. The preprocessed images were uploaded and annotated manually with bounding boxes for predefined classes representing key textual fields. The annotations were all reviewed manually again to ensure consistency and accuracy in proper localization of the designated fields.

Roboflow preprocessing included image resizing, noise clearance through normalization, and removing low-quality document samples. Data augmentation techniques were applied that include hue, saturation, and brightness adjustments. It was essential to make the model more robust. Before being exported in YOLO format for model training, the datasets were divided into training, validation, and test sets.

The model was trained for 100 epochs with a weight decay of 1×10^{-3} to reduce overfitting and improve generalization. Different hyperparameters were changed and the best ones to produce stable convergence and balanced performance across all classes were selected. Different validation metrics were considered to verify the performance.

3.7 Evaluation

The various evaluation metrics used in the study to determine the feasibility and efficiency of the YOLO model are loss curves, precision and mean Average Precision (mAP50-95) graphs, confusion matrix, and that of the extracted text from Tesseract model is Character Error Rate (CER).

1. Loss Graphs:

Loss graphs represent the model's training progress over time. It shows the change in loss values over time. The total loss in YOLO is made up of several components, one of the most important being the bounding box (localization) loss. It determines whether the predicted box aligns with the ground truth ones. The bounding box loss graph is plotted on both training and validation data. The decreasing training loss indicates a better fit to the training data. In validation data, it indicates better generalization on unseen data.

2. Precision:

Precision refers to the proportion of true positive detections to the total positives (true positives and false positives). Mathematically, it is represented as:

$$Precision = \frac{TP}{TP + FP} \quad 1$$

Where TP = True Positive, FP = False Positive.

The precision graph shows how accuracy in detection changes with training epochs or confidence thresholds.

3. Mean Average Precision 50-95:

Mean Average Precision (mAP50–95) is one of the main metrics which is used to determine the performance of YOLO-based models in object detections. It calculates the mean of Average Precision (AP) values over various Intersection over Union (IoU) thresholds from 0.50 to 0.95 with a step of 0.05. With the step of 0.05, this metric calculates the average of those 10 average precisions. This metric assesses strict localization, accurate classification, and makes it way better than mAP50 in complicated and sensitive tasks that seek the best accuracy. The formula for mAP_{50-95} is given as:

$$mAP_{50-95} = \frac{1}{10}\sum_{t=0.5}^{0.95} AP_t \quad (2)$$

Where t = threshold.

4. Confusion Matrix:

A confusion matrix, always a square matrix, is the performance evaluation tool used for classification models. It helps to compare the true and predicted labels produced by the trained model. Each row represents a predicted class, while each column speaks for actual class and a strong efficient model always has strong principal diagonal values showing how well the model correctly classifies each class.

5. Character Error Rate:

Character Error Rate is an OCR accuracy metric that evaluates the percentage of incorrect characters in the extracted text when compared to the correct ones. Mathematically, it is defined as,

$$CER = \frac{S + D + I}{N} \quad (3)$$

where S = Substitutions, D = Deletions, and I = Insertions are normalized by the total number of ground-truth characters (N).

4. Result and Discussion

4.1 Results

To evaluate the effectiveness of the model architecture, a series of experiments with different combinations of hyperparameters were conducted. By systematically tuning the learning rates, batch sizes, and image size, we aimed to identify the configuration of architecture that could

deliver most contented performance. These hyperparameters helped to get a significant Region of Interest (ROI), resulting in high diagonal values in the confusion matrix.

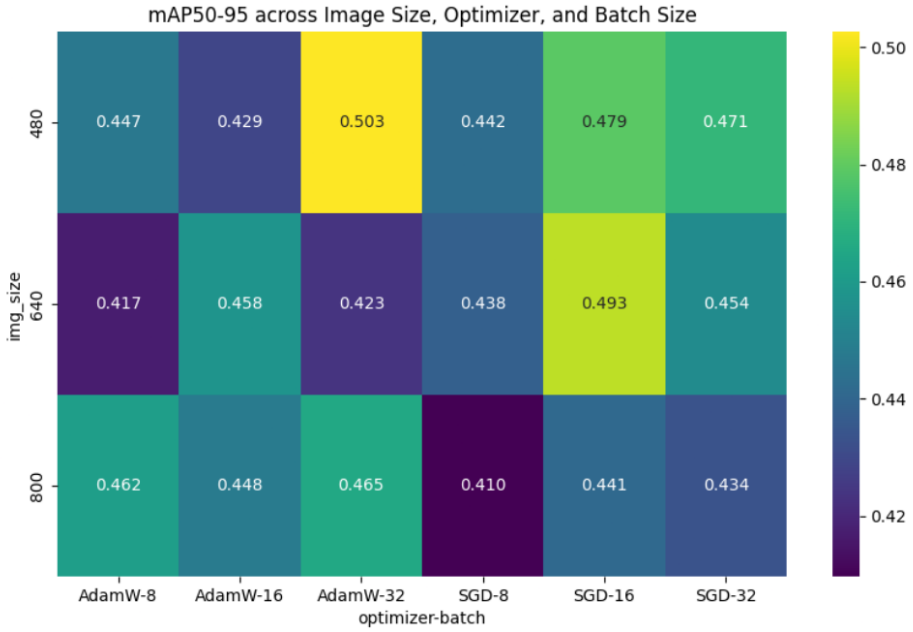

Figure 6: Combinations of image size, optimizer and batch size

The object detection model was evaluated on 18 different combinations involving 3 image square shapes of 480, 640, 800, 2 optimizers (AdamW and SGD), and 3 batch sizes of 8, 16, 32. The resulting mAP50-95 values are illustrated in the heatmap figure 6, where darker regions represent low performance and brighter high. From the diagram, AdamW optimizer produced the best results at an image size of 480 and batch size of 32, with a mAP50-95 of 0.503.

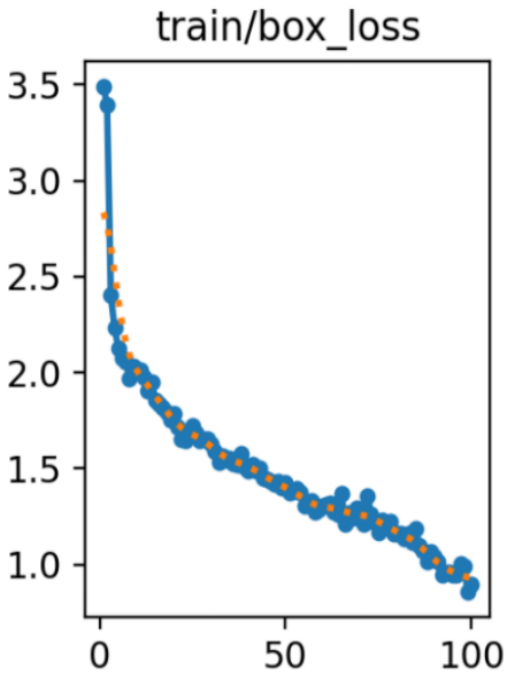

Figure 7: Bounding box graph of train loss

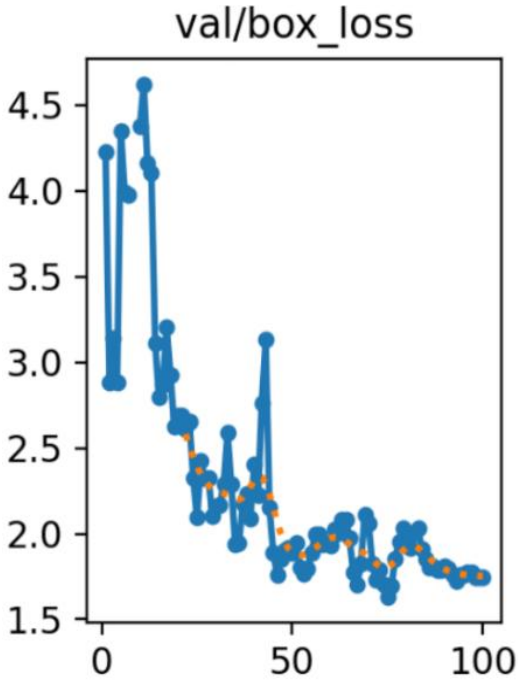

Figure 8: Bounding box graph of validation loss

Figures 7 and 8 represent the loss curve of the train dataset and validation dataset respectively. Loss value during training means the discrepancy between predicted and actual values at a certain epoch. The model training loss experienced a very steep decline from 3.5 to 1.5 in the first 25 epochs. Then there is gradual decrease to 100 epochs which is less than. On the other hand, validation loss fluctuates until the first 50 epochs while it tries to be in a steady state for the next 50 training periods.

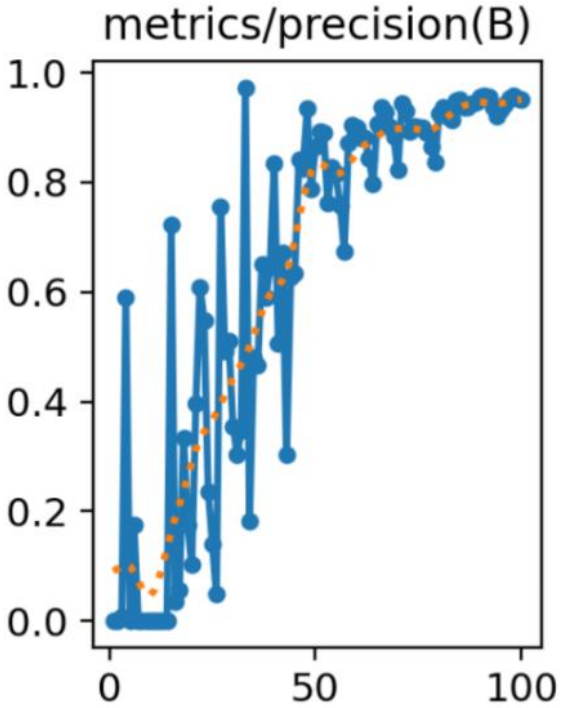

Figure 9: Precision graph

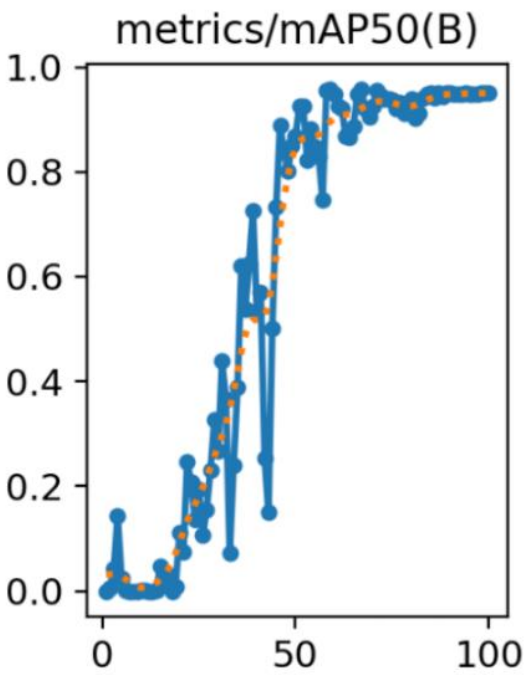

Figure 10: Mean average precision 50 graph

Likewise, figures 9 and 10 represent the precision and mAP50 evaluation metrics during the model training respectively. There is fluctuation in both precision and mAP50 till 50 epochs in the range of 0.1 to 0.75 and remains in stable state for next epoch. These metrics define how models are optimized from each data sample.

Figure 11 depicts the confusion matrix of dimension 7×7 incorporating Region of Interest (ROI) of the targets. However, there are 6 labels and 7 in dimension with consideration of background because predicting the regions, the model may label unnecessary regions as the specific target. The labels named 'DOB', 'address', 'karyalaya' (Issuing Office), and 'sex' correctly predict the region with 100% accuracy. In contrast, predicting 'Citizenship number' and 'name' regions, YOLO over captures these regions with the accuracy of 86% making it error prone. These fields from each test sample are cropped and sent to the Tesseract to evaluate CER Value.

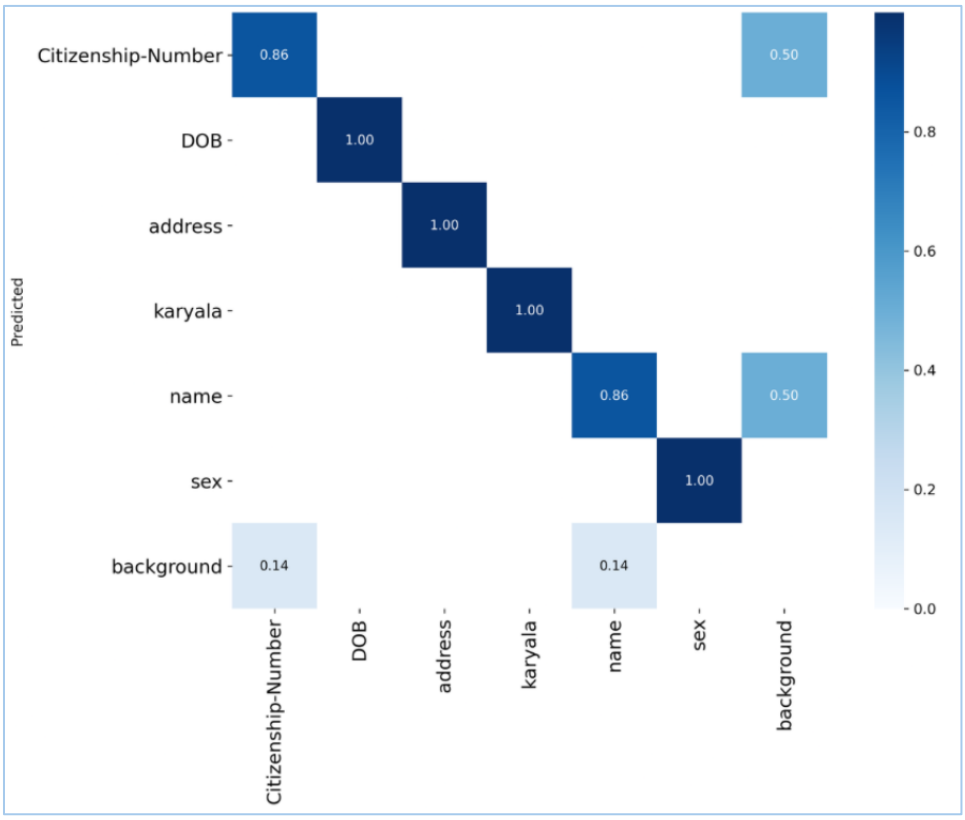

Figure 11: Confusion matrix of test dataset

Above experimental results are deliverable from Yolo model architecture.

Table 1: CER values in test sample

Test Files	CER Value
f1	0.156
f2	0.119
f3	0.130
f4	0.159
f5	0.069
f6	0.119
f7	0.168
Total: 7 files	**Average CER Value: 0.1315**

Table 1 presents 7 unseen files with their respective CER value after ROI fields being cropped from YOLO and assessed from Tesseract. Each of the tested file CER is below 20%, with highest and lowest value in 7th and 5th file with values 16.8% and approx. 7% respectively.

4.3 Limitations

The trained model still struggles with handwritten text and old format of the Nepalese citizenship. This is due to the overlapping of the target regions. As intended, it supports only the Devnagari script of the document while failing to extract findings from the back of the paper. Likewise, it also fails for other linguistic variant countries, and multi-script documents. The system's metrics quality lessens as the quality of the image record degrades. The hardware is adjusted as per the height and record size. For different shapes, the hardware system needs to be modified to make documents fit.

5. Conclusion

This research presented a hardware-integrated OCR system designed for the digitization of Nepali citizenship documents, addressing the challenges associated with the intensive manual data entry and document handling. By combining a custom relay-controlled paper feeder mechanism, synchronization of hardware, backend and machine learning model with Tesseract, the system demonstrates an effective pipeline for automated extraction of required information from citizenship records.

The YOLO-based object detection model showed promising results by detecting the ROI of to-extracted labels like Name, Citizenship Number, Gender, etc., where Tesseract extracts Devnagari text from those regions with good character error rate supporting automated digitization. The result showed an average CER value with approximately 0.13 across test samples reflecting the model's capability to perform on structured printed fields. Nonetheless, the study validates that the end-to-end system with hardware is a faster and more convenient alternative for different banks, government institutions and agencies to handle the information of the Nepalese documents improving digitization workflows in Nepal.

6. Recommendation for future Research

Even though the proposed system provides an efficient foundation for automated Nepali citizenship digitization, there is still room for improvements and advancements. The first extension would be the support of a multilingual pipeline including English scripts as many government documents support English language. Next would be incorporating deep learning-based attention models like Transformer for the character recognition. Testing on these models would significantly improve character error rate.

From the hardware perspective, the paper feeder mechanism should be adjusted making it compatible with any type of government records based on document sizes, thickness and color. Encrypting it with security keys will further enhance the usability of the system in real-world cases. Additionally, real-time mechanisms can enable centralized storage, faster verification. Ultimately, all above future work can significantly strengthen digital governance initiatives in Nepal.

References

- Cutter, M. and Manduchi, R. (2017) Improving the Accessibility of Mobile OCR Apps Via Interactive Modalities | ACM Transactions on Accessible Computing. Available at: https://dl.acm.org/doi/abs/10.1145/3075300 (Accessed: December 13, 2025).

- Dessai, B. and Patil, A. (2019) "A Deep Learning Approach for Optical Character Recognition of Handwritten Devanagari Script," in 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), pp. 1160–1165. Available at: https://doi.org/10.1109/ICICICT46008.2019.8993342.

- Dhakal, S. et al. (2024) "Mero Nagarikta: Advanced Nepali Citizenship Data Extractor with Deep Learning-Powered Text Detection and OCR." arXiv. Available at: https://doi.org/10.48550/arXiv.2410.05721.

- Hasan, S.M.R. et al. (2024) "Optical Text Recognition in Nepali and Bengali: A Transformer-based Approach." arXiv. Available at: https://doi.org/10.48550/arXiv.2404.02375.

- Karayil, T., Ul-Hasan, A. and Breuel, T.M. (2015) "A segmentation-free approach for printed Devanagari script recognition," in 2015 13th International Conference on Document Analysis and Recognition (ICDAR). 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 946–950. Available at: https://doi.org/10.1109/ICDAR.2015.7333901.

- Kumar, S. et al. (2024) "Text Extraction from Images Using Tesseract," in Deep Learning Techniques for Automation and Industrial Applications. John Wiley & Sons, Ltd, pp. 1–18. Available at: https://doi.org/10.1002/9781394234271.ch1.

- Liu, C. et al. (2018) "Object Detection Based on YOLO Network," in 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC). 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), pp. 799–803. Available at: https://doi.org/10.1109/ITOEC.2018.8740604.

- Luthra, P., Chauhan, P. and Ahmad Ansari, I. (2020) "Low Cost Automatic Document Scan Machine Using Raspberry Pi and Tesseract Ocr." Rochester, NY: Social Science Research Network. Available at: https://doi.org/10.2139/ssrn.3575389.

- Matveev, I.V., Khalyasmaa, A.I. and Matrenin, P.V. (2025) "Analysis of Modern Models Capabilities and Software for Automatic Annotation of Graphical Data in the Power Industry," in 2025 IEEE 26th International Conference of Young Professionals in Electron Devices and Materials (EDM). 2025 IEEE 26th International Conference of Young Professionals in Electron Devices and Materials (EDM), pp. 990–994. Available at: https://doi.org/10.1109/EDM65517.2025.11096708.

- Mehta, D. and Mehta, P. (2025) "Devanagari Handwritten Character Recognition using Convolutional Neural Network." arXiv. Available at: https://doi.org/10.48550/arXiv.2507.10398.

- Mittal, R. and Garg, A. (2020) "Text extraction using OCR: A Systematic Review," in 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA). 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 357–362. Available at: https://doi.org/10.1109/ICIRCA48905.2020.9183326.

- Parween, G. and Saha, S. (2021) "Design of an OCR System and its Hardware Implementation," International Journal for Research in Applied Science and Engineering Technology, 9(12), pp. 159–174. Available at: https://doi.org/10.22214/ijraset.2021.39217.

- Prajapati, S. et al. (2018) "Evaluating Performance of Nepali Script OCR using Tesseract and Artificial Neural Network," in 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS). 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), pp. 104–107. Available at: https://doi.org/10.1109/CCCS.2018.8586808.

- Rybalkin, V. et al. (2018) "iDocChip: A Configurable Hardware Architecture for Historical Document Image Processing: Percentile Based Binarization," in Proceedings of the ACM Symposium on Document Engineering 2018. New York, NY, USA: Association for Computing Machinery (DocEng '18), pp. 1–8. Available at: https://doi.org/10.1145/3209280.3209538.

- Saint-Vil, A. et al. (2022) "High Resolution Headband Document Scanner and Transmitter for the Disabled People," in 2022 IEEE International Conference on Imaging Systems and Techniques (IST). 2022 IEEE International Conference on Imaging Systems and Techniques (IST), pp. 1–2. Available at: https://doi.org/10.1109/IST55454.2022.9827664.

- Tafti, A.P. et al. (2016) "OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym," in G. Bebis et al. (eds.) Advances in Visual Computing. Cham: Springer International Publishing, pp. 735–746. Available at: https://doi.org/10.1007/978-3-319-50835-1_66.