Hindi Script Refinement & Improved OCR

BS-2342 Satyaprakash Sahdev Jaiswar
BS-2344 Saurabh Bhikan Nankar
BS-2308 Aman Singh
BS-2353 Subhadip Roy
BS-2317 Aryan Sahu
BS-2335 Pallav Sahay

April 9, 2024

Abstract

Hindi, one of the most widely spoken languages globally, presents unique challenges for Optical Character Recognition (OCR) due to its complex script. This project addresses the issue by proposing a new set of 60 characters optimized for ease of writing and mechanical differentiation, enhancing OCR efficiency. Inspired by simple shapes, limited complexity, and consistent distinctive features, these characters aim to streamline both manual writing and automated recognition processes. Leveraging the K-nearest neighbors (KNN) classification algorithm, a model is trained on a dataset comprising handwritten characters contributed by the project team. Preprocessing and segmentation utilize established algorithms within the Python OpenCV module. The resulting OCR application demonstrates improved accuracy and usability for Hindi text recognition, marking a significant advancement in bridging the gap between manual and digital Hindi writing systems.

1 Introduction

The Hindi language, one of the most widely spoken languages globally, poses unique challenges for optical character recognition (OCR) due to its complex script. While Hindi is rich in cultural heritage and widely used in literature, administration, education, and everyday communication, its intricacies present obstacles for mechanical interpretation.

In this project, we address the challenges of OCR for Hindi by proposing a novel set of characters designed to enhance ease of writing and mechanical distinguishability. Recognizing the significance of simplicity in character design, we meticulously crafted a set of 60 symbols tailored to streamline both writing and OCR processes for Hindi.

Our approach aimed to bridge the gap between traditional script complexity and modern computational requirements. By introducing characters that are intuitive to write and easily discernible by machines, we sought to revolutionize the OCR landscape for Hindi language applications.

The culmination of our efforts resulted in the development of an OCR application specifically tailored for the newly designed Hindi characters. This application promises to significantly enhance the efficiency and accuracy of Hindi text recognition, opening doors to a myriad of applications across various domains.

Through this project, we aim to contribute to the advancement of Hindi language technology and facilitate greater accessibility and usability of Hindi content in the digital age.

PART 1: Character Design

The Devanagari script, used for writing Hindi among other languages, presents challenges in both writing and recognition due to its intricate characters and writing rules. Traditionally, writing Hindi involves sketching a guiding line followed by the characters, often resulting in curvy and complex shapes that require refinement.

To address these challenges, our design considerations for creating a new set of Hindi characters were as follows:

1. **Simple Shapes:** We aimed to utilize basic geometric shapes such as lines, circles, squares, and triangles, as well as their combinations. This simplification minimizes the complexity of characters, making them easier to distinguish in scans and write by hand.

2. **Distinctive Features:** Each symbol was designed with distinctive features to ensure easy differentiation from others. Variations in shape, size, orientation, or strokes were incorporated to enhance readability and recognition.

3. **Consistency:** Consistency in design across the alphabet was crucial. We maintained uniform stroke width and proportions to ensure that each symbol is unique and recognizable, aiding both writing and scanning processes.

4. **Limited Complexity:** Keeping the overall complexity of the alphabet low was essential for ease of learning and memorization. This approach also contributes to improved accuracy in both writing and scanning.

5. **Legibility:** We prioritized legibility by avoiding symbols that are too similar or easily confused. Clear differentiation between characters is vital for accurate writing and scanning.

6. **Test and Refine:** The alphabet underwent rigorous testing with various handwriting styles and scanning methods to validate its efficacy. Feedback and usability testing guided continuous refinement to meet desired criteria.

7. **Size Consistency:** Maintaining a consistent size for all symbols was crucial to prevent confusion during interpretation and ensure uniformity across the alphabet.

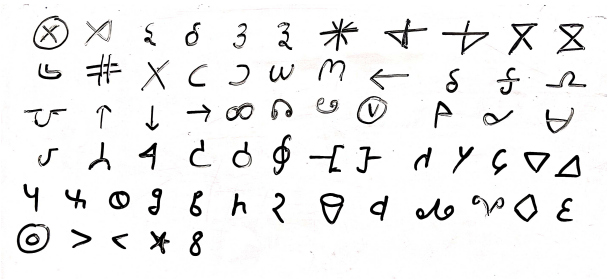

Figure 1: Prototype Characters for Hindi Language

After initial prototyping and refinement, our final set of characters was achieved (image to be shown later). While some unworthy characters were discarded during the iterative design process, the finalized characters represent a significant improvement in both ease of writing and mechanical differentiation.

Next, we proceed to create a training dataset and develop an OCR application tailored to these newly designed Hindi characters.

Part 2: Training OCR Model

After finalizing the characters, we created a diverse training dataset by writing each character multiple times to capture various handwriting styles and variations. This dataset serves as the foundation for training our OCR model, ensuring robustness and accuracy. By exposing the model to a wide range of handwriting variations, we aim to enhance its capability to accurately recognize characters in real-world scenarios. In the next phase, we utilize this dataset to train our OCR model, focusing on preprocessing, feature selection, and algorithm optimization to achieve superior performance and accuracy in recognizing handwritten Hindi text.

I: Preprocessing Function

After finalizing the design of the characters, we proceeded with the development of the OCR application. The first step in this process involved creating a preprocessing function. This function is crucial as it refines the input image to contain only the necessary features required for OCR. The preprocessing function consists of the following steps:

1. **Resize:** Resizing the image ensures uniformity in processing and reduces computational complexity.

2. **Deskew:** Correcting the skew of the image aligns the text horizontally, facilitating character recognition.

3. **Adjust Brightness and Contrast:** Enhancing brightness and contrast improves image quality, aiding in text legibility.

4. **Thresholding:** Converting the image into a binary format simplifies further processing by separating text from the background.

5. **Greyscale:** Converting the image to greyscale removes color information while preserving essential text features.

6. **Binarization:** Converting the image to black and white enhances contrast, making text stand out from the background.

7. **Noise Removal:** Removing noise improves OCR accuracy by eliminating irrelevant details.

8. **Erosion and Dilation:** Morphological operations shrink or expand text regions to improve text connectivity and remove small artifacts.

These preprocessing steps collectively prepare the image for OCR, enhancing text visibility and reducing background noise to improve character recognition accuracy.

Next, we proceed to implement the OCR algorithm and integrate it into the application workflow.

II: Segmentation Function

Following preprocessing, the next step in the OCR pipeline is segmentation, which involves identifying individual characters in the image. This is achieved using a segmentation function that utilizes contour detection.

Contours are essentially outlines or boundaries of objects in an image. In the context of OCR, contours represent the boundaries of individual characters. They are calculated using algorithms such as the Suzuki algorithm or the Moore-Neighbor Tracing algorithm.

Here's how contours are calculated and how they help in finding characters in the binarized image:

1. **Contour Detection:** Contour detection algorithms analyze the binarized image to identify connected regions of similar intensity. These regions correspond to the boundaries of characters in the image.

2. **Bounding Boxes:** Once contours are detected, bounding boxes are drawn around them to encapsulate individual characters. These bounding boxes serve as regions of interest for character extraction.

3. **Character Extraction:** The segmented characters are extracted from the original image based on the bounding boxes. Each bounding box represents a separate character that can be isolated for further processing.

By leveraging contour detection and bounding box techniques, the segmentation function effectively identifies and isolates individual characters in the binarized image. This step is crucial for creating the training dataset required for training the OCR model.

Next, we proceed to extract characters from the segmented regions and prepare the training data for the OCR model.

III: Training using KNN

In the training phase of the OCR pipeline, the K-nearest neighbors (KNN) algorithm is employed for classification. KNN is a simple and intuitive machine learning algorithm used for classification tasks. It operates on the principle of proximity, where an unknown sample is classified based on the majority class of its K nearest neighbors in the feature space.

Image Space: In the context of OCR, the image space refers to the feature space where images are represented as vectors of features. Each feature represents a characteristic of the image that aids in classification.

Using KNN to Classify Images: KNN can be used to classify images by first extracting relevant features from the images, such as the Histogram of Oriented Gradients (HOG), and then representing each image as a vector in the feature space. During classification, KNN calculates the distance between the test image and all training images in the feature space and assigns the test image to the majority class of its nearest neighbors.

Histogram of Oriented Gradients (HOG): HOG is a feature descriptor technique used to extract relevant information from images. It calculates the distribution of gradient orientations in localized regions of the image, capturing the shape and texture characteristics. HOG features are calculated by dividing the image into small cells, computing gradient orientations within each cell, and then concatenating the histograms of these orientations to form a feature vector.

SMOTE (Synthetic Minority Over-sampling Technique): SMOTE is a technique used to address class imbalance in the training dataset. It synthesizes new minority class samples by interpolating between existing samples, thereby balancing the class distribution. This is essential for improving the performance of the classifier, especially in scenarios where one class is significantly underrepresented.

Joblib File: Joblib is a library in Python used for serialization and deserialization of Python objects. It is particularly useful for saving machine learning models to disk, allowing for easy retrieval and reuse. The trained KNN model is dumped to a joblib file for later use in predicting characters from new images.

Next, we proceed to implement the trained KNN model in the OCR application for character recognition.

PART 3: OCR Application

After training the OCR model and saving it to a joblib file, we developed an OCR application to provide a user-friendly interface for character recognition. The application takes an input image as its input and generates two output images:

1. Output Image with Predicted Characters: This image replaces the characters in the original input image with the predicted characters obtained from the OCR model.

2. Output Image with Bounding Boxes: This image overlays bounding boxes around each identified character in the input image and provides the respective Hindi character below each box in text format.

The OCR application utilizes the previously developed functions and codes for preprocessing, segmentation, and character recognition.

Below are the results of evaluating the OCR model using the confusion matrix and classification report:

Confusion Matrix:

Classification Report:

2 Results

Now, we provide some sample outputs of the program for reference:

These outputs demonstrate the effectiveness of the OCR application in accurately recognizing characters from input images.

Next, we plan to further enhance the OCR application by incorporating additional features and improving its usability.

3 Conclusions

In this project, we embarked on a journey to enhance the Optical Character Recognition (OCR) capabilities for the Hindi language. Recognizing the challenges posed by the intricate Devanagari script, we began by redesigning a set of characters optimized for ease of writing and mechanical differentiation. Through careful consideration of design principles such as simplicity, distinctiveness, and consistency, we developed a new set of characters that streamline both manual writing and automated recognition processes.

With our characters finalized, we proceeded to create a comprehensive training dataset by capturing diverse handwriting styles and variations for each character. This dataset served as the foundation for training our OCR model, aiming to achieve robustness and accuracy in recognizing handwritten Hindi text. Leveraging preprocessing techniques, feature selection, and algorithm optimization, we trained our model to accurately classify and recognize characters across various contexts and scenarios.

Throughout the project, our focus remained on bridging the gap between manual and digital Hindi writing systems. By developing an OCR model tailored to our newly designed characters, we aimed to enable seamless and accurate recognition of handwritten Hindi text, thereby facilitating digital transformation and accessibility in Hindi language processing.

In conclusion, our project represents a significant step forward in enhancing OCR capabilities for the Hindi language, contributing to advancements in digital literacy and accessibility for Hindi speakers worldwide. As we continue to refine and optimize our OCR model, we envision a future where handwritten Hindi text can be effortlessly and accurately digitized, opening up new opportunities for communication, education, and information dissemination.

Further Improvements

While our current OCR model shows promising results, there are several avenues for further improvement and enhancement:

1. **Exploration of Different Classification Algorithms:** While the K-nearest neighbors (KNN) algorithm was chosen for its simplicity and effectiveness, exploring other classification algorithms such as Support Vector Machines (SVM), Random Forests, or Deep Learning models like Convolutional Neural Networks (CNNs) could lead to improved accuracy and performance. CNNs, in particular, have shown remarkable success in image classification tasks due to their ability to learn hierarchical features directly from pixel data.

2. **Incorporation of Line and Word Detection:** By implementing techniques such as vertical and horizontal histogram projections, we can enhance our OCR system to detect lines and words within the text. This would allow for more accurate segmentation and recognition of individual words, improving overall accuracy and readability of the extracted text.

3. **Language Formatting and Spell Checking:** Integration of language formatting and spell checking functionalities would enable the OCR system to output words and sentences in the correct language format. Additionally, incorporating a dictionary to identify and correct misspelled words or words incorrectly identified by the OCR would further enhance the accuracy and usability of the system.

4. **Utilization of Designed Characters for Faster and Easier Writing:** The newly designed characters aim to make Hindi writing faster, easier, and more distinguishable. By incorporating these characters into our OCR system, we can streamline the process of Hindi text input and make it more accessible to a wider audience.

5. **Extension to Different Languages and Fonts:** Finally, the OCR system can be extended to support different languages and fonts, allowing for broader applicability and usability. By adapting the system to recognize characters from various languages and fonts, we can cater to diverse user needs and facilitate text recognition in multilingual environments.

Incorporating these improvements would not only enhance the performance and accuracy of our OCR system but also extend its usability and applicability to a wider range of scenarios and languages. By continuously refining and evolving the system, we can pave the way for advancements in text recognition technology and contribute to the accessibility and digital transformation of linguistic diversity.

4 Reference

1. `https://medium.com/analytics-vidhya/ocr-with-machine-learning-55c7d082fe78`

2. `https://github.com/wjbmattingly/ocr_python_textbook/blob/main/02_02_working%20with%20opencv.ipynb`

3. https://sanaazahra.medium.com/ocr-segmentation-with-python-code-f3251114ee48
4. https://git.sr.ht/~vladh/clumsycomputer/tree/main/item/from-scratch-2-ocr
5. https://towardsdatascience.com/segmentation-in-ocr-10de176cf373