RECOGNITION AND CONVERSION OF BRAHMI CHARACTERS TO DEVANAGARI CHARACTERS

Gauri Kardekar
School of Computer Engineering and Technology
MIT Academy of Engineering,Alandi, Pune,India
email id: gauri.kardekar@mitaoe.ac.in

Gargi Gundawar
School of Electronics and Telecommunication Engineering
MIT Academy of Engineering,Alandi, Pune,India
email id: gargi.gundawar@mitaoe.ac.in

Shrawani Chaudhari
School of Electronics and Telecommunication Engineering
MIT Academy of Engineering,Alandi, Pune,India
email id: shrawani.chaudhari@mitaoe.ac.in

Sakshi Deshmukh
School of Electronics and Telecommunication Engineering
MIT Academy of Engineering,Alandi, Pune,India
email id: deshmukh.sakshi@mitaoe.ac.in

Vinayak Kulkarni
School of Electronics and Telecommunication Engineering
MIT Academy of Engineering,Alandi, Pune,India
email id: vbkulkarni@mitaoe.ac.in

ABSTRACT

This work explores template matching, a technique for Brahmi character recognition, using Python libraries. The code implements grayscale conversion, image resizing, and template matching to identify a Brahmi letter within an image. While this approach is suitable for specific, controlled scenarios, it may not generalize well to handwritten or degraded characters. Future advancements could involve Convolutional Neural Networks (CNNs) to achieve more robust recognition of Brahmi characters across various writing styles. Additionally, the system could be extended to map recognized Brahmi characters to their corresponding Devanagari counterparts, facilitating translation tasks.

KEYWORDS: Brahmi script recognition, Template matching, Convolutional Neural Networks (CNNs), Devanagari script

I. INTRODUCTION

The Brahmi script, a foundational writing system for many modern South and Southeast Asian languages, holds immense historical and cultural significance. Deciphering ancient Brahmi inscriptions unlocks valuable knowledge about past civilizations. However, manual character recognition from these inscriptions is a time-consuming and expertise-dependent task.

Optical character recognition (OCR) techniques offer a promising solution for efficient Brahmi script analysis. This paper presents a preliminary investigation into Brahmi character recognition using template matching, a fundamental image processing technique.

Template Matching for Brahmi Character Recognition:

Template matching involves comparing an image fragment called template against a larger image to identify potential matches. In the context of Brahmi script recognition, a template representing a specific Brahmi character is compared against a digital image containing an inscription. The code presented in this paper implements template matching to locate and identify Brahmi characters within an image.

Limitations and Future Directions:

While template matching offers a straightforward approach, it has limitations. It performs best with well-preserved, isolated characters and may struggle with variations in character size, orientation, or degradation.

This paper acknowledges these limitations and proposes Convolutional Neural Networks (CNNs) as a potential future direction. CNNs are a powerful deep learning architecture widely used in image recognition tasks. By training a CNN on a large dataset of Brahmi characters, a more robust and versatile recognition system can be achieved.

Furthermore, the system can be extended to map recognized Brahmi characters to their corresponding Devanagari counterparts. This functionality would bridge the gap between ancient and modern scripts, facilitating translation tasks and promoting accessibility of historical texts.

This work contributes to the ongoing research in Brahmi script recognition by exploring the potential of template matching and outlining promising avenues for further development using CNNs and script mapping.

Convolutional Neural Networks (CNNs) for Brahmi Character Recognition:

CNNs have shown effectiveness in image recognition tasks by virtue of their inherent capacity to autonomously acquire hierarchical characteristics from unprocessed input images. Through the training of a Convolutional Neural Network (CNN) using an extensive dataset of Brahmi characters, the system can effectively adjust to differences in script style, noise, and other distortions that are commonly found in historical inscriptions. While template matching necessitates exact matches to predetermined templates, convolutional neural networks (CNNs) acquire the ability to generalize across various instances of each character, so enhancing their resilience and adaptability for practical applications. Advantages of CNN based solution:

- Enhanced management of inscriptions that are noisy, incomplete, or degraded.
- Capability to identify characters written in different sizes, orientations, and handwriting styles.
- Improved scalability enabling the incorporation of additional Brahmi script variations.

II. METHODOLOGY

Image Pre-processing:

The goal of pre-processing is to improve image data by suppressing unwanted distortions or enhancing some image features that are important for subsequent processing, though geometric transformations of images are classified as pre-processing methods here because similar techniques are used.

Effective image pre-processing is essential. The pre-processing techniques used in this study are:

a) Noise removal: Noise reduction aims to eliminate unwanted or random variations in pixel values from an image.

Figure 1: Noise Removal of Brahmi Script on Ashoka Pillar.

(image: Brahmi script on Ashoka Pillar in Sarnath (c. 250 BCE) source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Brahmi_pillar_inscription_in_Sarnath.jpg/270px-Brahmi_pillar_inscription_in_Sarnath.jpg Website accessed on: 10 Oct 2024)

b) Brightness Adjustment: Changes the intensity of all pixels uniformly to improve visibility. To adjust the brightness of an image, we change the value of each pixel by a constant. Adding a positive constant to all of the image's pixel values increases its brightness. Similarly, we can darken the image by subtracting a positive constant from all of its pixel values.

c) Contrast Enhancement: Increases the contrast between the image's lightest and darkest areas, allowing for better feature distinction. Contrast enhancements improve the visibility of objects in a scene by increasing the brightness difference between objects and backgrounds. Contrast enhancements are typically done in two steps: a contrast stretch and a tonal enhancement, though they could be done in one step.

d) Warmth Adjustment: Changes the colour temperature to produce the desired visual effect. Warmth adjustment in image pre-processing changes the colour temperature of an image, making it appear warmer (more red/orange) or cooler (bluer). This technique improves the visual appeal and consistency of images before they are analyzed in applications.

e) Grayscale Conversion: Converts the image from colour to grayscale, simplifying the data and lowering computational complexity.

It eliminates all colour information, leaving only different shades of gray, the brightest being white and the darkest being black.

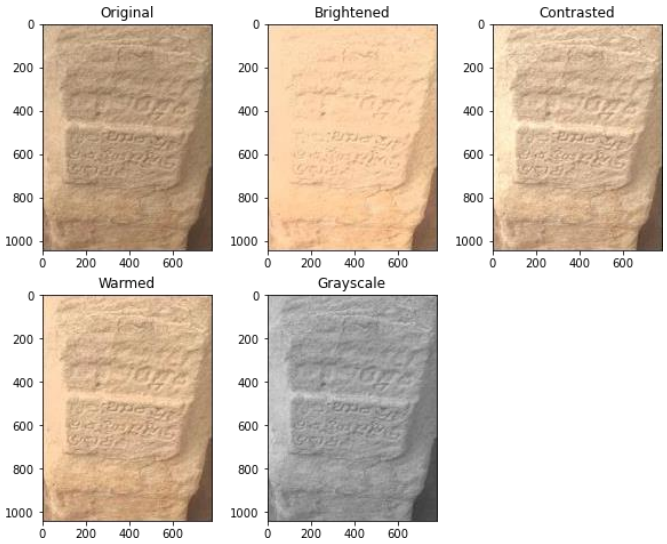

Figure 2: Image pre-processing techniques applied on Brahmi script from Aihole, Karnataka.
(image: Brahmi script from Aihole, Karnataka.
source: captured by us)

f) High Pass Filter: Enhances high-frequency components, such as edges, to bring out fine details. The most common sharpening method is based on a high pass filter. An image is sharpened when the contrast between adjacent areas is increased with little variation in brightness or darkness. A high pass filter tends to keep high frequency information in an image while removing low frequency information.

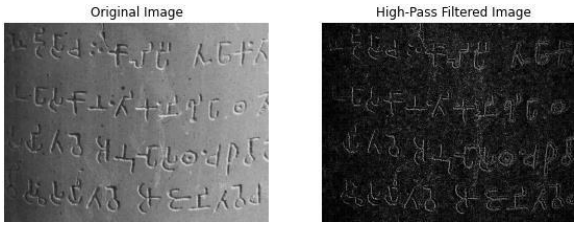

Figure 3: High Pass Filter applied on Brahmi Script on Ashoka Pillar.
(image: Brahmi script on Ashoka Pillar in Sarnath (c. 250 BCE)
source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Brahmi_pillar_inscription_in_Sarnath.jpg/270px-Brahmi_pillar_inscription_in_Sarnath.jpg
Website accessed on: 10 Oct 2024)

g) Low Pass Filter: Smooth the image by reducing high-frequency components, which aids in noise reduction. The most common smoothing method is based on a low pass filter. Smoothing an image involves averaging nearby pixels to reduce the disparity between pixel values. Using a low pass filter tends to keep low frequency information in an image while reducing high frequency data.

Figure 4: Low Pass Filter applied on Brahmi Script on Ashoka Pillar.
(image: Brahmi script on Ashoka Pillar in Sarnath (c. 250 BCE)
source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Brahmi_pillar_inscription_in_Sarnath.jpg/270px-Brahmi_pillar_inscription_in_Sarnath.jpg
Website accessed on: 10 Oct 2024)

h) Edge Detection: Identifies and highlights significant pixel intensity transitions, which help to define object boundaries. This technique is critical for determining the contours of characters in images. It improves the clarity of structural details, resulting in more accurate recognition tasks.

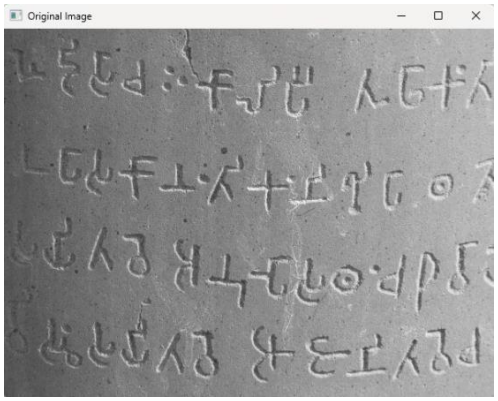

Figure 5: Original Grayscale Brahmi script image of Ashoka Pillar in Sarnath for Edge Detection

(image: Brahmi script on Ashoka Pillar in Sarnath (c. 250 BCE)
source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Brahmi_pillar_inscription_in_Sarnath.jpg/270px-Brahmi_pillar_inscription_in_Sarnath.jpg
Website accessed on: 10 Oct 2024)

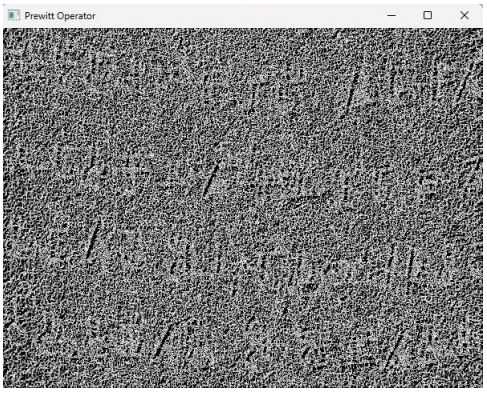

Figure 7: Prewitt Operator applied on Brahmi script of Ashoka Pillar in Sarnath for Edge Detection

(image: Brahmi script on Ashoka Pillar in Sarnath (c. 250 BCE)
source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Brahmi_pillar_inscription_in_Sarnath.jpg/270px-Brahmi_pillar_inscription_in_Sarnath.jpg
Website accessed on: 10 Oct 2024)

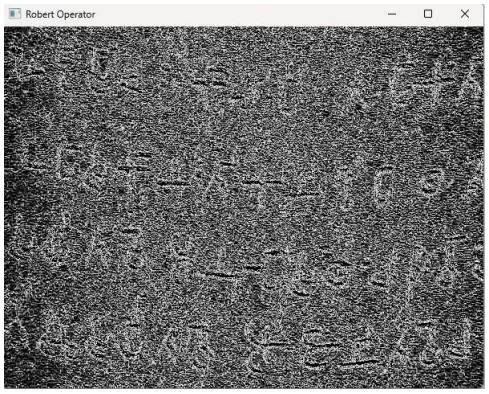

Figure 6: Robert's Operator applied on Brahmi script of Ashoka Pillar in Sarnath for Edge Detection

(image: Brahmi script on Ashoka Pillar in Sarnath (c. 250 BCE)
source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Brahmi_pillar_inscription_in_Sarnath.jpg/270px-Brahmi_pillar_inscription_in_Sarnath.jpg
Website accessed on: 10 Oct 2024)

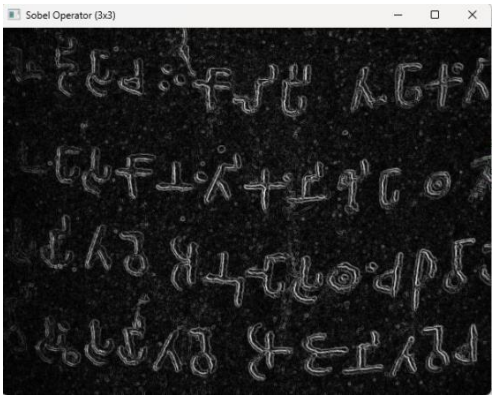

Figure 8: Sobel Operator (3x3) applied on Brahmi script of Ashoka Pillar in Sarnath for Edge Detection

(image: Brahmi script on Ashoka Pillar in Sarnath (c. 250 BCE)
source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Brahmi_pillar_inscription_in_Sarnath.jpg/270px-Brahmi_pillar_inscription_in_Sarnath.jpg
Website accessed on: 10 Oct 2024)

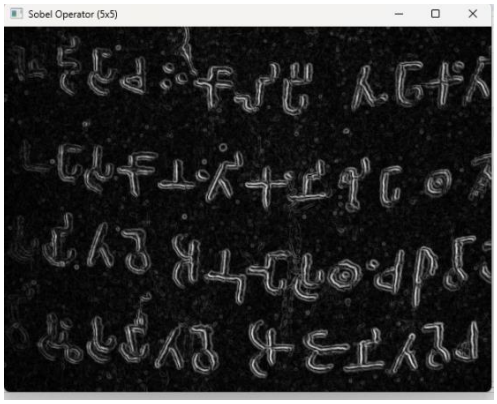

Figure 9: Sobel Operator (5x5) applied on Brahmi script of Ashoka Pillar in Sarnath for Edge Detection

(image: Brahmi script on Ashoka Pillar in Sarnath (c. 250 BCE)
source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Brahmi_pillar_inscription_in_Sarnath.jpg/270px-Brahmi_pillar_inscription_in_Sarnath.jpg
Website accessed on: 10 Oct 2024)

i) Shape detection: Recognizes and separates distinct forms within the image, making it easier to identify letters. It streamlines the recognition process by focusing on the characters' distinguishing geometric features.

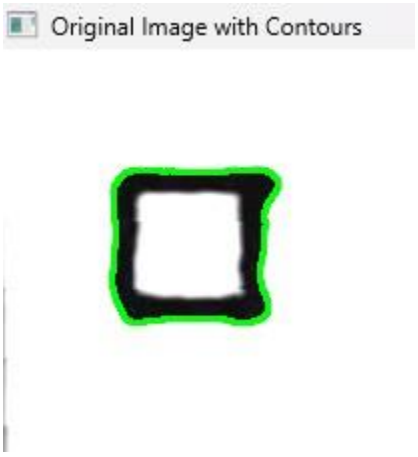

Figure 10: Shape detection of Brahmi Character

(image: 'ब' - 'ba' character in Brahmi
source:from our Brahmi database)

Template Matching:

Template matching is a fundamental image processing technique that detects instances of a template (or pattern) in an input image. Our methodology begins with acquiring digital images of Brahmi inscriptions. These images go through pre-processing steps such as grayscale conversion and template resizing to match the input image's dimensions. We then calculate the normalized cross-correlation between the template and the input image to identify regions with high similarity scores. The coordinates of the maximum correlation point indicate the position of the detected letter in the image.

Steps:

Step 1: Template Creation

1) Data collection: Collect a diverse range of Brahmi characters, including handwriting samples.
2) Template Generation: Create binary images of each Brahmi character to use as templates. The images are labelled with their corresponding Devanagari characters.

Step 2: Matching Process

1) Image pre-processing: Apply basic image pre-processing techniques, noise removal, edge detection, etc., on the input image.
2) Sliding Window: Apply a sliding window approach to move the template over the input image and calculate correlation coefficients at each position.
3) Best Match Selection: Identify the position with the highest correlation coefficient as the best fit for each character.

Step 3: Character Mapping

1) Character Recognition: Use a predefined input image to map Brahmi character to their in the given scripture.
2) Output Generation: Create a window for the input character in the Brahmi script.

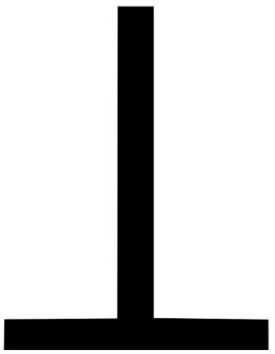

Figure 11: Template used for Template matching technique

(image: Brahmi character
source:https://en.m.wikipedia.org/wiki/File:Japanese_Map_symbol_%28Graveyard%29.svg)

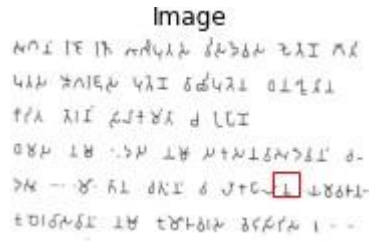

Figure 12: Template matched in Brahmi script
(image: Brahmi script (output of the code executed) source:https://www.designindia.net/thoughts/history/indian-letter-type-design)

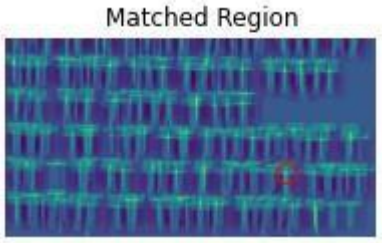

Figure 12: Matched Region in Brahmi script
(image: Brahmi script
source: output of the code executed)

Optical Character Recognition (OCR):
Optical Character Recognition (OCR) is a technology that converts scanned documents into machine-encoded text. It is essential for Brahmi Character Recognition, which converts ancient Brahmi script to modern Devanagari characters. Tesseract, a popular OCR tool, does not support Brahmi script, requiring custom solutions to accurately convert and preserve ancient manuscripts and inscriptions. We discovered this while trying to implement OCR on a Brahmi image. It worked for the English text image but did not produce any results for Brahmi.

Figure 13: English text used for application of OCR technique
(image: English text
source: created by us)

```
1 img = cv2.imread("C:/Users/91766/Downloads/image.png")
```

```
1 print(pytesseract.image_to_string(img))
```

```
Our project title is
Recognition and conversion of
Brahmi Script using ML.
Group Members are:
Shrawani Chaudhari
Gargi Gundawar
Gauri Kardekar
Sakshi Deshmukh
```

Figure 14: OCR result for English text
(image: English text output of OCR technique
source: output image of the code executed by us)

```
print(pytesseract.image_to_string(img))
```

Figure 15: OCR result for Brahmi text
(image: Brahmi text output of OCR technique
source: output image of the code executed by us)

The use of OCR for Brahmi character recognition was hampered by notable limitations, namely the lack of Brahmi script support in most conventional OCR tools, such as Tesseract. These constraints did not allow further advancement, particularly when confronted with diverse handwriting styles, noisy inscriptions, and distorted characters. Therefore, we decided to investigate a deep-learning approach utilizing Convolutional Neural Networks (CNNs), which provides enhanced adaptability and precision in identifying intricate and deteriorated Brahmi script. The use of CNNs enables the system to acquire knowledge from a large database of Brahmi characters, so enhancing its robustness and ability to handle variations that conventional OCR techniques encounter difficulties with.

Data Preparation:

A manually curated dataset of Brahmi and Devanagari characters was created for this research. The Brahmi images were organized into nested subfolders, with each subfolder representing a specific Brahmi character. The corresponding Devanagari characters were placed in a parallel folder structure, ensuring alignment between two scripts. A function load_dataset_from_nested_ subfolder was used to load the image paths and labels from these subfolders.

Label Encoding for Brahmi-to-Devanagari Character Mapping:

For this project, we utilized label encoding to establish a connection between Brahmi characters and their corresponding Devanagari counterparts. Label encoding plays a vital role in transforming categorical character labels into numerical format, enabling our Convolutional Neural Network (CNN) model to effectively classify the characters. The label encoding process facilitated a smooth transition between Brahmi and Devanagari scripts. The system's ability to learn and apply knowledge from the dataset allowed for accurate translation of even intricate characters found in deteriorated inscriptions to their Devanagari equivalents.

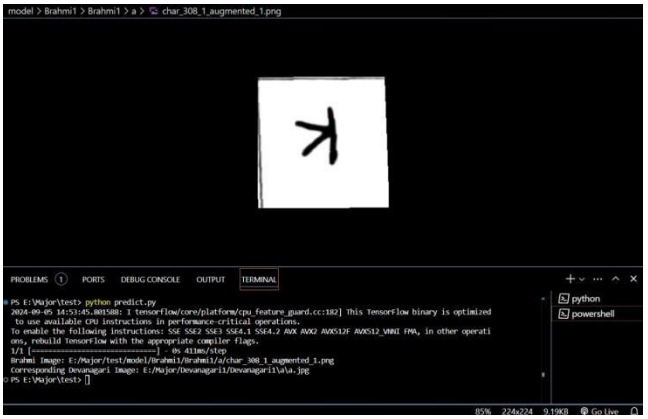

Figure 16: Brahmi Character "अ"

(image: Brahmi character for 'a' or 'अ' source: from our Brahmi database)

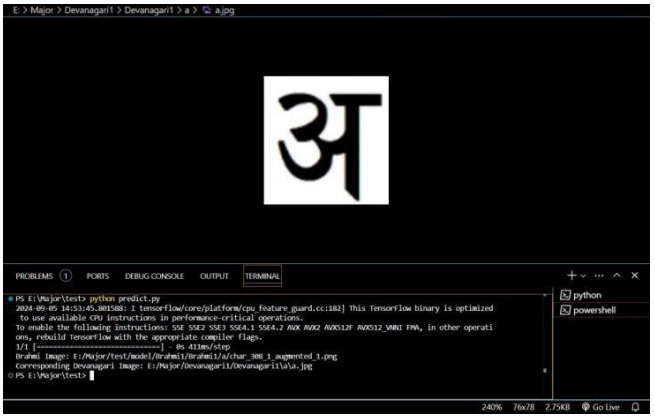

Figure 17: Corresponding Devanagari character Character "अ"

(image: output Devanagari character for 'a' or 'अ' source: from our Devanagari database)

Figure 18: Brahmi Character "आ"

(image: Brahmi character for 'aa' or 'आ' source: from our Brahmi database)

Figure 19: Corresponding Devanagari character Character "आ"

(image: output Devanagari character for 'aa' or 'आ'

source: from our Devanagari database)

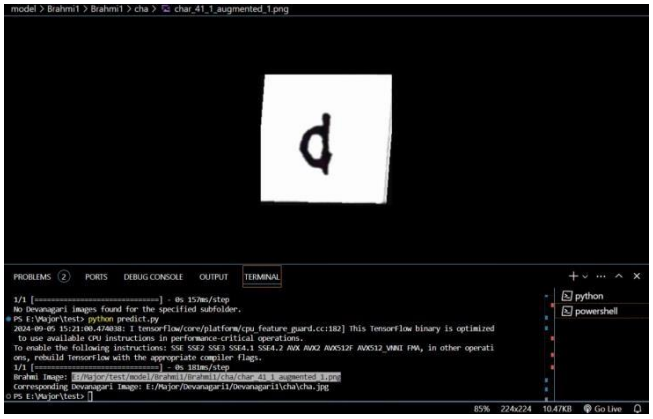

Figure 20: Brahmi Character "च"

(image: Brahmi character for 'cha' or 'च'

source: from our Brahmi database)

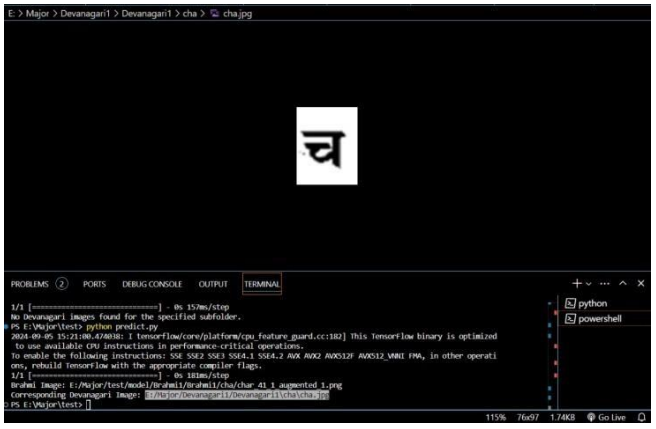

Figure 21: Corresponding Devanagari character Character "च"

(image: output Devanagari character for 'cha' or 'च'

source: from our Devanagari database)

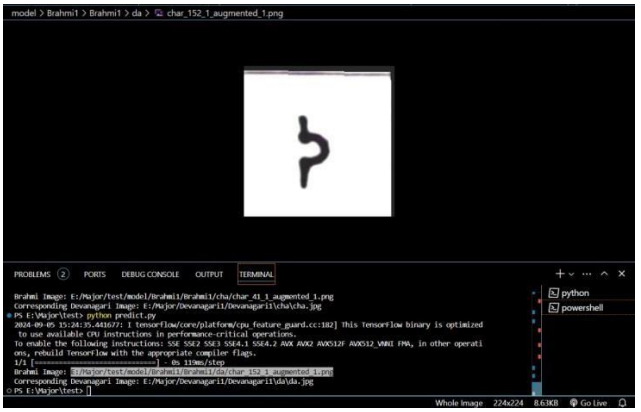

Figure 22: Brahmi Character "ड"

(image: Brahmi character for 'da' or 'ड'

source: from our Brahmi database)

Figure 23: Corresponding Devanagari character Character "ड"

(image: output Devanagari character for 'da' or 'ड'

source: from our Devanagari database)

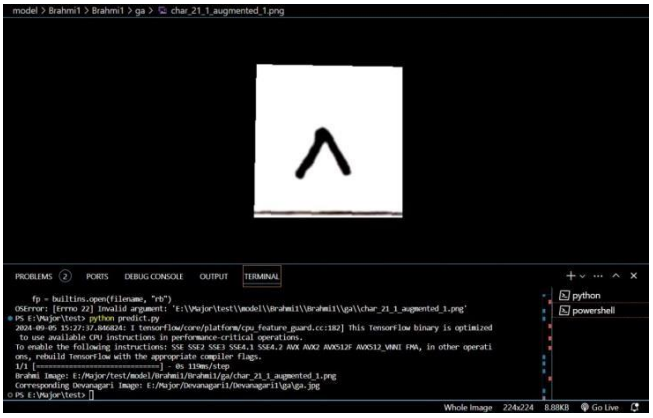

Figure 24: Brahmi Character "ग"

(image: Brahmi character for 'ga' or 'ग'

source: from our Brahmi database)

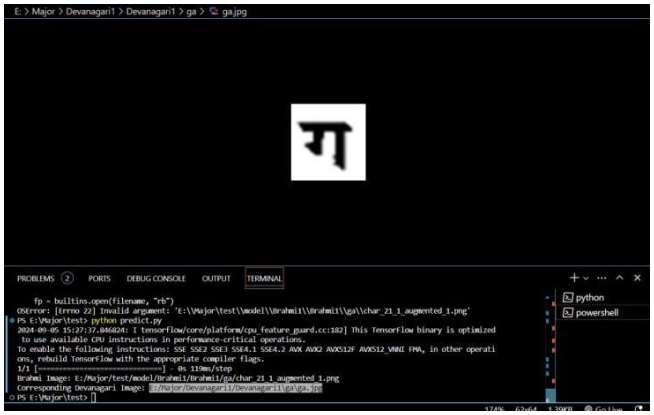

Figure 25: Corresponding Devanagari character Character "ग"

(image: output Devanagari character for 'ga' or 'ग'

source: from our Devanagari database)

Training Process:

Label encoding was performed using a LabelEncoder to convert character labels into numerical representations for the model. The dataset was then split into training and testing sets using train_test_split. The training set was used to train the model, while the testing set was used to evaluate its performance. The model was trained using the Adam optimizer with the sparse_categorical_crossentropy loss function, which is suitable for multi-class classification problems. Accuracy was chosen as the primary metric to monitor the model's performance during training. The model was trained for a specified number of epochs (e.g., 10 epochs) with validation data to monitor performance on unseen data and prevent overfitting.

CNN-based Brahmi-to-Devanagari Character Recognition:

1. CNN Model Architecture:

The architecture of the model follows a sequential design, incorporating the following layers:

a) Convolutional layers: Two convolutional layers are used to extract features from the input images. The first layer has 32 filters, while the second layer has 64 filters. Both layers have a kernel size of 3x3.

b) Max –pooling layers: After each convolutional layer, there are max-pooling layers (2x2) to decrease dimensionality and computation time.

c) Dense Layers: Once the output is flattened, the model applies a dense layer with 128 units and ReLU activation. This is then followed by an output layer that uses sigmoid activation to classify images as either Brahmi or Devanagari characters.

d) The model was trained using the Adam optimizer and a learning rate of 0.001. It utilized binary cross-entropy loss to classify between Brahmi and Devanagari scripts.

2. Data Augmentation and Data Training:

Utilizing 'ImageDataGenerator' allowed for the implementation of data augmentation techniques, enhancing the model's robustness. With the help of various random transformations like rotation, width shift, height shift, zoom, and flipping, the model was able to improve its ability to adapt to different input samples.

Splitting the dataset into training and validation sets allowed for the model to be fitted over 10 epochs, implementing early stopping to prevent over-fitting. The performance of the model was closely monitored using validation data.

3. Image Pre-processing and Label Encoding:

The input images were resized to a resolution of 64x64 pixels and then normalized so that the pixel values ranged from 0 to 1. Mapping Brahmi characters to their corresponding Devanagari counterparts was achieved using label encoding. This encoding guarantees that predictions will be transformed into easily readable Devanagari script once they are classified.

4. Model Deployment:

An application was developed using Flask to enable users to upload images of Brahmi characters. The CNN model then processes these images and converts them into Devanagari script. After pre-processing, the uploaded image is fed into the CNN model, and the character that is predicted is then shown. The application incorporates the model via an API, allowing users to engage with the system and evaluate the translation of Brahmi script to Devanagari.

The CNN model consistently demonstrated a high level of precision in identifying Brahmi characters, effectively associating them with their Devanagari equivalents. Integration of the model into a web-based application enables instantaneous translation, so establishing it as a practical tool for recognizing and preservation of historical texts.

III. RESULT

We tested the Brahmi-to-Devanagari character recognition system on a carefully selected dataset of images written in Brahmi script. The Convolutional Neural Network (CNN) model demonstrated

impressive results, accurately converting Brahmi characters to Devanagari counterparts while effectively handling challenges such as diverse handwriting styles, noise, and character distortions.

IV. CONCLUSION

This research introduces a fresh method for recognizing Brahmi script by utilizing Convolutional Neural Networks (CNNs) to facilitate its translation into Devanagari script. When faced with the challenge of dealing with Brahmi characters, traditional Optical Character Recognition (OCR) methods proved to be inadequate. However, a CNN-based model showcased exceptional accuracy and adaptability, providing a promising solution. Through the implementation of a web-based interface, the project has made notable progress in enhancing the accessibility of historical texts.

Further exploration may involve broadening the dataset to encompass a wider range of Brahmi inscriptions, along with the inclusion of additional ancient scripts for more comprehensive analysis. In addition, improving the model to handle worn-out inscriptions and incorporating the ability to recognize multiple scripts would enhance the system's practicality.

V. ACKNOWLEDGEMENT

We would like to express our gratitude to MIT Academy of Engineering,Alandi,Pune for their support throughout this project. A sincere appreciation is extended to Dr. Diptee Sakhare, the Dean of the School of Electronics and Telecommutation. We especially appreciate guidance and support of our project guide, Professor Vinayak Kulkarni. Additionally, we would like to express our gratitude to all faculty members for their support.

VI. REFERENCES

[1] How to study Brahmi coins: Brahmi Teacher. Author: Pradip Dattuji Wankar.

[2] Bora, M. B., Daimary, D., Amitab, K., & Kandar, D. (2020). Handwritten character recognition from images using CNN-ECOC. Procedia Computer Science, 167, 2403-2409.

[3] Moudgil, A., Singh, S., Gautam, V., Rani, S., & Shah, S. H. (2023). Handwritten devanagari manuscript characters recognition using capsnet. International Journal of Cognitive Computing in Engineering, 4, 47-54.

[4] Nagane, A. S., Patil, C. H., & Mali, S. M. (2023, January). Classification of Brahmi script characters using HOG features and multiclass error-correcting output codes (ECOC) model containing SVM binary learners. In 2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE) (pp. 448-451). IEEE.

[5] Gautam, N., Chai, S. S., & Jose, J. (2020). Recognition of Brahmi words by using deep convolutional neural network.

[6] Gautam, N., Chai, S. S., Afrin, S., & Jose, J. (2020). Brahmi word recognition by supervised techniques.

[7] Nagane, A. S., & Mali, S. M. (2020). Segmentation of characters from degraded Brahmi script images. In Applied Computer Vision and Image Processing: Proceedings of ICCET 2020, Volume 1 (pp. 326-338). Springer Singapore.

[8] Sharma, A. (2019). Devanagari Online Handwritten Character Recognition (Doctoral dissertation, Ph. D. Dissertation. Indian Institute of Science, Bangaluru, Karnataka, India).

[9] Mali, S. M. Aniket S. Nagane.

[10] Agrawal, M., Chauhan, B., & Agrawal, T. (2022). Machine learning algorithms for handwritten Devanagari character recognition: a systematic review. Journal of Science & Technology (JST), 7(1), 1-16.

[11] Gautam, N., Chai, S. S., & Gautam, M. (2020). The Dataset for Printed Brahmi Word Recognition. In Micro-Electronics and Telecommunication Engineering: Proceedings of 3rd ICMETE 2019 (pp. 125-133). Springer Singapore.

[12] Munivel, M., & Enigo, V. F. (2022). Optical Character Recognition for Printed Tamizhi

Documents using Deep Neural Networks. DESIDOC Journal of Library & Information Technology, 42(4), 227-233.

[13] Bhuvaneswari, S., & Kathiravan, K. (2024). ScriptSpecific Character Recognition: A Deep Learning Framework for Analyzing Tamil Ancient Inscriptions in Temples.

[14] Subadivya, S., Vigneswari, J., Yaminie, M., & Diviya, M. (2020). Tamilbrahmi script character recognition system using deep learning technique. International Journal of Computer Science and Mobile Computing, 9(6), 114-119.

[15] Roy, A. Typographical Investigation of Mauryan Brahmi.

[16] Wolberg, G. (1987). A syntactic omni-font character recognition system. International Journal of Pattern Recognition and Artificial Intelligence, 1(03n04), 303- 322.

[17] SRIRAM, S. Virtual Reality Based Enhanced Brahmi Inscription Script Recognition.

[18] Daggumati, S., & Revesz, P. Z. (2023). Convolutional Neural Networks Analysis Reveals Three Possible Sources of Bronze Age Writings between Greece and India. Information 2023, 14, 227.

[19] Ruwanmini, D. A. S., Liyanage, K. V., Karunarathne, K. G. N. D., Dias, G. K. A., & Nandasara, S. T. (2016). An architecture for an inscription recognition system for sinhala epigraphy. International Journal of Research-Granthaalayah, 4, 48-64.

[20] Tomar, A., Choudhary, M., & Yerpude, A. (2015). Ancient Indian scripts image pre-processing and dimensionality reduction for feature extraction and classification: a survey. International Journal of Computer Trends and Technology (IJCTT), 21(2), 101- 124.