Kathmandu University Journal of Science, Engineering and Technology, Vol. 18, No. 2, December 2024

Kathmandu University
Journal of Science, Engineering and Technology

Developing Ran2Dev: A Model Converting Ranjana Lipi to Devanagari Script (Without Modifiers)

Nischal Baidar*[a], Surag Basukala[a], Sarif Tachamo[a], Nekesh Koju[a], and Shree Ram Khaitu[b]

[a]Department of Computer Engineering, Khwopa Engineering College, Purbanchal University, Nepal.
[b]ITD, Nepal Telecom

Abstract

Nepal Bhasa (Newar language), traditionally spoken by the Newar community in the Kathmandu Valley, carries significant cultural and historical importance. Over the years, the number of native speakers has gradually declined. In 2024, Nepal Bhasa gained official recognition as an indigenous language of Nepal and currently holds official status in Bagmati Province and local governments such as Kathmandu. One of the primary writing systems for this language is the Ranjana script, which includes 16 vowels, 36 consonants, and 10 numerals. This paper introduces an Optical Character Recognition (OCR) system that converts characters from the Ranjana script into the Devanagari script. To achieve this, two convolutional neural network (CNN) models were used: the standard LeNet-5 and a proposed model named Ran2Dev. The system incorporates processes such as image preprocessing, feature extraction, and model evaluation to enhance accuracy. The Ran2Dev model outperformed LeNet-5 with an accuracy of 99.74% compared to 99.10%, showing strong effectiveness in recognizing Ranjana script characters.

Keywords: Ranjana Script; Nepal Bhasa; Convolutional Neural Network; Deep Learning; Character Segmentation, Indigeneous Language Preservation

1. Introduction

Nepal Bhasa, also known as the Newar language, is the native language of people of the Newar, who are the indigenous inhabitants of the Kathmandu Valley and surrounding regions of Nepal [1]. With a history stretching over a millennium, Nepal Bhasa has played a pivotal role in shaping the cultural, administrative, and artistic heritage of Nepal, particularly during the Malla period when it served as the primary language of governance, literature and trade [1]. Belonging to the Tibeto-Burman family, Nepal Bhasa is distinct from the Indo-Aryan Nepali language and has evolved through rich interaction with Sanskrit, Pali, and other regional languages [1]. Its origins traces back to ancient languages, with early inscriptions and literary works dating from the Lichhavi dynasty. Over successive dynasties, its written form transitioned through various Brahmic scripts, including Gupta Lipi, Siddham, Purba Lichhavi, and Uttar Lichhavi, before the emergence of Ranjana script in the 7th century [2].

The Ranjana Script also known as Lantsa, is one of the most prominent traditional scripts used for writing Nepal Bhasa, Sanskrit, and Pali [2]. Developed in the 7th century and reaching artistic maturity during the medieval period. Ranjana is celebrated for its visually striking, ornamental characters comprising 36 consonants,16 vowels and 10 numerals [2] . Its unique geometric and calligraphic qualities made it an ideal choice for religious manuscripts, stone inscriptions, temple carving, and mantras, not only in Nepal but also in many other countries [2]. The script is typically written from left to right and is distinguished by horizontal lines and elaborated curves [2].

According to the National Population and Housing census 2021 of Nepal, there are approximately 863,380 native Nepal Bhasa speakers, representing 2.96% of the total population [3]. This data reflects the continuing presence of Nepal Bhasa as a significant indigenous language spoken primarily by the Newar community in the Kathmandu Valley and surrounding regions of Nepal. Efforts to revive Ranjana Lipi have gained momentum since the 1950s, with the publication of alphabet books, the development of metal and computer typefaces, and community driven initiatives to teach the script [2]. Recent research efforts have increasingly focused on leveraging modern technology, particularly the development of Optical Character Recognition (OCR) systems, to digitally preserve and promote handwritten texts of indigenous languages.

This research presents an OCR (Optical Character Recognition) system designed to convert images of Ranjana Lipi characters into Devanagari text. This work marks an important initial step toward building a robust and reliable tool that can convert Ranjana Lipi images at the word level into Devanagari text with high accuracy. By doing so, this system can enable people without knowledge of the Ranjana script to easily access, read, and understand the rich content written in historical and culturally significant materials, thereby supporting preservation efforts and promoting wider linguistic and cultural awareness.

Two convolutional neural network models were used: the predesigned LeNet-5 and a new proposed model called Ran2Dev. Both models were trained and tested on Ranjana characters to evaluate their performance. The evaluation included monitoring training over multiple epochs, analyzing accuracy and loss curves, and generating a confusion matrix to visualize classification performance across different character classes (62 character classes), ensuring a comprehensive understanding of the model's recognition capabilities.

*Corresponding author. Email: nischalbaidar@gmail.com

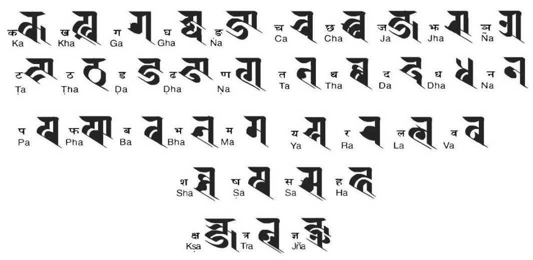


Figure 1: Set of 36 consonant characters in the Ranjana script.

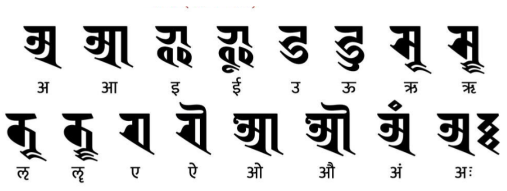


Figure 2: Set of 16 vowel characters in the Ranjana script.

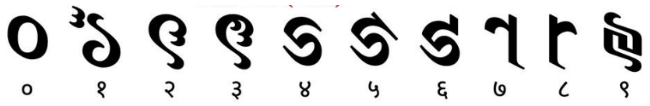


Figure 3: Set of 10 numeric characters in the Ranjana script.

2. Related Works

Several studies have contributed to the field of handwritten character recognition for Brahmic scripts, offering valuable insights relevant to this work. One notable approach directly addresses the recognition of handwritten Ranjana Lipi characters using Convolutional Neural Networks (CNNs), providing a closely aligned methodology for the recognition component of the proposed system. In this study, Bati and Dawadi achieved promising accuracy despite the challenges posed by Ranjana Lipi's intricate calligraphy and the scarcity of large, annotated datasets, demonstrating the feasibility of applying deep learning techniques to traditional scripts with limited resources [4].

Another notable study employs a CNN model with two convolutional layers to recognize handwritten Devanagari characters. By working with the Devanagari Handwritten Character Dataset, which contains 36 different character classes, the research showed how CNNs can effectively learn the unique spatial patterns and handwriting variations found in this script. The model achieved strong performance on test data, demonstrating the potential of CNN based approaches for digitizing complex Indic scripts [5].

In addition to pure CNN models, research exploring hybrid deep learning models, specifically combining CNN and Recurrent Neural Networks (RNN), presents a promising direction for future extensions of this work. Ma proposed a hybrid CNN-RNN architecture for handwritten recognition, leveraging CNNs for spatial feature extraction and RNNs for capturing sequential dependencies. This model demonstrated improved performance in scenarios involving sequential patterns, offering a potential strategy for managing sequential elements such as vowel modifiers and conjunct characters in Brahmic scripts [6].

Furthermore, studies emphasizing script identification and image preprocessing have highlighted the importance of robust input preparation, which is essential for achieving reliable recognition results. Damade et al. focused on extracting text from images with complex backgrounds, using a morphological method for text extraction and projection profiles for segmentation. Their system identified Devanagari script by applying visual feature extraction and heuristic classification. This work highlights how strong preprocessing like noise removal and careful segmentation is key to handling noisy or textured images, a challenge similar to recognizing handwritten Ranjana Lipi [7].

These studies collectively provide a strong foundation for the proposed Ran2Dev model, informing both the design of its deep learning based recognition component and the preprocessing strategies necessary for accurate and efficient script conversion.

3. Methods and materials

3.1. Data Collection

For each of the 62 classes, including numerals, vowels, and consonants, around 450 handwritten character images were collected. To improve the model's accuracy and generalization, the dataset was augmented by introducing noise, increasing the total number of images to 33,961. The data was split into training, test, and validation sets using a 70:10:20 proportion. Model training was performed over several experiments with epochs varying from 30 to 70 in steps of 5 (i.e., 30, 35, 40, and so on), allowing assessment of performance across different training lengths.

3.2. Preprocessing Steps

- **Grayscale Conversion:** Input RGB images were converted to grayscale, reducing dimensionality and computational load while retaining essential character features.

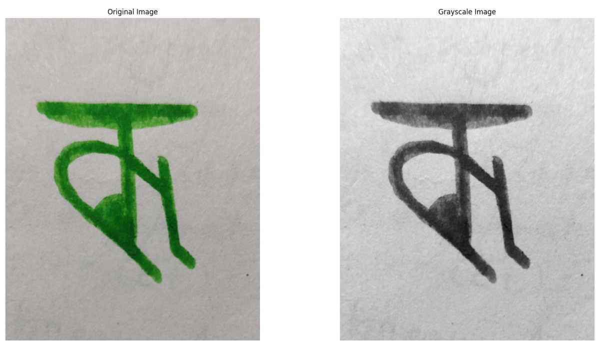


Figure 4: Process of Converting Original Image into Grayscale Image.

- **Aspect Ratio Scaling:** Images were resized while preserving their original proportions to prevent distortion, ensuring accurate feature extraction during model training.

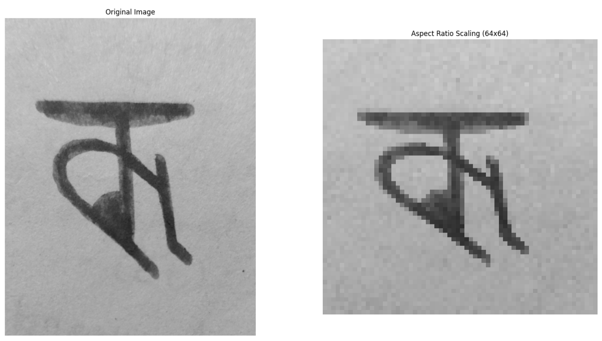


Figure 5: Resizing the input character image while preserving its original aspect ratio.

- **Otsu's Thresholding and Binary Inversion:** Otsu's thresholding was applied to perform foreground-background separation. This method automatically selects an optimal threshold value T^* that maximizes the variance between classes, ensuring effective binarization.

$$\sigma_B^2(T) = \omega_0(T) \cdot \omega_1(T) \cdot [\mu_0(T) - \mu_1(T)]^2 \tag{1}$$

where:

  - $\omega_0(T)$ and $\omega_1(T)$ are the probabilities of background and foreground classes,
  - $\mu_0(T)$ and $\mu_1(T)$ are the means of the background and foreground pixel intensities.

This method efficiently handles variations in stroke thickness and ink intensity found in handwritten Ranjana Lipi characters.
Binary inverse thresholding was then applied to enhance the visibility of lighter strokes using the following rule:

$$I(x,y) = \begin{cases} 255, & \text{if } I(x,y) < T \\ 0, & \text{if } I(x,y) \geq T \end{cases} \tag{2}$$

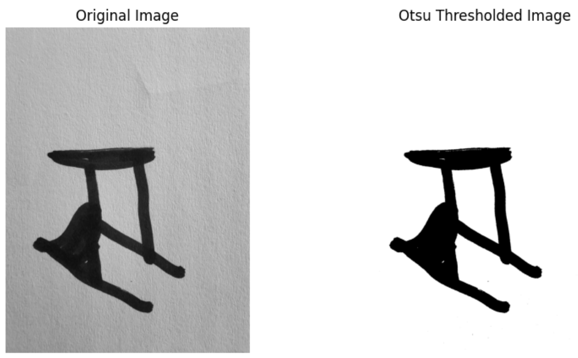


Figure 6: Otsu Thresholding with Binary Inversion to segment the character region from the background.

- **Gaussian Blur:** To minimize scanning artifacts and ink irregularities, a Gaussian Blur was used. It smooths the image by convolving it with a Gaussian kernel:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{3}$$

where σ determines the level of blurring. The technique reduces noise effectively without eliminating important edges and shapes.

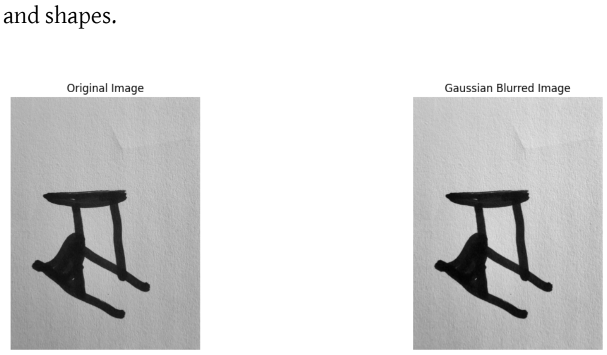


Figure 7: Gaussian Blur to the image for noise reduction and smoothing before further preprocessing.

3.3. Model Architectures

LeNet-5: It is a convolutional neural network architecture tailored for image classification and adapted in this work for Ranjana script recognition. It processes 32×32 grayscale images using the convolution layer and the average pooling layer twice, and finally three fully connected layers, concluding with a softmax output for classifying 62 character classes. Trained using the Adam optimizer and sparse categorical cross-entropy, the model is efficient for CPU based training and offers reliable performance, making it a practical choice for digitizing traditional scripts.

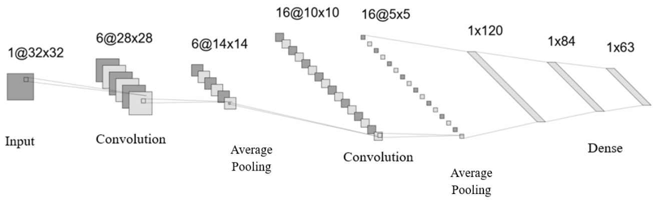


Figure 8: LeNet-5 CNN Architecture Showing Layer-Wise Structure from Input to Output.

Ran2Dev: Ran2Dev is a proposed Convolutional Neural Network (CNN) architecture specifically developed for Optical Character Recognition (OCR) of the Ranjana script. Designed for high accuracy and efficiency, the model processes 64×64 grayscale images through a sequence of convolutional, max pooling, average pooling, and fully connected layers. It includes three convolutional layers for hierarchical feature extraction, followed by pooling layers for spatial reduction, and concludes with dense layers leading to a softmax output over 62 Ranjana character classes. Trained using the Adam optimizer and sparse categorical cross-entropy, Ran2Dev is designed to avoid learning unnecessary patterns from the training data (overfitting) while accurately recognizing characters across multiple classes.

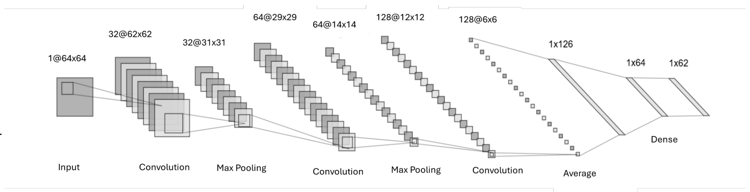
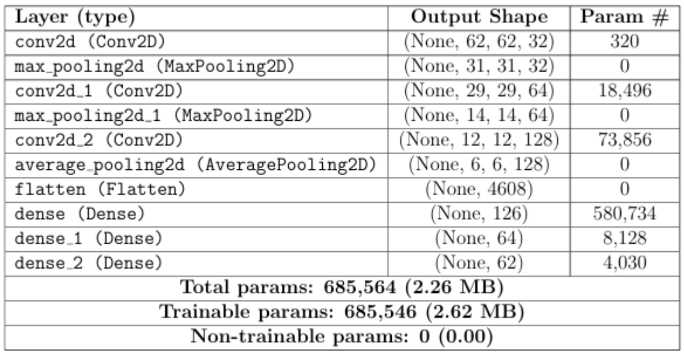


Figure 9: Proposed Ran2Dev CNN-Architecture Showing Layer Wise Structure from Input to Output.



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	320
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	73,856
average_pooling2d (AveragePooling2D)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 126)	580,734
dense_1 (Dense)	(None, 64)	8,128
dense_2 (Dense)	(None, 62)	4,030
Total params: 685,564 (2.26 MB)		
Trainable params: 685,546 (2.62 MB)		
Non-trainable params: 0 (0.00)		

Figure 10: Summary of the Proposed Ran2Dev Model with Layer Types, Output Shapes, and Parameters.

Both models employed the Adam optimizer and sparse categorical cross-entropy loss:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{4}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability.

4. Architectural Justification for Ran2Dev:

- **Higher Input Resolution:** Uses 64×64 grayscale images (vs. 32×32 in LeNet-5) to capture the fine curves and strokes of Ranjana script.
- **Increased Depth:** Three convolutional layers are used for richer and more detailed feature extraction.
- **Hybrid Pooling:** Combines max pooling (keeps the most important details) and average pooling (preserves overall character shape).
- **Modern CNN Consideration:** Heavier models like ResNet or DenseNet were avoided due to high computational demands; Ran2Dev is lightweight, script-specific, and can be trained efficiently even without a GPU.

5. Result and discussion

5.1. Evaluation Metrics

We evaluated the model using different metrics such as accuracy, precision, recall, F1 score, and analysis of the confusion matrix. The evaluations were performed using data divided into training, test, and validation sets with a 70:10:20 split.

5.2. Quantitative Results

The proposed Ran2Dev model demonstrated a peak classification accuracy of 99.74%, outperforming the 99.10% accuracy achieved by LeNet-5. Ran2Dev also maintained consistently higher F1-scores across all classes, indicating improved balance between precision and recall. From epochs 30 to 70, the F1-score, precision, recall, and test accuracy remained consistently above 99.6%, with a brief dip at epoch 60, followed by recovery in subsequent epochs highlighting the robustness and reliability of the model's performance over time.

Epoch	F1 Score	Precision	Recall	Test Accuracy
30	0.986	0.987	0.987	0.987
35	0.980	0.931	0.980	0.980
40	0.987	0.987	0.987	0.987
45	0.995	0.995	0.995	0.995
50	0.991	0.991	0.991	0.991
55	0.992	0.993	0.985	0.992
60	0.998	0.995	0.985	0.985
65	0.995	0.995	0.995	0.995
70	0.996	0.995	0.995	0.996

Table 1: Performance Metrics of the LeNet-5 Model Across Different Epochs Including F1 Score, Precision, Recall, and Test Accuracy.

Epoch	F1 Score	Precision	Recall	Test Accuracy
30	0.996	0.996	0.996	0.996
35	0.996	0.996	0.996	0.996
40	0.997	0.997	0.997	0.997
45	0.997	0.997	0.997	0.997
50	0.997	0.997	0.997	0.997
55	0.997	0.997	0.997	0.997
60	0.986	0.987	0.988	0.986
65	0.992	0.992	0.992	0.992
70	0.997	0.997	0.997	0.997

Table 2: Performance Metrics of the Ran2Dev Model Across Different Epochs Including F1 Score, Precision, Recall, and Test Accuracy.

5.3. Confusion Matrix Analysis

The confusion matrix presented in Fig. 11 demonstrates very few misclassifications, indicating that the Ran2Dev model performs well in accurately distinguishing even closely resembling characters.

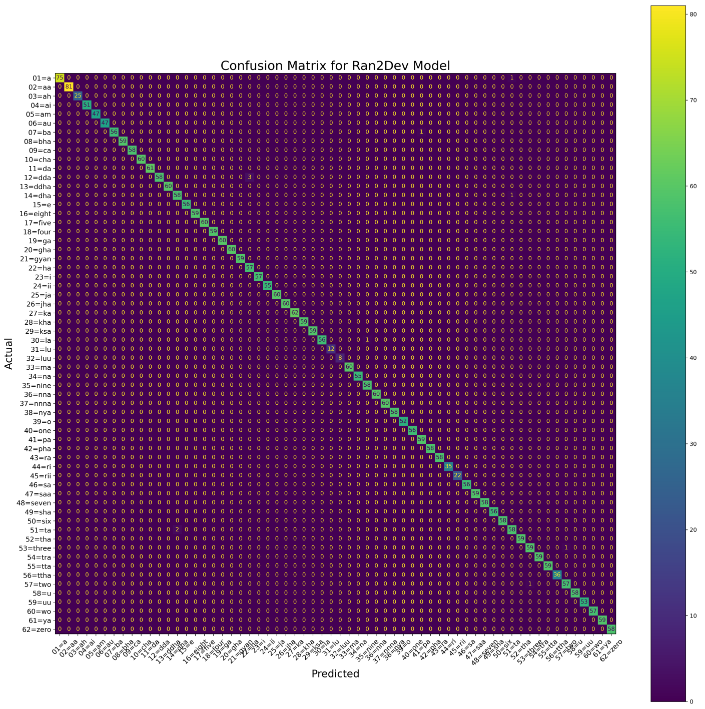


Figure 11: Confusion matrix of the Ran2Dev model showing accurate classification across 62 classes.

5.4. Loss and Accuracy Curves

The training and validation graphs showed that both models learned well and successfully converged. However, Ran2Dev learned a bit faster and ended with lower loss values, likely because its deeper design helps it understand features more effectively.

As illustrated in Fig. 12, the model demonstrated stable learning with little sign of overfitting. The validation accuracy closely matched the training accuracy, and the validation loss steadily decreased, suggesting that the model generalized well to unseen data.

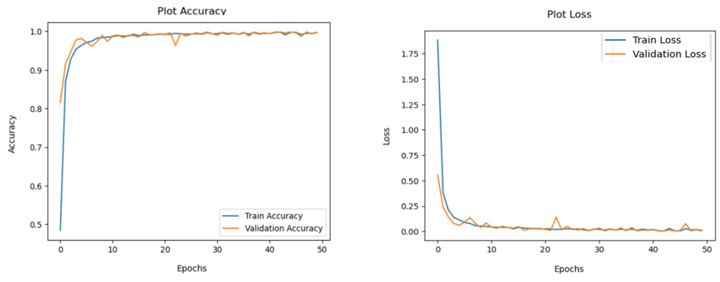


Figure 12: Training and Validation Accuracy and Loss Curves for Ran2Dev Model

6. Conclusion

This study introduced Ran2Dev, a CNN-based OCR system developed for the conversion of handwritten Ranjana Lipi characters into Devanagari script. The proposed model was trained using manually collected and preprocessed character images, utilizing essential image preprocessing techniques such as grayscale conversion, aspect ratio preservation, normalization, binarization (including Otsu and inverse thresholding), and noise removal using Gaussian blur.

To assess its performance, Ran2Dev was compared against a standard LeNet-5 model using various evaluation metrics, including F1-score, precision, recall, and test accuracy. The Ran2Dev model consistently outperformed LeNet-5, reaching a maximum accuracy of 99.74% at an input resolution of 64×64 pixels, in contrast to LeNet-5's 99.10% accuracy at 32×32 pixels. It also maintained consistently high performance across multiple epochs, with several epochs achieving F1-scores of 0.997 and above.

These findings demonstrate Ran2Dev's high reliability and accuracy in recognizing Ranjana Lipi characters, establishing it as a valuable tool for digitizing historical scripts and advancing OCR research in lesser represented writing systems.

6.1. Challenges and limitations

- **Dataset Availability:** Limited dataset availability due to scarcity of handwritten Ranjana samples and few proficient writers.
- **Scope Restriction:** This study focuses only on the recognition of Ranjana Lipi characters without modifiers.
- **Controlled Input Conditions:** The model is trained on images of Ranjana characters that are properly written, vertically aligned, and captured against a plain white background.

6.2. Future Work

- **Extension to Word Level Recognition:** Future research will focus on scaling the current character level recognition system to support word level recognition.
- **Support modifiers:** Future research may focus on supporting characters with modifiers.
- **Diverse Input Conditions:** Train on more diverse datasets with varied backgrounds, orientations, and noise for real world robustness.

References

[1] Explore All About Nepal. History of newari languages in nepal: Culture & significance (2025). URL https://exploreallaboutnepal.com/history-of-newari-languages-in-nepal, [Online].

[2] The Kathmandu Post. A timeless nepali cultural treasure (2025). URL https://kathmandupost.com/art-culture/2025/02/08/a-timeless-nepali-cultural-treasure, [Online].

[3] Central Bureau of Statistics, Nepal. National population and housing census 2021: Caste/ethnicity and mother tongue report (2024). URL https://censusnepal.cbs.gov.np/results/files/result-folder/Caste%20Ethnicity_report_NPHC_2021.pdf, [Online].

[4] Bati J & Dawadi P, Ranjana script handwritten character recognition using cnn, *JOIV : International Journal on Informatics Visualization*, 7.

[5] Mehta D & Mehta P. Devanagari handwritten character recognition using convolutional neural network (2025). URL https://arxiv.org/abs/2507.10398.

[6] Ma J. A study on a hybrid cnn-rnn model for handwritten recognition based on deep learning. In: *Proceedings of the 1st International Conference on Data Analysis and Machine Learning - Volume 1: DAML*. SciTePress, pp. 268–273. https://doi.org/10.5220/0012801000003885.

[7] Damade S R, Adhiya K P & Zinjore R S, Identification of devanagari script from image document, *International Journal of Computer Engineering and Technology (IJCET)*, 4(5) (2013) 224–231.