A Case Study of Handwritten Text Recognition from Early Modern Sanskrit Manuscripts

Kartik Chincholikar*, Shagun Dwivedi*, Kaushik Gopalan, and Tarinee Awasthi
FLAME University, Pune 412115, Maharashtra, India
kartik.chincholikar@flame.edu.in, shagun.dwivedi@flame.edu.in
kaushik.gopalan@flame.edu.in, tarinee.awasthi@flame.edu.in
*Equal contribution

Abstract

Extracting text from handwritten documents and converting it to a digital form, commonly called Handwritten Text Recognition (HTR), opens up new ways to access and study scanned texts. In this case study, we perform HTR on Sanskrit manuscripts from the Early Modern period, namely *Vādakautūhala* of Svāmiśāstrin and Bhāskararāya (early eighteenth century), and *Mahāvākyārtha* and *Dvādaśamahāvākyārthavicāra* of unknown authorship. The first is a Pūrva Mīmāṃsā text from the 18th century, while the other two are texts on the tradition of Vedānta. The proposed HTR method consists of three steps: line segmentation, line recognition, and post-correction. For segmenting line images from page images, we propose an approach that uses the scene text detection method Character Region Awareness for Text Detection (CRAFT) to detect individual characters and applies customised logic to segment them into discrete lines of text. To recognise text content from the segmented line images, we fine-tune a pre-trained recognition model for the Devanāgarī script provided by the EasyOCR library. The post-correction model, which makes language-aware corrections to the recognition model outputs, is a fine-tuned version of ByT5-Sanskrit, a pre-trained language model for Sanskrit. We observe that the proposed method performs better than out-of-the-box HTR solutions by addressing problems that are unique to older Sanskrit manuscripts, such as cramped and irregular line spacing, changes in the script over centuries, and stylistic differences in writing due to location, period, and scribe.

1 Introduction

Handwritten Text Recognition (HTR) from historical manuscripts and converting the text to a digital form allows researchers to efficiently search the manuscripts, track changes in word usage over time, or simply count the frequency with which certain ideas appear. This allows for a deeper comprehension of a period's intellectual culture in a manner that would not be feasible otherwise (Tabrizi, 2008; Chandna et al., 2016).

In this work, we consider three Sanskrit manuscripts from the Early Modern era. The first of these is the 517-page manuscript *Vādakautūhala* (Svāmiśāstrin and Bhāskararāya, 1721), whose title translates as "Delight in Dispute" (hereafter referred as VK). This is a text in the school of Mīmāṃsā, a discipline concerned with the analysis of Vedic statements. The text is concerned with a technical issue within Mīmāṃsā, and reveals, amongst other things, the vibrancy and sophistication of Sanskrit intellectual culture as late as the eighteenth century.[1] The other two manuscripts are concerned with *mahāvākyas*, or "Great Sentences," particularly as applied in certain kinds of Vedānta (Uskokov, 2018), where the term refers to identity statements stating, for instance, that "This self is the supreme reality." Both texts are affiliated with Śaṅkarācārya's school of Vedānta, are of unknown authorship and bear no date. One is a 16-page manuscript called *Mahāvākyārtha* (Unknown, ndb), which translates to "Meaning of the Great Sentences"

[1] Lawrence J McCrea and Tarinee Awasthi are currently preparing the first installment of an edition and translation of the text.

(hereafter MV) and the other is a 44-page manuscript called *Dvādaśamahāvākyavicāra* (Unknown, nda), which translates to "Examination of Twelve Great Sentences" (hereafter DMV). The text defends the nondual interpretation of the Great Sentences. All three texts are part of the Lalchand Research Library Collection[2], which was formerly in Lahore and is now housed in the DAV College in Chandigarh.

Although there have been significant advances in techniques to digitise text from handwritten documents over the decades, challenges remain. Primarily, modern HTR techniques have difficulty generalising across the diversity of layouts and complexity of template structures found in various types of documents, each having its own reading order (left to right, right to left, top to bottom, bottom to top, or boustrophedon), footnotes, scribbles, and doodles (Kiessling et al., 2024; Nikolaidou et al., 2022; Janes et al., 2021; Kiessling, 2020; Simistira et al., 2016).

Furthermore, the Devanāgari script, which is used to write Sanskrit, Hindi, Marathi, and several other languages, presents unique challenges during HTR due to their inherent structural complexity and the extensive use of conjunct consonants. Each character in Devanāgari can have multiple forms depending on its position in a word: whether it stands alone, begins a word, or forms part of a conjunct. Sanskrit contains over a thousand bi-consonantal conjuncts and several more conjuncts of three or more consonants. Further, the script features characters that combine vertically and horizontally, creating visually distinct ligatures that lead to numerous potential unique glyphs that an HTR technology is required to recognise.

In addition, the historical and stylistic variations in manuscripts introduce additional layers of complexity. Calligraphy can vary widely between regions, periods, and individual scribes, resulting in variations in script style, size, and ornamentation. These discrepancies pose a challenge for traditional pattern recognition algorithms, which rely on consistency to match scanned images with known characters. Furthermore, the degradation of manuscripts, including fading ink, smudges, and paper wear, can obscure characters and make it difficult to perform handwritten text recognition. AlKendi et al. (2024) provide a comprehensive survey of the complexities and hurdles of digitising text from historical manuscripts.

In this case study, we overcome the challenges faced in performing HTR to digitise text from the aforementioned Sanskrit manuscripts through a combination of deep learning, deterministic logic, and domain-specific requirements of the use case (Philips and Tabrizi, 2020; Cutting and Cutting-Decelle, 2021; Christy et al., 2017).

2 Literature Review

The recorded history of Handwritten Text Recognition (and Optical Character Recognition) dates back to the 19th century, when patents were filed for reading devices to aid blind people (Schantz, 1982). In the 20th century, the discipline evolved to develop character readers for standardised documents. At this point, the standard approach to character recognition was based on template matching and the use of handcrafted feature detectors, along with making use of lexical information about the language (Sethi and Chatterjee, 1977; Sarkar et al., 2015; Kant and Vyavahare, 2016; Pal and Chaudhuri, 2004). Bansal and Sinha (2001) successfully handled both character segmentation and character recognition of printed Devanāgari documents (which included conjunct characters) using an intricate rule-based logic derived from domain knowledge of the Devanāgari script. More recent methods (Acharya et al., 2015; Deore and Pravin, 2020; Moudgil et al., 2023; Bhardwaj and others, 2020; Mohite and Shelke, 2018; Chavan et al., 2014), segment document images into isolated characters and then train models to recognise those characters.

Segmenting lines and words into isolated characters and sub-characters before recognition, however, is a common source of error when digitising Devanāgarī text because of the complex nature of the Devanāgari script, which contains more than 800 possible character shapes, which are made up of combinations of a base character set of 127. Each of the consonant-consonant and

[2] `https://dav.splrarebooks.com/`

consonant-vowel combinations takes different forms based on their position in the word (Karayil et al., 2015). To tackle this issue, Sankaran et al. (2013) and Karayil et al. (2015) built on the work of Graves et al. (2007) and removed the need to segment document images into isolated words and characters using BiLSTMs (Bi-Directional Long Short Term Memory Networks) and CTC (Connectionist Temporal Classification) loss. In these papers, the neural network takes an image of a line of printed text as input and predicts a sequence of Devanāgari Unicode code points as its output. First, the normalised input line image is converted into a sequence of vectors by splitting the image into individual columns. These columns are then sequentially fed to a BiLSTM, which bidirectionally learns the long-term dependencies between the columns of the image. Shi et al. (2016) introduced an end-to-end trainable neural network architecture (CNN-BiLSTM-CTC) which naturally handles sequences of arbitrary lengths. One difference in (Shi et al., 2016) is that they first pass the input line image through a CNN (Convolutional Neural Network) to get a rich feature map, which is then split into individual columns and passed to the BiLSTM. Krishnan and Jawahar (2019) introduced a handwritten word dataset and benchmarked it using a similar CNN-RNN hybrid neural network architecture. The work presented by Dwivedi et al. (2020) presents an attention-based CNN-LSTM architecture that is trained on a combination of synthetic and real data to recognise Sanskrit text of arbitrary length from line images. They also introduced a dataset of Sanskrit manuscripts annotated at the line level. To summarise, owing to the developments discussed above, it is now not required to segment individual words and characters; however, extracting line images from a manuscript page is still a necessity. In this paper, we fine-tune a pre-trained recognition model (CNN-BiLSTM-CTC) for Devanāgarī Script provided by EasyOCR[3].

The aforementioned developments in line recognition have led to line segmentation being a prerequisite for recognising the textual contents of historical pages. Early approaches to text line segmentation often utilised projection profiles. A projection profile is a histogram we get when we add up the pixel values of an image along the x-axis (horizontal projection profile) or the y-axis (vertical projection profile). Projection profile approaches work best when the lines are straight and parallel to the projection axis (Alberti et al., 2019; Chamchong and Fung, 2012); however, they have difficulty segmenting lines from complex layouts and when the lines are curved and with less line spacing. In addition to this, historical manuscripts often suffer from degradation, which can introduce noise and make it difficult to distinguish text from the background, further hindering the effectiveness of projection profiles (Nguyen et al., 2022; Alberti et al., 2019). Character Region Awareness for Text Detection (CRAFT) (Baek et al., 2019) is an approach that builds on the U-Net architecture (Ronneberger et al., 2015) for scene text detection from natural images. CRAFT works by predicting two types of scores: a region score, which predicts the likelihood of each pixel belonging to a character region, and an affinity score, which predicts the likelihood of there being a link between adjacent characters. These scores are then used to identify individual characters and group them into words or text lines. Although CRAFT is not specifically meant for segmenting line images from historical manuscripts, it has been observed to be flexibly adaptable to perform similar tasks (Leow et al., 2023; Shtok et al., 2021; Phung et al., 2020).

Notable recent strategies to segment line images specifically from historical Sanskrit manuscripts include Jindal and Ghosh (2023), Palmira (Sharan et al., 2021) and SeamFormer (Vadlamudi et al., 2023). Sharan et al. (2021) introduced the Indiscapes2 dataset, a significantly more diverse dataset of handwritten Indic manuscripts than its predecessor. They also present a novel, deformation-aware, robust instance segmentation method for regions in handwritten manuscripts. Jindal and Ghosh (2023) were the first to use a faster region-convolution neural network (R-CNN)-based method to robustly segment text lines from an ancient handwritten document in Devanāgarī script. SeamFormer (Vadlamudi et al., 2023) is a high-precision approach to segment text lines from handwritten Sanskrit manuscripts. The

[3] `https://www.jaided.ai/easyocr/`

SeamFormer pipeline comprises of two distinct stages. In the first stage, a multi-task Transformer processes the manuscript image to produce a Binarized Manuscript Image and a Scribble Map (coarse line identifiers called "scribbles" which represent approximate medial axes of the text lines). The next stage generates tight-fitting line segmentation polygons by utilising the scribble maps and feature maps derived from the binarized image. Other notable line segmentation methods that are not specific to the Devanāgarī script are LCG (Alberti et al., 2019), Nguyen et al. (2022) and Kraken (Kiessling, 2020; Kiessling, 2019).

Once text content from segmented line images is converted into editable, digitally encoded text, it is passed to the post-correction stage, where Sanskrit experts make further corrections to the automatically recognised text. Comprehensive Indic post-correction frameworks and methods such as OpenOCRCorrect (Saluja et al., 2017a; Saluja et al., 2017b; Adiga et al., 2018), (Das, 2021) and (Vinitha and Jawahar, 2016) have been developed to assist human annotators by creating domain-specific dictionaries on the fly, grouping similar recurring errors together, and automatically detecting misspelt words, among other tools. Krishna et al. (2018) successfully performed post-OCR correction on scanned text-line images in romanized Sanskrit using an encoder-decoder model equipped with copying mechanism (Gu et al., 2016) by training the model on synthetically generated training data.

In this case study, we fine-tuned the foundational Sanskrit language model Byt5-Sanskrit (Nehrdich et al., 2024) to perform the task of post-correction. ByT5-Sanskrit is pre-trained on a large body of Sanskrit data and achieves state-of-the-art performance when fine-tuned on a number of downstream NLP tasks. It is based on ByT5 (Xue et al., 2022), which is a tokeniser-free language model architecture that operates directly on raw text (bytes or characters), supporting processing of morphologically rich languages such as Sanskrit. As a descendant of mT5 (Xue, 2020) and T5 (Raffel et al., 2020), ByT5 adopts the successful paradigm of pre-training a Transformer (Vaswani, 2017) on a large text corpus, followed by training it to perform a number of downstream tasks, where the transformer learns to take in text as input and generate the appropriate target text depending on the downstream task.

Document AI by Google is a commercially available OCR platform that enables users to digitise printed and handwritten documents via API calls. In this study, we used Document AI as a baseline for comparison since Hegghammer (2022) found that Document AI performs better than competing solutions such as Amazon Textract. Other notable commercially available OCR systems for Devanāgarī script are Ind.Senz (Hellwig, nd), and Surya AI (Paruchuri, nd). Among open-source software, Tesseract (Weil et al., nd), Sanskrit-OCR (Dwivedi et al., 2020), EasyOCR (JaidedAI, nd), and PaddleOCR (Community, nd) are notable mentions. The open-source document analysis and annotation platform eScriptorium (Kiessling et al., 2019; Kiessling et al., 2021; Kiessling, 2020) enables a standardised way to comprehensively digitise text from historical and non-Latin manuscripts using tools such as Kraken (Kiessling, 2019) and Segmento (Janes et al., 2021). The project provides a user-friendly web interface that allows uploading document collections, performing manual or automated layout analysis, and multi-script recognition support, among other features. Other initiatives that provide a full suite of tools for historical document processing are HisDoc 2.0 (Liwicki, 2014), IMPACT (Papadopoulos et al., 2013), Transcibus (Kahle et al., 2017) and Transcriptorium (Sánchez et al., 2013). Platforms that are designed specifically for the Devanāgari Script are Project HInDoLA (Trivedi and Sarvadevabhatla, 2019) and Project Akshar Anveshini[4].

We describe in detail the proposed HTR method in the next section.

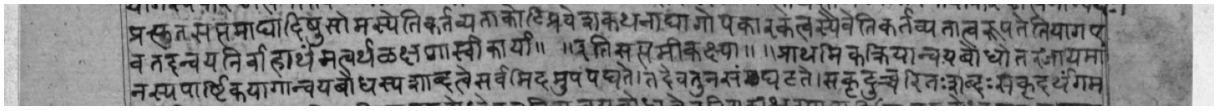
3 Method

After selecting documents that meet the layout-based selection criteria, text lines are segmented from the manuscripts using a line segmentation strategy that is tightly coupled with the selection criteria. The next step entails manual annotation of a small percentage of the segmented line

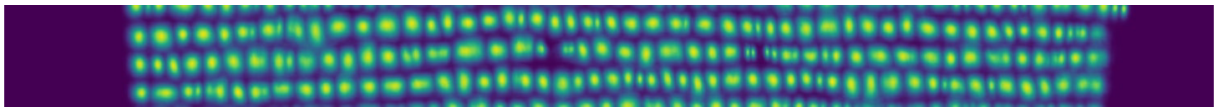
[4] `https://www.cse.iitb.ac.in/~ocr/`

images. These input-label pairs *(segmented line images, ground truth annotations)* are used to fine-tune a pre-trained recognition model for the Devanāgarī script, which recognises the text from the segmented line images. We then fix the mistakes commonly made by the recognition model in the post-correction step. To do this, a pre-trained Sanskrit language model is fine-tuned to learn the task of spelling correction using the input-label pairs *(recognition model predictions, ground truth annotations)*. Finally, the performance of the pipeline is quantified by comparing the digitised text with ground truth annotations from the test dataset. For VK, we annotated 50 pages, of which we used 40 pages as training data, reserved 3 pages as validation data, and held out 7 pages as test data. With MV, we annotated all 16 pages, out of which we used 7 pages as the training data, reserved 3 pages as validation data, and held out 6 pages as the test data. For DMV, we annotated 17 pages, out of which we used 7 pages as training data, reserved 3 pages as validation data, and 7 pages as test data.

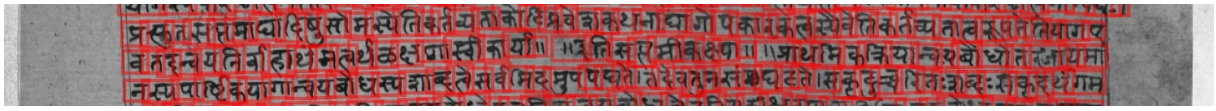
In the following subsections, we provide a detailed explanation of each step, with the corresponding code available on GitHub[5].



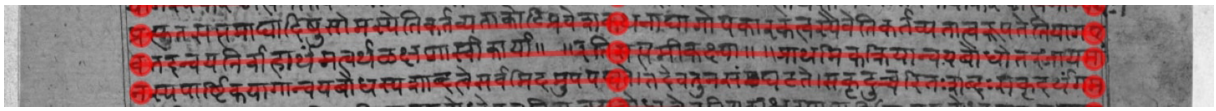
(a). High-resolution greyscale image of manuscript page.



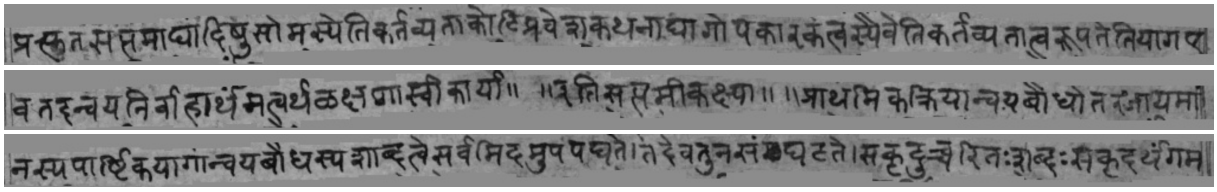
(b). Region scores (heat map) predicted with the CRAFT model.



(c). Using the region scores to detect bounding rectangles.



(d). Using the region scores to fit each text line with two piecewise linear segments.



(e). Final segmented line images.

Figure 1. Stages of line segmentation.

3.1 Line segmentation

Generalised robust text line segmentation remains challenging when dealing with diverse page layouts, as shown by recent studies (Kiessling et al., 2024; Nikolaidou et al., 2022; Janes et al., 2021; Kiessling, 2020; Simistira et al., 2016). To address this challenge, our study focuses specifically on manuscripts that follow a consistent layout template: they contain no pictorial

[5] https://github.com/flame-cai/case-study-handwritten-sanskrit-ocr

elements and are single-columned texts with no line breaks. While this is an important limitation, we find that a vast majority of handwritten Early Modern Sanskrit manuscripts fit this selection criteria. The line segmentation method that we discuss below is tightly linked with this layout-based selection criterion.

To extract text line images from the manuscripts, the page images are first converted to greyscale (Fig. 1a) since the colour of the manuscript has no information about what the characters are. Next, we use the CRAFT model to get character region scores of the manuscript page images (Fig. 1b). The region scores represent the probability that a given pixel is the centre of a character. In other words, the greyscale images are converted to heat maps, where the presence of a character is highlighted. Below, we explain a custom algorithm designed to process these heatmaps for segmenting line images from the manuscripts that meet the selection criteria. We developed this algorithm because we found the default connected component method recommended in the CRAFT paper (Baek et al., 2019) to perform undesirably when applied to the target manuscripts considered in this case study.

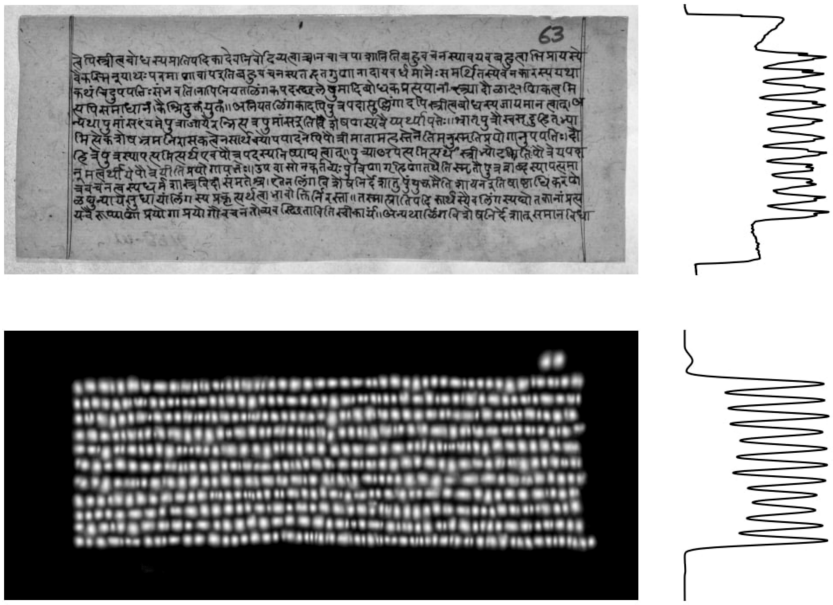


Figure 2. Comparison between horizontal projection profiles of a manuscript page image and the corresponding CRAFT output (heat map). When applying projection profile approaches to CRAFT outputs, we have the following advantages: (a) the local maxima in the projection profiles of the heat map are much more apparent, and (b) we can easily differentiate between characters (text) and non-characters (drawings, smears, lines)

We observe that applying projection profile based approaches on the heat maps is significantly easier compared to the greyscale images, as illustrated in Fig. 2 (Leow et al., 2023; Shtok et al., 2021; Phung et al., 2020). To convert these probability heat maps into discrete character regions, the pixel values are scaled to the range of 0-255 before applying a binarization threshold of `100`. This thresholding creates a binary image where pixels above the value of 100 are considered potential character regions. Then, the border following algorithm by Suzuki and others (1985) (implemented via OpenCV's `findContours`) is used to trace the boundaries of the binarized contour. Next, the OpenCV method `boundingRect` is used to fit bounding rectangles around these contours (Fig. 1c). A bounding rectangle is the smallest rectangle that can completely enclose a contour. In cases where handwritten lines are too close to each other, characters from adjacent lines can get grouped together into a single tall bounding rectangle. To address this, we first identify the y-coordinates of text lines by finding peaks in the horizontal projection profile of the heat map. We then calculate the typical line spacing by measuring the vertical distances between consecutive lines (peaks). The `80th` percentile of these line-to-line distances serves as the reference for typical line spacing in the manuscript. This reference helps in identifying

bounding rectangles that are unusually tall and likely contain characters from multiple lines. A bounding rectangle with height `h` exceeding the reference line spacing is subdivided into multiple rectangles. Specifically, the number of needed subdivisions are calculated and then the rectangle is split into sections of equal height, with any remainder height being added to the final section. Each section maintains the original width and x-coordinate but is assigned an appropriate vertical position and height. It should be noted that the binarization threshold of `100`, and the percentile value of `80` are ad-hoc design choices in this procedure and can be adjusted depending on the manuscript if required.

Next, to correctly handle the curvature of handwritten lines, we fit each handwritten line of the page with two piecewise linear segments (Fig. 1d). A piecewise line is made up of multiple straight line segments, each defined by a pair of coordinates. First, the vertical projection profile of the heat map is computed to identify the approximate x-coordinates of the start, end, and centre of the text lines of a given page. Following this, the horizontal projection profiles are calculated along a 100-pixel width at these x-coordinates. This step gives us the corresponding y-coordinates for the vertices of the piecewise line at the start, end, and centre. As a result, we obtain the x and y coordinates of the three points that define each piecewise line. Finally, we use the proximity of the bounding rectangles to the piecewise lines to assign a line number to each bounding rectangle, and then we extract all bounding rectangles with the same line numbers and paste them all into new images (Fig. 1e). The background colour of the new images is the median colour of the page.

For the given target manuscripts, we found that piecewise lines made up of two line segments were sufficient to handle curvy handwritten lines. However, it should be straightforward to fit piecewise lines, which are made up of more than two line segments if the manuscripts necessitate it.

After segmenting the line images, we manually annotated the text content from the segmented line images. To do this, we used a home-brewed annotation tool designed to reduce effort on the annotator's part—by placing text boxes right below the line images. The tool also supports manual segmentation of lines and marginalia from pages with tricky edge-case layouts. With this, the dataset creation is complete, with the segmented line images as inputs and the ground truth annotations as the labels. This dataset is used to fine-tune a pre-trained Devanāgari recognition model (provided by the EasyOCR Python package).

3.2 Line recognition

The line recognition model takes a segmented line image as its input and learns to predict a sequence of Unicode codepoints. This transformation from a segmented line image to text is done by passing the image through the CNN-BiLSTM-CTC architecture. The CNN-BiLSTM-CTC architecture implemented in the EasyOCR Python package has desirable inductive biases, making it effective in solving such image-to-sequence tasks. Notably, CNNs are translation-equivariant (Cohen and Welling, 2016) and the state of the network in the BiLSTM architecture depends not only on the current input but also on the previous and future states (Graves and Schmidhuber, 2005). LSTMs also handle sequences of varying lengths and can explicitly capture and store information over long sequences (Hochreiter and Schmidhuber, 1997). Such neural network architecture designs, which make use of task-specific prior knowledge in the form of inductive biases, allow learning from data to be more sample-efficient (Batzner et al., 2022). The CTC loss is designed to handle sequences where the input length does not perfectly align with the target length; this is significant for handwritten text where the input text can have varying handwriting sizes. Moreover, CTC is suitable for handwritten characters, where characters are irregularly spaced, as it allows for training without explicit alignment between the input image and output label. As CTC does not explicitly segment its input sequences, it removes the need to locate inherently ambiguous label boundaries, such as those present in handwritten text, allowing label predictions to be grouped together if proven useful (Graves et al., 2006).

The recognition model, being an RCNN, has an architecture made up of ResNet for feature extraction with `512` output channels, followed by a BiLSTM for sequence modelling with a hidden state of size `512`, and CTC for prediction. The recognition model is fine-tuned using the code provided in Clova AI's deep-text-recognition-benchmark repository[6], as suggested by EasyOCR.

Once we segment line images from the page images using the line segmentation method discussed in Section 3.1, the segmented line images for each page and their corresponding labels are converted into the lightning memory-mapped database format, which is used to fine-tune the recognition model. Data filtering is turned off to prevent filtering out of any data samples containing special characters that were not passed to the character argument. This way, even if one character from the line will never be predicted correctly due to its absence in the character set, the model would not miss out on learning other characters, which may be present in the character set.

When fine-tuning a target manuscript, the image height and image width arguments are set to the average height and average width of the segmented line images, respectively; these may vary across the manuscripts. We enable padding and set the seed to the default value of `1111`. The recognition model is fine-tuned for `2000` iterations, batch size is set to `16`, the maximum label length (using the `batch_max_length` argument) is set to `250`. The batch ratio is set to `1`. The model is fine-tuned using Adadelta Optimiser with a learning rate of `1`, ρ (the decay rate) being `0.95`, and ϵ (for maintaining numerical stability) being `1e-8`. Over the `2000` iterations, the model with the lowest normalised edit distance (on the validation dataset) is saved. This recognition model is then used to recognise the characters present in the segmented line images.

3.3 Post-correction

Once the recognition model predicts the text from the line images, we fine-tune the language model ByT5-Sanskrit to do the downstream task of post-correction, i.e., making corrections to the outputs of the recognition model. More precisely, we fine-tune a dedicated post-correction model to learn to correct mistakes made by each dedicated recognition model of each manuscript.

We keep the tokenizer and model configuration as given in the Sanskrit-ByT5 Hugging Face repository unchanged[7]. As ByT5-Sanskrit has been pre-trained on transliterated Sanskrit (IAST), we convert the outputs of the recognition model into IAST, perform the post-correction, and then transliterate the post-corrected text back to Devanāgarī script. The Aksharamukha (Rajan, nd) Python package is used to do the transliteration.

The model is fine-tuned for `1500` steps while checkpointing and validating after every `100` steps, using AdaFactor as the optimisation strategy, with the learning rate parameter set to `None`. The model with the best validation accuracy is saved as the final fine-tuned model. Inference is done using the beam search decoding strategy with the number of beams set to `3`.

It was observed that the fine-tuned Byt5-Sanskrit sometimes hallucinates and gets stuck in a loop while post-correcting some lines. To identify and discard such hallucinations (without using ground truth annotations), we employ a heuristic threshold that acts on the difference between a statistic computed for both the text before and after post-correction. The statistic used is the ratio of the number of characters in a line to the number of unique characters in the same line.

4 Results

We present a quantitative comparison between the proposed HTR method and out-of-the-box OCR solutions, namely Google's DocumentAI and TesseractOCR. For evaluating the performance, we employ two metrics that quantify the dissimilarity between predicted text and ground

[6] `https://github.com/clovaai/deep-text-recognition-benchmark`
[7] `https://huggingface.co/chronbmm/sanskrit-byt5-ocr-postcorrection`

truth text: the Character Error Rate (referred to as CER henceforth) and Grapheme Cluster Error Rate (Williams, 1984) (referred to as GCER henceforth). A grapheme cluster corresponds to a visually identifiable unit in the script. For example, the grapheme cluster (क्ष , "kṣa"), is made up of characters U+0915 (क , "ka"), U+094D (् , halanta) and U+0937 (ष , "ṣa"). The CER would consider this क्ष as 3 units, while the GCER will consider this as 1 unit. Hence, from a data annotation and usability perspective, using the GCER as the evaluation metric ensures a more faithful and stricter evaluation.

We utilised the input-label pairs *(segmented line images, ground truth annotations)* extracted from 7 pages as training data to fine-tune a dedicated recognition model for each manuscript. As can be seen in Table. 1, such manuscript-specific fine-tuned recognition models show considerable improvement in the error rates compared to out-of-the-box OCR solutions for all three manuscripts. The error rates of the recognition models drop as the fine-tuning dataset sizes increase, as seen in Fig. 3. For each manuscript, the GCER of the corresponding fine-tuned recognition model decreases by 40–50% after being fine-tuned on just one page. Furthermore, perhaps more surprisingly, it is observed that fine-tuning on just one line image of the target manuscript reduced the GCER by 37%.

Table 1. Comparison of recognition models (fine-tuned on 7 pages) and out-of-the-box OCR solutions.

Manuscript Name	Proposed Method		Document AI		TesseractOCR	
	CER	GCER	CER	GCER	CER	GCER
VK	0.097	0.193	0.230	0.330	0.803	0.922
MV	0.085	0.155	0.261	0.501	0.507	0.857
DMV	0.095	0.170	0.342	0.481	0.760	0.975

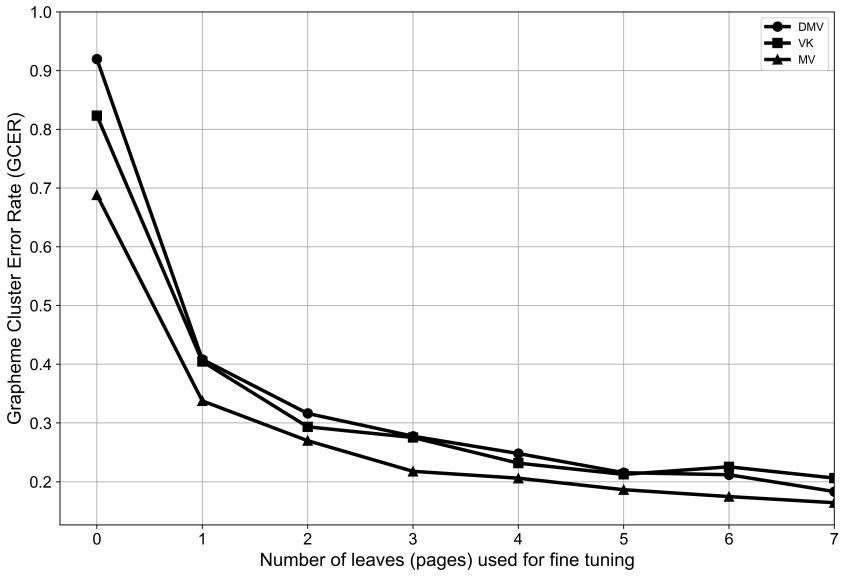


Figure 3. For all three manuscripts, the GCER decreases as the number of pages the recognition model is fine-tuned on increases.

When evaluating the performance of the line segmentation methods, we opt for a crude evaluation metric due to resource constraints, as manual pixel-level ground truth annotation for segmentation requires careful boundary delineation and is labour-intensive (Grüning et al., 2019; Müller et al., 2022). We compare three line segmentation methods, SeamFormer (Vadlamudi et al., 2023), LCG (Nguyen et al., 2022) and the proposed document-specific method, which builds on CRAFT (Baek et al., 2019) as follows: First, lines are extracted from three pages of each manuscript using each of the 3 line segmentation methods to create 3 datasets per manuscript

(each having training, validation, and test splits). A dedicated line recognition model is then fine-tuned on each of the 3 datasets with all the hyperparameters held constant. The accuracies of the downstream recognition models on the respective test sets act as a proxy for the quality of line segmentation done by each method. Using this crude metric, it was observed that the proposed approach performed better than SeamFormer and LCG for the target manuscripts (Table. 2).

Table 2. Comparison of line segmentation methods: LCG, SeamFormer, and the proposed method, using error rates of downstream fine-tuned recognition models as a crude metric.

Manuscript Name	Proposed Method		SeamFormer		LCG	
	CER	GCER	CER	GCER	CER	GCER
VK	0.147	0.275	0.169	0.336	0.433	0.651
MV	0.153	0.259	0.211	0.365	0.201	0.336
DMV	0.116	0.211	0.389	0.596	0.336	0.507

When evaluating post-correction models, it is important to note that in the line recognition step we fine-tuned a *dedicated* recognition model for each of the target manuscripts. Then, we fine-tune a *dedicated* post-correction model to learn to correct mistakes made by a particular dedicated recognition model. This raises the question of budgeting the available annotated data between the recognition model and the post-correction model. For example, as we have `40` pages of ground truth annotated data (for VK), we may use `2` pages to fine-tune the recognition model using the input-label pairs *(segmented line images, ground truth annotations)* and have it predict the text content of the remaining `38` pages. Then, the post-correction model is fine-tuned using these input-label pairs *(recognition model predictions, ground truth annotations)* of the `38` pages. In other words, we may budget the annotated data such that `2` pages are used to fine-tune the recognition model and `38` pages are used to fine-tune the post-correction model. The performance of a range of such dataset budgets is reported below and is visualised in Fig. 4

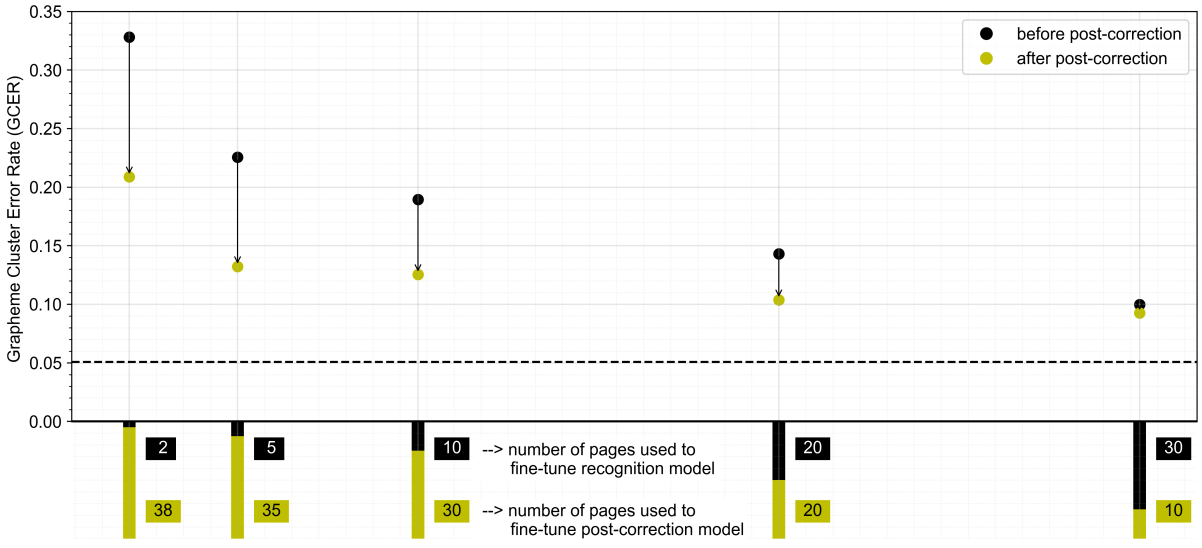


Figure 4. The bars along the x-axis denote various ratios of budgeting of the total available annotated data (40 pages) between fine-tuning the recognition model and fine-tuning the post-correction model. For each budget, we plot two points: one black and one yellow. The black point denotes the GCER of the outputs of the recognition model, while the yellow point denotes the GCER after post-correcting the recognition model outputs. The black dashed line denotes the error when all 40 pages are used to fine-tune a "pure" recognition model (no post-correction).

We observe that fine-tuning the recognition model is more sample-efficient than fine-tuning

the post-correction model. For instance, we see in Fig. 4 that the data budget of `10-30` between the recognition model and post-correction model achieves a better GCER than the data budget of `2-38`. In other words, we observe that the best way to budget the ground truth annotated data is to use *all 40 pages* to fine-tune the "pure" recognition model. The pure recognition model performs the best with a GCER of 5% (and CER of 2.3%) which is shown as the black dotted line in Fig. 4. The best-performing "hybrid" model (recognition model + post-correction model) has a GCER of 9.3% (and CER of 4.5%).

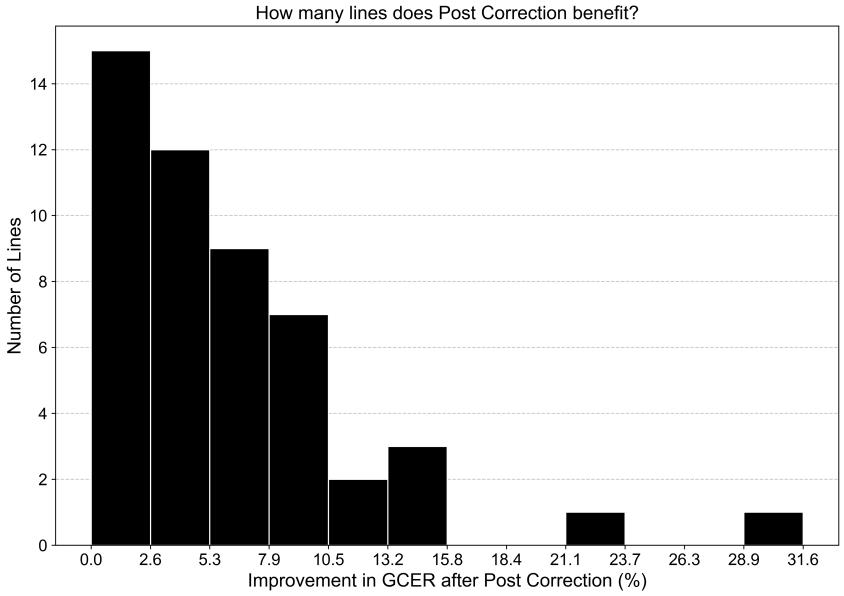


Figure 5. A histogram with the x-axis denoting the difference between GCER before post-correction and GCER after post-correction. The y-axis denotes the number of lines per bucket. For most lines, the post-correction model reduces the GCER. We observe this plot when we use 10 pages to fine-tune the recognition model and 30 pages to fine-tune the post-correction model.

Figure 6. An illustrative example of a line where the post-correction model $(b) \rightarrow (c)$ fixes errors made by the recognition model $(a) \rightarrow (b)$. Notice the extra characters "ol" and "ṃ" in the recognition model outputs are due to diacritics from the line above overlapping with the segmented line (a).

While the pure recognition model performs better than the hybrid models *on average*, the hybrid models often produce more accurate outputs on *individual lines*. A fine-tuned post-correction model improves the GCER of the recognition model outputs for most lines (Fig. 5), and makes language-aware spelling corrections, causing its outputs to be different than those of a pure recognition model. For example, sometimes diacritic marks of characters from adjacent lines can get partially cropped in with the current line image due to the lines being too close to each other. This may trick the recognition model into incorrectly predicting a (पि , "pi") as a (पिं , "piṃ") as illustrated in Fig. 6. We observe that such errors by the recognition model are often corrected by the post-correction model.

5 Discussion

Deep learning models exhibit significant performance degradation when encountering test data from distributions that deviate from their training distribution, even in cases where such distributional shifts appear negligible to human observers (Kiessling et al., 2024; Das, 2021; European Union Aviation Safety Agency et al., 2023; Aubreville et al., 2021; Recht et al., 2019; Hendrycks et al., 2021). In this case study too, we highlight the crucial role fine-tuning plays in helping the recognition model adapt to period-specific writing conventions, as well as to subtle distribution shifts due to variability in page texture, ink characteristics, imaging conditions, camera used, etc. This explains our observation that fine-tuning on line images from just a few pages of a target manuscript significantly improves the recognition model's accuracy, thus proving to be beneficial in this data scarce regime.

Hence when digitising text from manuscripts VK and DMV, we fine-tune a dedicated recognition-model-VK on line images from manuscript VK, and a dedicated recognition-model-DMV on line images from manuscript DMV. We observe that recognition-model-VK, when applied to manuscript DMV, works well but is less accurate than the dedicated recognition-model-DMV, and vice versa - suggesting a distribution shift between line images from manuscript VK and manuscript DMV. Moreover, we observe that the dedicated post-correction-model-VK, which is fine-tuned to make corrections to the predictions of recognition-model-VK, does not significantly reduce the error rate when applied to predicions of recognition-model-DMV and vice versa. Similarly, we observed that the dedicated post-correction model fine-tuned to make corrections to the predictions of a Tesseract-based OCR engine (Maheshwari et al., 2022), does not significantly reduce the error rate when applied to predictions of recognition-model-VK and recognition-model-DMV. This suggests overfitting and less overlap between types of mistakes made by recognition-model-VK, recognition-model-DMV and the Tesseract-based OCR engine used by Maheshwari et al. (2022). Perhaps the post-correction model can generalise better and overfit less if ByT5-Sanskrit is fine-tuned to make corrections to the predictions of *various different* base recognition models (Löfgren and Dannélls, 2024).

The post-correction model, although less sample-efficient, shows promise due to its language-aware correction abilities. Thus, using an ensemble of pure recognition models, hybrid models (recognition + post-correction), and existing tools such as OpenOCRCorrect (Saluja et al., 2017a; Saluja et al., 2017b) may amplify the impact of post-correction by allowing annotators to select the best prediction out of the available options. A more granular form of post-correction (instead of line-level post-correction) may also enhance the benefits further.

The line segmentation method we implement, which builds on CRAFT (Baek et al., 2019), does not require data annotation and fine-tuning. A limitation of the proposed line segmentation method is that it can only be used on target manuscripts that pass the layout-based selection criteria. It also does not currently support digitising footnotes, page numbers, marginal notes, and signatures.

In the next annotation cycle, Sanskrit scholars can thus use this pipeline and make faster corrections to the HTR'ed text rather than annotating entire lines manually (Rijhwani et al., 2023).

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Credit author statement

Kartik Chincholikar: Visualization, Conceptualization, Methodology, Software, Writing - Original draft preparation, Data curation, Formal analysis, Validation, Investigation. **Shagun Dwivedi:** Visualization, Conceptualization, Methodology, Software, Data curation, Writing - Original draft preparation, Formal analysis, Validation, Investigation. **Kaushik Gopalan:**

Conceptualization, Methodology, Software, Supervision, Writing - Reviewing and Editing, Project administration. **Tarinee Awasthi:** Writing - Original draft preparation, Data curation, Resources.

Acknowledgment

The authors wish to express their gratitude to Dr. Oliver Hellwig for his invaluable guidance and support during our discussions. The authors also wish to express their thanks and appreciation to Lalchand Research Library, DAV College, Chandigarh, India, for making old and rare manuscript data available online for educational and research purposes. Finally, the authors thank the anonymous reviewers for their constructive comments, which greatly improved the paper.

References

[Acharya et al.2015] Shailesh Acharya, Ashok Kumar Pant, and Prashnna Kumar Gyawali. 2015. Deep learning based large scale handwritten devanagari character recognition. In *2015 9th International conference on software, knowledge, information management and applications (SKIMA)*, pages 1–6. IEEE.

[Adiga et al.2018] Devaraja Adiga, Rohit Saluja, Vaibhav Agrawal, Ganesh Ramakrishnan, Parag Chaudhuri, K Ramasubramanian, and Malhar Kulkarni. 2018. Improving the learnability of classifiers for Sanskrit OCR corrections. In *The 17th World Sanskrit Conference, Vancouver, Canada. IASS*, pages 143–161.

[Alberti et al.2019] Michele Alberti, Lars Vögtlin, Vinaychandran Pondenkandath, Mathias Seuret, Rolf Ingold, and Marcus Liwicki. 2019. Labeling, cutting, grouping: an efficient text line segmentation method for medieval manuscripts. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1200–1206. IEEE.

[AlKendi et al.2024] Wissam AlKendi, Franck Gechter, Laurent Heyberger, and Christophe Guyeux. 2024. Advancements and challenges in handwritten text recognition: A comprehensive survey. *Journal of Imaging*, 10(1):18.

[Aubreville et al.2021] Marc Aubreville, Christof Bertram, Mitko Veta, Robert Klopfleisch, Nikolas Stathonikos, Katharina Breininger, Natalie ter Hoeve, Francesco Ciompi, and Andreas Maier. 2021. Quantifying the scanner-induced domain gap in mitosis detection. *arXiv preprint arXiv:2103.16515*.

[Baek et al.2019] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9365–9374.

[Bansal and Sinha2001] Veena Bansal and MK Sinha. 2001. A complete ocr for printed hindi text in devanagari script. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 0800–0800. IEEE Computer Society.

[Batzner et al.2022] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. 2022. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453.

[Bhardwaj and others2020] Anuj Bhardwaj et al. 2020. Handwritten devanagari character recognition using deep learning-convolutional neural network (cnn) model. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 17(6):7965–7984.

[Chamchong and Fung2012] Rapeeporn Chamchong and Chun Che Fung. 2012. Text Line Extraction Using Adaptive Partial Projection for Palm Leaf Manuscripts from Thailand. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 588–593, Bari, Italy, September. IEEE.

[Chandna et al.2016] Swati Chandna, Francesca Rindone, Carsten Dachsbacher, and Rainer Stotzka. 2016. Quantitative exploration of large medieval manuscripts data for the codicological research. In *2016 IEEE 6th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 20–28. IEEE.

[Chavan et al.2014] Prasad Chavan, Suyog Sankpal, Akshay Sonawane, and Shahid Shaikh. 2014. Application for handwritten devnagari optical character recognition. *Artificial Intelligent Systems and Machine Learning*, 6(2).

[Christy et al.2017] Matthew Christy, Anshul Gupta, Elizabeth Grumbach, Laura Mandell, Richard Furuta, and Ricardo Gutierrez-Osuna. 2017. Mass digitization of early modern texts with optical character recognition. *Journal on Computing and Cultural Heritage (JOCCH)*, 11(1):1–25.

[Cohen and Welling2016] Taco Cohen and Max Welling. 2016. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.

[Communitynd] PaddlePaddle Community. n.d. Paddleocr. `https://paddlepaddle.github.io/PaddleOCR/main/en/index.html`. Accessed: 23-Nov-2024.

[Cutting and Cutting-Decelle2021] Graham A. Cutting and Anne-Francoise Cutting-Decelle. 2021. Intelligent document processing – methods and tools in the real world.

[Das2021] Deepayan Das. 2021. *Enhancing OCR Performance with Low Supervision.* Ph.D. thesis, International Institute of Information Technology Hyderabad.

[Deore and Pravin2020] Shalaka Prasad Deore and Albert Pravin. 2020. Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset. *Sādhanā*, 45(1):243.

[Dwivedi et al.2020] Agam Dwivedi, Rohit Saluja, and Ravi Kiran Sarvadevabhatla. 2020. An ocr for classical indic documents containing arbitrarily long words. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 560–561.

[European Union Aviation Safety Agency et al.2023] European Union Aviation Safety Agency, Jean Marc Cluzeau, Xavier Henriquel, Georges Rebender, Guillaume Soudain, Daedalean AG, Dr. Luuk van Dijk, Dr. Alexey Gronskiy, David Haber, Dr. Corentin Perret-Gentil, and Ruben Polak. 2023. Concepts of design assurance for neural networks (codann). Accessed: 2023-10-25.

[Graves and Schmidhuber2005] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052. IEEE.

[Graves et al.2006] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

[Graves et al.2007] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2007. Multi-dimensional recurrent neural networks. In *International conference on artificial neural networks*, pages 549–558. Springer.

[Grüning et al.2019] Tobias Grüning, Gundram Leifert, Tobias Strauß, Johannes Michael, and Roger Labahn. 2019. A two-stage method for text line detection in historical documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 22(3):285–302.

[Gu et al.2016] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

[Hegghammer2022] Thomas Hegghammer. 2022. Ocr with tesseract, amazon textract, and google document ai: a benchmarking experiment. *Journal of Computational Social Science*, 5(1):861–882.

[Hellwignd] Oliver Hellwig. n.d. Indsenz. `http://www.indsenz.com/int/index.php`. Accessed: 23-Nov-2024.

[Hendrycks et al.2021] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. 2021. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15262–15271.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[JaidedAInd] JaidedAI. n.d. Easyocr. `https://github.com/JaidedAI/EasyOCR`. Accessed: 23-Nov-2024.

[Janes et al.2021] Juliette Janes, Ariane Pinche, Claire Jahan, and Simon Gabay. 2021. Towards automatic TEI encoding via layout analysis. In *Fantastic future 21, 3rd International Conference on Artificial Intelligence for Librairies, Archives and Museums*, Paris, France, December. AI for Libraries, Archives, and Museums (ai4lam).

[Jindal and Ghosh2023] Amar Jindal and Rajib Ghosh. 2023. Text line segmentation in indian ancient handwritten documents using faster r-cnn. *Multimedia Tools and Applications*, 82(7):10703–10722.

[Kahle et al.2017] Philip Kahle, Sebastian Colutto, Günter Hackl, and Günter Mühlberger. 2017. Transkribus-a service platform for transcription, recognition and retrieval of historical documents. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 4, pages 19–24. IEEE.

[Kant and Vyavahare2016] Mr Akshay J Kant and AJ Vyavahare. 2016. Devanagari ocr using projection profile segmentation method. *International Research Journal of Engineering and Technology (IRJET)*, 3(7):132–134.

[Karayil et al.2015] Tushar Karayil, Adnan Ul-Hasan, and Thomas M Breuel. 2015. A segmentation-free approach for printed devanagari script recognition. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 946–950. IEEE.

[Kiessling et al.2019] Benjamin Kiessling, Robin Tissot, Peter Stokes, and Daniel Stokl Ben Ezra. 2019. eScriptorium: An Open Source Platform for Historical Document Analysis. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, pages 19–19, Sydney, Australia, September. IEEE.

[Kiessling et al.2021] Benjamin Kiessling, Gennady Kurin, Matthew Miller, and Kader Smail. 2021. Advances and limitations in open source arabic-script ocr: A case study. *Digital Studies / Le champ numérique*, 11, 12.

[Kiessling et al.2024] Benjamin Kiessling, Gennady Kurin, Matthew Thomas Miller, and Kader Smail. 2024. Advances and Limitations in Open Source Arabic-Script OCR: A Case Study, February. arXiv:2402.10943.

[Kiessling2019] Benjamin Kiessling. 2019. Kraken-an universal text recognizer for the humanities. In *ADHO, Éd., Actes de Digital Humanities Conference*.

[Kiessling2020] Benjamin Kiessling. 2020. A Modular Region and Text Line Layout Analysis System. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 313–318, Dortmund, Germany, September. IEEE.

[Krishna et al.2018] Amrith Krishna, Bodhisattwa P. Majumder, Rajesh Bhat, and Pawan Goyal. 2018. Upcycle your OCR: Reusing OCRs for post-OCR text correction in Romanised Sanskrit. In Anna Korhonen and Ivan Titov, editors, *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 345–355, Brussels, Belgium, October. Association for Computational Linguistics.

[Krishnan and Jawahar2019] Praveen Krishnan and CV Jawahar. 2019. Hwnet v2: an efficient word image representation for handwritten documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 22(4):387–405.

[Leow et al.2023] Chee Siang Leow, Hideaki Yajima, Tomoki Kitagawa, and Hiromitsu Nishizaki. 2023. Single-Line Text Detection in Multi-Line Text with Narrow Spacing for Line-Based Character Recognition. *IEICE TRANSACTIONS on Information and Systems*, E106-D(12):2097–2106, December. Publisher: The Institute of Electronics, Information and Communication Engineers.

[Liwicki2014] Marcus Liwicki. 2014. 3.8 divadia & hisdoc 2.0 approaches at the university of fribourg to digital paleography. *Digital Palaeography: New Machines and Old Texts*, page 123.

[Löfgren and Dannélls2024] Viktoria Löfgren and Dana Dannélls. 2024. Post-ocr correction of digitized swedish newspapers with byt5. In *Proceedings of the 8th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH-CLfL 2024)*, pages 237–242.

[Maheshwari et al.2022] Ayush Maheshwari, Nikhil Singh, Amrith Krishna, and Ganesh Ramakrishnan. 2022. A benchmark and dataset for post-ocr text correction in sanskrit. *arXiv preprint arXiv:2211.07980*.

[Mohite and Shelke2018] Aarati Mohite and Sushama Shelke. 2018. Handwritten devanagari character recognition using convolutional neural network. In *2018 4th International Conference for Convergence in Technology (I2CT)*, pages 1–4. IEEE.

[Moudgil et al.2023] Aditi Moudgil, Saravjeet Singh, Vinay Gautam, Shalli Rani, and Syed Hassan Shah. 2023. Handwritten devanagari manuscript characters recognition using capsnet. *International Journal of Cognitive Computing in Engineering*, 4:47–54.

[Müller et al.2022] Dominik Müller, Iñaki Soto-Rey, and Frank Kramer. 2022. Towards a guideline for evaluation metrics in medical image segmentation. *BMC Research Notes*, 15(1):210.

[Nehrdich et al.2024] Sebastian Nehrdich, Oliver Hellwig, and Kurt Keutzer. 2024. One model is all you need: Byt5-sanskrit, a unified model for sanskrit nlp tasks. *arXiv preprint arXiv:2409.13920*.

[Nguyen et al.2022] Tien-Nam Nguyen, Jean-Christophe Burie, Thi-Lan Le, and Anne-Valérie Schweyer. 2022. An effective method for text line segmentation in historical document images. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1593–1599. IEEE.

[Nikolaidou et al.2022] Konstantina Nikolaidou, Mathias Seuret, Hamam Mokayed, and Marcus Liwicki. 2022. A survey of historical document image datasets. *IJDAR*, 25(4):305–338, December.

[Pal and Chaudhuri2004] Umapada Pal and BB Chaudhuri. 2004. Indian script character recognition: a survey. *pattern Recognition*, 37(9):1887–1899.

[Papadopoulos et al.2013] Christos Papadopoulos, Stefan Pletschacher, Christian Clausner, and Apostolos Antonacopoulos. 2013. The impact dataset of historical document images. In *Proceedings of the 2Nd international workshop on historical document imaging and processing*, pages 123–130.

[Paruchurind] Vikas Paruchuri. n.d. Surya: A Sanskrit OCR tool. `https://github.com/VikParuchuri/surya`. Accessed: 23-Nov-2024.

[Philips and Tabrizi2020] James Philips and Nasseh Tabrizi. 2020. Historical document processing: A survey of techniques, tools, and trends. *KDIR*, pages 341–349.

[Phung et al.2020] Tuan Minh Phung, Minh Ngoc Dinh, Duy Pham Thien Dang, Hoang Minh Tu Van, and Louise Thwaites. 2020. A machine learning-based approach to vietnamese handwritten medical record recognition. In *ACIS*.

[Raffel et al.2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

[Rajannd] Vinodh Rajan. n.d. Aksharamukha - Indic script converter and transliterator. `https://github.com/virtualvinodh/aksharamukha-python`. Python library for Sanskrit and Indic script conversion. Accessed: 23-Nov-2024.

[Recht et al.2019] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR.

[Rijhwani et al.2023] Shruti Rijhwani, Daisy Rosenblum, Michayla King, Antonios Anastasopoulos, and Graham Neubig. 2023. User-centric evaluation of ocr systems for kwak'wala. *arXiv preprint arXiv:2302.13410*.

[Ronneberger et al.2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer.

[Saluja et al.2017a] Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan, and Mark Carman. 2017a. Error Detection and Corrections in Indic OCR using LSTMs. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 17–22. IEEE.

[Saluja et al.2017b] Rohit Saluja, Devaraj Adiga, Ganesh Ramakrishnan, Parag Chaudhuri, and Mark Carman. 2017b. A framework for document specific error detection and corrections in indic ocr. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 4, pages 25–30. IEEE.

[Sánchez et al.2013] Joan Andreu Sánchez, Günter Mühlberger, Basilis Gatos, Philip Schofield, Katrien Depuydt, Richard M Davis, Enrique Vidal, and Jesse De Does. 2013. transcriptorium: a european project on handwritten text recognition. In *Proceedings of the 2013 ACM symposium on Document engineering*, pages 227–228.

[Sankaran et al.2013] Naveen Sankaran, Aman Neelappa, and CV Jawahar. 2013. Devanagari text recognition: A transcription based formulation. In *2013 12th International Conference on Document Analysis and Recognition*, pages 678–682. IEEE.

[Sarkar et al.2015] Ram Sarkar, Bibhash Sen, Nibaran Das, and Subhadip Basu. 2015. Handwritten devanagari script segmentation: A non-linear fuzzy approach. *arXiv preprint arXiv:1501.05472*.

[Schantz1982] Herbert F Schantz. 1982. *History of OCR, optical character recognition*. Recognition Technologies Users Association.

[Sethi and Chatterjee1977] Ishwar K Sethi and B Chatterjee. 1977. Machine recognition of constrained hand printed devanagari. *Pattern recognition*, 9(2):69–75.

[Sharan et al.2021] SP Sharan, Sowmya Aitha, Amandeep Kumar, Abhishek Trivedi, Aaron Augustine, and Ravi Kiran Sarvadevabhatla. 2021. Palmira: a deep deformable network for instance segmentation of dense and uneven layouts in handwritten manuscripts. In *International Conference on Document Analysis and Recognition*, pages 477–491. Springer.

[Shi et al.2016] Baoguang Shi, Xiang Bai, and Cong Yao. 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304.

[Shtok et al.2021] Joseph Shtok, Sivan Harary, Ophir Azulai, Adi Raz Goldfarb, Assaf Arbelle, and Leonid Karlinsky. 2021. CHARTER: heatmap-based multi-type chart data extraction, November. arXiv:2111.14103.

[Simistira et al.2016] Foteini Simistira, Mathias Seuret, Nicole Eichenberger, Angelika Garz, Marcus Liwicki, and Rolf Ingold. 2016. Diva-hisdb: A precisely annotated large dataset of challenging medieval manuscripts. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 471–476. IEEE.

[Suzuki and others1985] Satoshi Suzuki et al. 1985. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46.

[Svāmiśāstrin and Bhāskararāya1721] Svāmiśāstrin and Bhāskararāya. 1721. Vādakautūhala. Lalchand Research Library, DAV College, Chandigarh, India.
Accessed online at: `https://dav.splrarebooks.com/collection/view/vaadkautuhalam`.
Catalogue entry: Acc No 3976, `https://archive.org/details/LalChandResearchIndologicalResearchCenterManuscriptCatalogDAVCollegeChandigarh/page/n229/mode/2up`.

[Tabrizi2008] MHN Tabrizi. 2008. Digital archiving and data mining of historic document. In *2008 International Conference on Advanced Computer Theory and Engineering*, pages 19–23. IEEE.

[Trivedi and Sarvadevabhatla2019] Abhishek Trivedi and Ravi Kiran Sarvadevabhatla. 2019. Hindola: a unified cloud-based platform for annotation, visualization and machine learning-based layout analysis of historical manuscripts. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 31–35. IEEE.

[Unknownnda] Unknown. n.d.a. Dvādaśa Mahāvākya Vicāraḥ. Lalchand Research Library, DAV College, Chandigarh, India.
Accessed online at: `https://dav.splrarebooks.com/collection/view/dvad-as-hah-mahavakya-vicharah`.
Catalogue entry: Acc No 2148, `https://archive.org/details/LalChandResearchIndologicalResearchCenterManuscriptCatalogDAVCollegeChandigarh/page/n367/mode/2up?view=theater`.

[Unknownndb] Unknown. n.d.b. Mahāvākyārthāḥ. Lalchand Research Library, DAV College, Chandigarh, India.
Accessed online at: `https://dav.splrarebooks.com/collection/view/mahavakyarthah`.
Catalogue entry: Acc No 3243, `https://archive.org/details/LalChandResearchIndologicalResearchCenterManuscriptCatalogDAVCollegeChandigarh/page/n371/mode/2up?view=theater`.

[Uskokov2018] Aleksandar Uskokov. 2018. *Deciphering the hidden meaning: scripture and the hermeneutics of liberation in early Advaita Vedānta*. The University of Chicago.

[Vadlamudi et al.2023] Niharika Vadlamudi, Rahul Krishna, and Ravi Kiran Sarvadevabhatla. 2023. Seamformer: High precision text line segmentation for handwritten documents. In *International Conference on Document Analysis and Recognition*, pages 313–331. Springer.

[Vaswani2017] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

[Vinitha and Jawahar2016] VS Vinitha and CV Jawahar. 2016. Error detection in indic ocrs. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 180–185. IEEE.

[Weil et al.nd] Stefan Weil, Ray Smith, and Zdenko Podobny. n.d. Tesseract. `https://github.com/tesseract-ocr/tesseract`. Accessed: 23-Nov-2024.

[Williams1984] Joanna P Williams. 1984. Phonemic analysis and how it relates to reading. *Journal of Learning Disabilities*, 17(4):240–245.

[Xue et al.2022] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

[Xue2020] L Xue. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.