VOLTAGE: A Versatile Contrastive Learning based OCR Methodology for ultra low-resource scripts through Auto Glyph Feature Extraction

Prawaal Sharma
Infosys
Pune, Maharashtra, India
prawaal_sharma@infosys.com

Poonam Goyal
BITS Pilani
Pilani, Rajasthan, India
poonam@pilani.bits-pilani.ac.in

Vidisha Sharma
Pune, Maharashtra, India
vidishasharma@gmail.com

Navneet Goyal
BITS Pilani
Pilani, Rajasthan, India
goel@pilani.bits-pilani.ac.in

Abstract

UNESCO has classified 2500 out of 7000 languages spoken worldwide as endangered. Attrition of a language leads to loss of traditional wisdom, folk literature, and the essence of the community that uses it. It is therefore imperative to bring digital inclusion to these languages and avoid its extinction. Low resource languages are at a greater risk of extinction. Lack of unsupervised Optical Character Recognition(OCR) methodologies for low resource languages is one of the reasons impeding their digital inclusion. We propose VOLTAGE - a contrastive learning based OCR methodology, leveraging auto-glyph feature recommendation for cluster-based labelling. We augment the labelled data for diversity and volume using image transformations and Generative Adversarial Networks. Voltage has been designed using Takri - a family of scripts used in 16th to 20th century in the Himalayan regions of India. We present results for Takri along with other Indic scripts (both low and high resource) to substantiate the universal behavior of the methodology. An accuracy of 95% for machine printed and 87% for handwritten samples on Takri script has been achieved. We conduct baseline and ablation studies along with building downstream use cases for Takri, demonstrating the usefulness of our work.

1 Introduction

The UNESCO "Atlas of the World's Languages in Danger" (UNESCO, 2021) is considered as a benchmark for the comprehensive list of the world's endangered languages. This study unveils more than 2500 languages and dialects as endangered out of which 200 come from Indian demography.

Optical character recognition (OCR) is used for digitizing historical archives, helping language conservation. There are plenty commercial and open-source OCR engines available for contemporary documents. However, very Low Resource Scripts (LRS) differ in their requirements mainly because of non-availability of large volume of data and limited users. The two most popular unsupervised (or semi-supervised) OCR methods available include Ocular (Berg-Kirkpatrick et al., 2013) and anyOCR (Bukhari et al., 2017). Both methods are designed for large datasets and hence cannot be applied to LRS effectively.

Another alternative is to apply pretrained models for a high resource language as a foundation model and apply few shot learning to customize it and get the desired results. Our experiments conducted for "Takri" using this approach do not result in good accuracy. We have discussed this in detail in results section.

We develop an automated versatile unsupervised OCR methodology (VOLTAGE) for very low resource scripts to address the gap. We use Takri as an example to develop our methodology due to (a) No available labelled data and scanty user base (b) Extremely low digital unlabelled resources. We further evaluate the proposed methodology on four other languages to validate the universal behavior.

As illustrated in Figure 1, VOLTAGE comprises of four steps, (a) Extraction: segmentation of available data into pages, lines, words, characters and symbols; (b) Annotation: feature extraction and recommendation followed by cluster based labelling; (c) Re-enforcement: augmentation of dataset using image transformation and generative AI (GANs); (d) Identification: contrastive learning based classification for character identification. The novelty of the methodology is that the manual intervention including human oracles is bare-minimum. The proposed glyph pattern-based feature recommender system can be applied to any

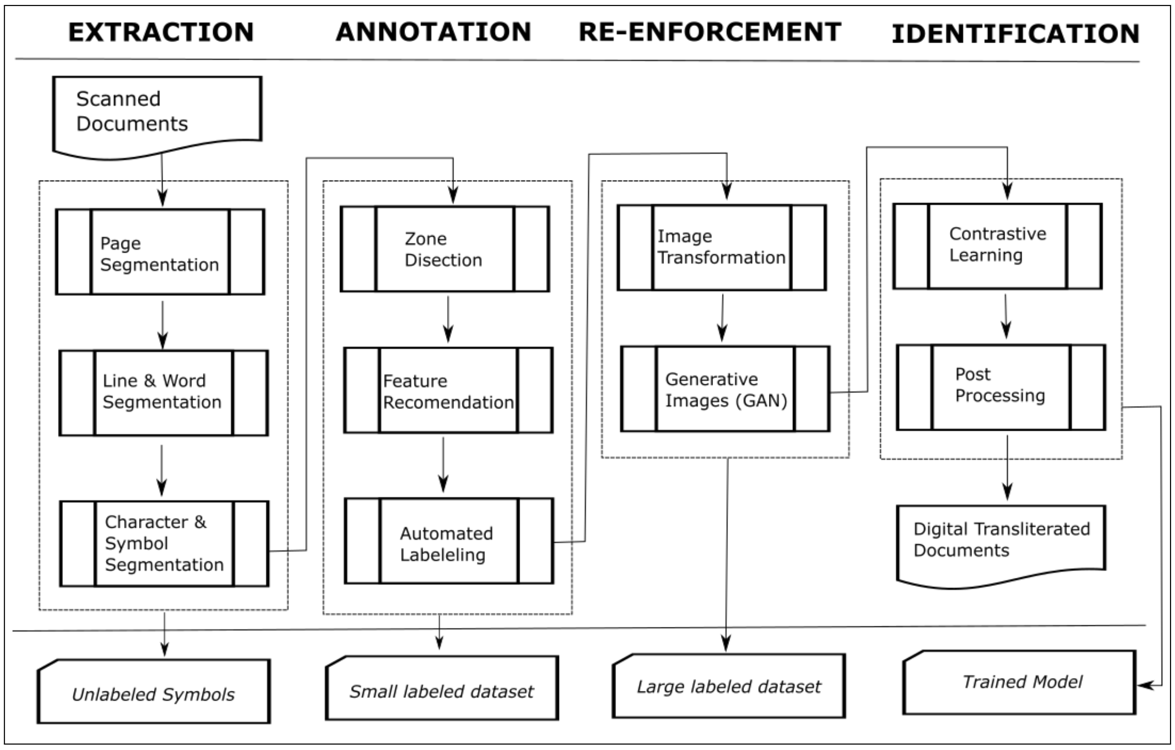


Figure 1: High level design for **V**ersatile unsupervised **OCR** methodology for **L**ow-resource scripts **T**hrough **A**uto **G**lyph feature **E**xtraction (VOLTAGE)

script to recommend apropos feature set.

We empirically discuss our results in detail for Takri and also evaluate on other Indic scripts to validate its generalization capabilities. We also conduct baseline and ablation studies to substantiate our results. The contribution summary of our work is three fold:

- We build versatile and automated OCR methodology using contrastive learning approach for ultra low resource scripts.
- We build a novel glyph feature recomendor system for unsupervised labelling of symbols which can be applied universally.
- We build the largest labelled Takri dataset containing approximately 226,000 symbols, along with downstream use cases on transliteration and synthetic symbol generation models for public use.

2 Related Work

2.1 Optical Character Recognition (OCR)

The early ideas of OCR dates back to 1870's (Chaudhuri et al., 2017). Since then, the OCR systems have evolved, and in the modern world there are many open source OCR systems like Tessaract (Smith, 2007), OCRopus (Breuel, 2008), Kraken (Kiessling et al., 2017) and Calamari (Wick et al., 2018) etc. Although research on OCR for Indic scripts started only in the mid 1970s, however the scope of research was restricted to Devanagari, Tamil and Telugu scripts only (Govindan and Shivaprasad, 1990). Even today, the major work in Indic OCR is limited to the ten scripts namely, Bangla, Devanagari, Gurumukhi, Gujarati, Kannada, Malayalam, Oriya, Tamil, Telugu, and Urdu (Chaudhuri, 2009). Most of the current OCRs are based on deep neural networks which tends to be hungry on data and computational power.

OCR pipeline generally goes through multiple individual tasks including (a) Image acquisition (extracting images containing text from multiple sources for offline images, and capturing live images for online extraction) (b) Pre-processing (application of image processing techniques, to increase raw image quality) (b) Binarization (for scenarios where text and images/videos are mixed, we need to isolate text images from background) (c) Layout Analysis (dividing the images into regions) (d) Segmentation (segmentation of image into pages, lines, words, characters and symbols) (e) Feature Analysis (identification and extraction of key features) (f) Classification (Recognition of symbol with scrip character-set) (g) Post processing (use of pre-compiled vocabulary and language rules to auto correct the unrecognized words) (Tomaschek, 2018).

Supervised OCR: uses labelled dataset for training the classifier. Supervised methods give better performance however, annotation of character level images needs a lot of efforts and is not practical for low resource languages (LRL) where availability of annotators is scarce. Most supervised SOTA OCR systems like Tessarct and OCRopus are pre-trained on a very large image data sets based on deep CNN neural networks (Zeiler and Fergus, 2014).

Unsupervised OCR: Unsupervised transformers like BERT (Devlin et al., 2018), GPT (Radford et al., 2019) etc. have become very successful for diverse NLP tasks. In case of OCR systems only a few unsupervised (or semi-supervised) methods are available like Ocular (Berg-Kirkpatrick et al., 2013) and anyOCR (Bukhari et al., 2017). Ocular uses generative modelling approach incorporating font typesetting, inking and noise. AnyOCR on the other hand is semi-supervised and language agnostic which consumes historical documents and clusters them for training purpose.

From the best of our knowledge, there is no general purpose OCR methodology suitable for ultra low resource scripts used by very limited user groups, with limited or no digital data available. VOLTAGE fills that gap, with the design of autonomous OCR pipeline enabling digitization of ultra low resource scripts.

2.2 Data Augmentation

Data Augmentation (DA) helps with increase of volume and diversity of data. With the advent of deep learning methods where the efficiency and accuracy of models is proportional to the training data, it has become imperative to use data augmentation approaches to generate large volumes of synthetic data and improve the performance of the model (Saini et al., 2022).

Data augmentation for images is classified into two categories (a) Extractive, which augments data applying rules and transformations in form of rotation, brightness, sheer, zoom, flips etc. (Spruck et al., 2021; Kumar et al., 2022) and (b) Generative, which synthesises data based on existing patterns using Generative Adversarial Networks (GAN) (Aggarwal et al., 2021). Generative methods helps in expanding the diversity of textual images along with inclusion of noise, and is therefore is very close to human generated samples (Kukreja et al., 2020; Abedin et al., 2022; Wu et al., 2021).

2.3 Contrastive Learning

The use of contrastive learning in various NLP and computer vision tasks is becoming very popular in recent years (Zhang et al., 2022), including sentence embeddings (Gao et al., 2021), language translation (Pan et al., 2021), text generation (Shu et al., 2021) etc. Contrastive learning can be applied in a self supervised mode, where the anchor and the positive sample are pulled together in embedding space, the negative samples are pushed apart (Chen et al., 2020; Tian et al., 2020). Another approach of using contrastive learning is in supervised mode where multiple positives per anchor are pulled closer together along with many negatives anchors which are pulled further (Khosla et al., 2020). The contrastive losses in this case is the generalization of triplet (Weinberger and Saul, 2009) and N-Pair (Sohn, 2016) losses. In our work, we use supervised contrastive learning to build an character recognition model for ultra low scripts.

3 Scripts and Datasets

3.1 The choice of script

Our methodology is designed for scripts with very low digital resources, hence the choice of script to validate our methodology is an important decision. Takri script, has extremely low available digital resources, no labelled dataset along with low user base. George Grierson, in his Linguistic Survey of India, describes Takri and its variations as a script with shared inherent characteristics consequently classifying it as a "class of scripts" rather than a single script (Grierson, 1909).

To further validate the claim of having common linguistic characteristics within the dialects using Takri as a script, we use a set of 25 sentences used in day to day conversation, and translate them to seventeen dialects (used in the Himalayan regions of Himachal Pradesh, India) by the use of human annotators who are fluent in these dialects. We empirically study various semantic, lexical and syntactic features for these dialects and explore interdependence among these dialects using agglomerative hierarchical clustering (Roux, 2018). Appendix C illustrates the relationship between various dialects used in the Himalayan regions and how the use of one script binds all of them together.

Takri, like most Indic script falls under *Abugidas* class of writing systems (Daniels, 2017), and some of the salient characteristics are summarised below:

- The character set of Takri comprises of 11 vowels, 33 consonants and 10 numbers.
- There are 10 vowel modifiers which can occur on the top, below, left or right of the consonants.
- Takri script does not contain headline unlike other Indic scripts like Devanagari
- Half forms are not used in most versions of Takri.
- Ligatures are also infrequently written.
- Most characters consists of connected components only.
- Compound characters are not present in Takri.

3.2 Datasets

To the best of our knowledge, there is only single source of a good quality dataset sourced from machine printed Takri books collected manually consisting of 272 text blocks containing 2,584 lines and 10,880 words with the resolution of 200dpi (Magotra et al., 2019). We use this dataset as the base and add more samples to increase volume and diversity of samples using data augmentation techniques.

For Gujarati, we use the machine printed limited dataset by Goswami et al. consisting of 7,221 symbols (Goswami and Mitra, 2014). For Modi, we use available dataset by Chandankhede et al (Chandankhede and Sachdeo, 2023a). For Ol chiki and Wancho, there is no dataset available so we use the available printed books and build our own dataset.

4 VOLTAGE: The proposed methodology

Versatile contrastive learning based **OCR** methodology for ultra **L**ow-resource scripts **T**hrough **A**uto **G**lyph feature **E**xtraction (VOLTAGE) follows the pipeline of tasks including, pre-processing and segmentation, automated feature engineering and unsupervised labelling, data augmentation and classification, post processing and evaluation (see Figure 1). We validate our results for Takri on end to end errors and character/word error rates. We further validate VOLTAGE for Modi, Ol Chiki, Gujarati and Wancho to establish the universal effectiveness of our work. We use Python 3.8.12 with conda, opencv for image processing along with ml libraries (keras, numpy, transformers) for our experiment.

4.1 Extraction

It is imperative to extract and segment the input source into lines, words, characters and symbols before it can be put to use for downstream OCR tasks. Segmentation of page into lines and lines into words leverages computation of horizontal and vertical projections (HX and VY as illustrated in Eq. 1 and 2) and find valleys within the threshold (Likforman-Sulem et al., 2007; Shinde and Chougule, 2012; Magotra et al., 2021).

$$VY_i = \sum_{x=0}^{x=Width} (\text{No. of black pixels for } x_i) \quad (1)$$

$$HX_i = \sum_{y=0}^{y=Height} (\text{No. of black pixels for } y_i) \quad (2)$$

Segmentation of words into characters is slightly more complex due to close vicinity of characters and overlaps. To solve this issue, we have enhanced Eq. 2 and compute the enhanced horizontal projection (EHX) which applies additional penalty in downward direction for character segmentation (Eq. 3) because most overlaps in Takri occur in the upper parts. We have observed that this technique, helps in overall reduction of segmentation errors by 3%.

We have observed that abugidas class of scripts overlap their symbols (like Takri) and alphabetic scripts are isolated. Hence when we conduct our experiment for other scripts, we use EHX for Modi/Gujarati and HX for Ol chiki/Wancho.

$$EHX_i = \sum_{y=0}^{y=Height} (\text{No. of black pixels for } y_i) + (\text{Penalty Wt. } * y_i) \quad (3)$$

Furthermore, we further break the individual characters into sub-characters (also called as symbols) by dividing the space into three zones. We design a three step procedure to achieve this. (i) The first step is *Skeletonization* which reduces the thickness of the character into single pixel, and helps to bring uniformity in the thickness irrespective of the input variation (Saha et al., 2016); (ii) Once the characters obtain uniform thickness, we apply *Connected Component Labelling* to label disjoint components (He et al., 2017). Most symbols in Takri are connected (apart from few exceptions

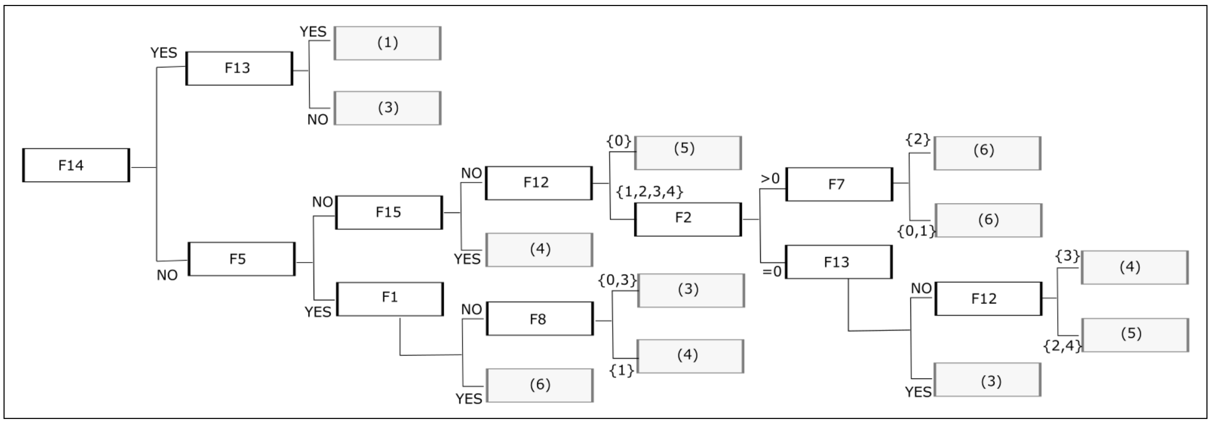


Figure 2: Partition of characters into small groups based on glyph features as recommended by our design, where F# are feature identifiers in the glyph feature inventory

like 𑚌 and hence this step helps in marking disconnected sections within the character. (iii) In the last step, we apply rule based method (to consider exceptions) and perform *Zone Classification* and classify symbols in three different zones. This step is not needed for alphabetic scripts like English, and for those scripts characters and symbols are analogous to each other. The entire process of extraction is illustrated further in Appendix F.

4.2 Annotation

Takri consists of 50 symbols in middle zone, 6 in upper zone and 4 in bottom zone respectively. Annotation of extracted 14,000 symbols into appropriate category is a very critical activity for effective OCR design. Labelling each symbol manually may be the most accurate method but not scalable. Moreover, it becomes further tedious since there are only a handful of people who can read Takri.

We perform unsupervised clustering of symbol images individually for each zone to label the dataset. Unsupervised clustering of images partitions the dataset into visually similar clusters without any access to ground truth labels. We use pretrained models on ImageNet and perform partition into individual characters [1] (Van Gansbeke et al., 2020). We cluster and label images effectively with 96% accuracy for upper and bottom zone characters (see Table 1). However the accuracy for middle zone characters was 69% due to large spread of labels. It has been observed, that the errors have a linear dependency on the number of clusters (Fränti and Sieranoja, 2019). This is also evident from our experiment, and hence it is advisable to divide middle-zone into smaller groups to overcome this.

[1]https://paperswithcode.com/task/image-clustering

Glyph Feature Recommendation System (GFRS): We design a novel recommender system "GFRS", which analyses and recommends the most appropriate glyph features for a given script from our inventory of glyph features without any human intervention. It takes a set of characters used in a language as input and recommends a tree structure with the recommended glyph features, and distribute the characters into smaller groups for more effective annotation. We have validated this for Takri along with 4 other Indic scripts. As illustrated in Figure 2, when GFRS is applied on Takri it recommends 9 features (F1: Presence of headline; F2: Number of loops; F5: Presence of right sidebar; F7: Number of endpoints; F8: Number of junctions; F12: Aspect ratio; F13: Horizontal symmetry; F14: Vertical symmetry; and F15: Number of dots) from the feature store and each identified subgroup does not contain more than 6 symbols. Our process of building the feature store is iterative after analysing the shape characteristics of multiple Indic scripts. We have observed that using the approach, the unsupervised labelling accuracy improves to 96% for middle zone (which was 69% earlier).

Table 1: Unsupervised clustering accuracy for various zones for various k-means combinations.

	Upper Zone	**Middle Zone**	**Bottom Zone**
Distribution	23%	70%	7%
No. of Labels	5	50	4
Accuracy			
50 iterations	91%	61%	93%
300 iterations	**96%**	69%	**97%**
With feature recommendor	-	**96%**	-

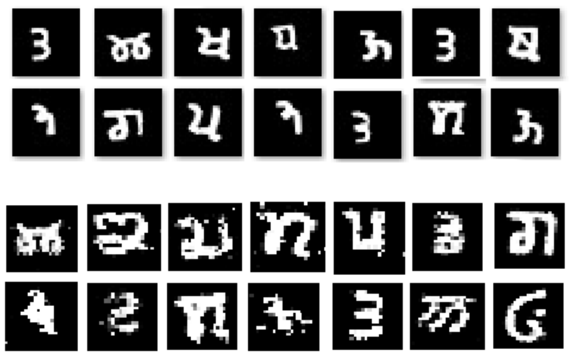


Figure 3: Samples from augmented characters (row 1 and 2) using image transformations (row 3 and 4) using GAN

Appendix A illustrates the entire list of glyph features (32 in total count), which forms the inventory of shape feature set (analysing the distinctive characteristics of these glyphs like number of loops, lines, endpoints, junctions, symmetry etc.). GRFS recommends the most appropriate feature set for a particular script which helps in appropriate distribution of characters and facilitate automatic labelling. We also illustrate as part of Appendix A, the various recommended feature sets for other scripts used in our paper including Modi, Ol Chiki, Gujarati and Wancho.

4.3 Re-enforcement

Our labelled dataset contain approximately 14,000 symbols. We augment this dataset applying transformations for rotations, sheer and brightness. We limit the angular transformation to 9^o, and 10% range on sheer and brightness. Further, we use the transformed images and build a GAN for each character, by the use of four layer generator and discriminator networks, with a learning rate of 2 x 10^{-4} and train on 400 epochs. Figure 3 illustrates the examples from this. The final dataset contains 225,000 symbols for Takri.

4.4 Identification

The earlier forms of OCR designs, either use CNN (Zeiler and Fergus, 2014) or a combination of CNN and LSTM (Staudemeyer and Morris, 2019) as a deep learning method for character identification. We use supervised contrastive learning for our classifier, leveraging multiple positive and negative samples. We discuss the benefits of using this approach empirically later in this paper. Table 2 illustrates multiple error metrics in our work.

Appendix E illustrates details on the architecture of contrasting learning used in our work. We illustrate how transformations (including image processing and GAN) are put to use in an encoder model which maps them to a latent representation space, encapsulating features and similarities. We apply supervised contrastive loss function from SupCon, to maximise the agreement between positive pairs (same character images) and minimize the agreement between negative pairs (different character images) (Khosla et al., 2020).

$$L = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} log \frac{exp(z_i.z_p/\tau)}{\sum_{a \in A(i)} exp(z_i.z_a/\tau)}$$

Here $z_l = Proj(Enc(\tilde{x}_l)) \in R^{Dp}$, the $\cdot$ symbol denotes inner dot product, $\tau \in R^+$ is scaler temperature parameter. Index i is called anchor, index $j(i)$ is called positive and other indices are called negative. $P(i) \equiv p \in A(i) : \tilde{y_p} = \tilde{y_i}$ is the set of indices of all positive in multiviewed batch (2N augmented samples) distinct from i, and $|P(i)|$ is its cardinality.

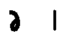
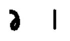
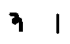
4.5 Post Processing

Table 3 illustrates some examples where the characters are misinterpreted due to over segmentation of characters or incorrect classification due to similarity in visual characteristics. It is therefore pivotal to have some post processing and correct these errors based on language grammar and patterns. Appendix B illustrates an inventory of principles and guidelines for Indic languages. These principles

Table 2: High level summary on multiple error metrics for Takri dataset.

Error Metric	VOLTAGE
E2E (End to End)	18%
WER (Word Error)	12%
CER (Character Error)	4%

Table 3: Misinterpretation of characters, due to similarity of glyph or part and whole relationships

Actual	Recognised	Type of error
		Over Segmentation
		
		
		Mis Classification

help in improving the overall accuracy of recognised text, considering linguistic context along with the syntax. We have identified a set of generic and specialised linguistic rules for Indic scripts, and apply them towards the end of our pipeline.

5 Results and Discussion

Quantification of errors in OCR pipeline, specially for very low resource scripts is ambiguous unless properly defined (Lopresti, 2008). The document page needs to be segmented into lines, words, characters and symbols for OCR engine. Any error which occurs in this step would fall under layout segmentation error. Most OCR literature do not include segmentation errors as part of OCR errors.

OCR systems generally consider errors either for the entire End to End (E2E) pipeline, at word level or character levels. E2E includes errors at various stages of pipeline such as Pre-processing, Segmentation, Classification, and Post processing. Word recognition accuracy or Word Error Rate (WER) is the average percentage of mis-recognised words. Character Errors Rate (CER) is the ratio of mis-recognised symbols within accurately segmented symbols.

$$WER = \frac{N_w^{'}}{N_w} \quad \& \quad CER = \frac{N_s^{'}}{N_s}$$

where $N_w^{'}$ is count of mis-recognised words, N_w is correctly segmented words, $N_s^{'}$ is count of mis-recognised symbols, and N_s is correctly segmented symbols.

The concept for character/symbol is also ambiguous unless defined clearly, due to the linguistic peculiarity for each script. Symbols are monolithic for *Alphabetic* languages (like English) which uses each symbol in same form, however for *Abugida* scripts (like Indic scripts) symbols are of multiple types, namely (a) Root Symbols: like 𑚊 ("Ka") and 𑚌 ("Ga") and (b) Modifier/ Marker Symbols: like vowel modifiers 𑚰 ("U") and 𑚵 ("Au") (Daniels, 2017). We consider root and markers separately independent of each other in our empirical study. An error in root symbol does not contribute to the error in associated marker symbols.

Most OCR studies, considers errors within (0-2%), (2-10%) and (>10%) as good, average and Poor (Holley, 2009), respectively. However for Indic scripts with limited training data and unknown vocabulary along with heterogeneous handwritten forms, a CER value as high as around (10-20)% is considered satisfactory (Shaffi and Hajamohideen, 2021; Tomoiaga et al., 2019). We compute many error metrics (see Table 2) but observed that all metrics are co-related. With this observation and existing practices, we use the standard metric, CER, for further evaluation not including errors during segmentation and pre-processing (Holley, 2009).

Table 4: Empirical study for VOLTAGE on Takri on Machine Printed (MP) and Hand Written (HW) samples.

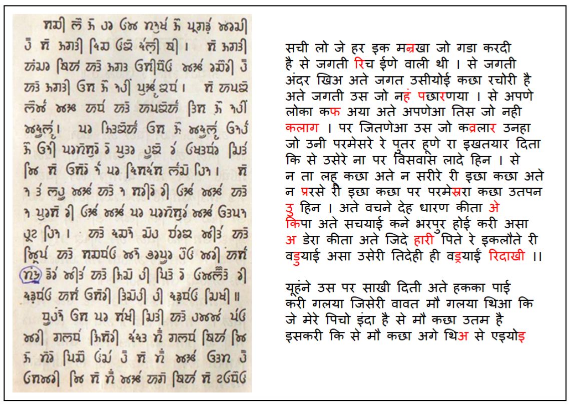
Zone	MP	HW
UZ (Upper Zone)	96%	88%
MZ (Middle Zone)	94%	85%
BZ (Bottom Zone)	97%	89%

5.1 Empirical study on Takri

We evaluate our results both for *machine printed* and *handwritten* samples. We identify a group of 22 participants (13 male, 9 female; diverse age groups; belonging to Himachal Pradesh) who were made familiar with our work and Takri characters. We asked them to record symbols and label them.

We also evaluate our results separately for each zone and analyse the results. As illustrated in Table 4 we make the following observations, (a) The upper-zone symbols which account for approx. 16% of corpus have recognition accuracy of 96% (88% for handwritten samples). (b) The middle-zone symbols account for the majority of characters and is most busiest zone. This contributes for approx 79% of symbols and recognition accuracy is 94% (85% for handwritten samples). (c) The bottom-zone is the most infrequent zone with approx. 5% symbols and recognition accuracy is 97% (89% for handwritten samples).

Figure 4 presents an illustrative example contain-


सची लो जे हर इक मन्नखा जो गडा करदी
है से जगती रिच ईणे वाली थी । से जगती
अंदर खिअ अते जगत उसीयोई कछा रचोरी है
अते जगती उस जो नहं पछारणया । से अपणे
लोका कफ अया अते अपणेआ तिस जो नही
कलाग । पर जितणेआ उस जो कव्रलार उनहा
जो उनी परमेसरे रे पुतर हूणे रा इखतयार दिता
कि से उसेरे ना पर विसवासे लादे हिन । से
न ता लहू कछा अते न सरीरे री इछा कछा अते
न प्ररसे री इछा कछा पर परमेस्रा कछा उतपन
उु हिन । अते वचने देह धारण कीता अे
किपा अते सचयाई कने भरपुर होई करी असा
अ डेरा कीता अते जिदे हारी पिते रे इकलौते री
वडुयाई असा उसेरी तिदेही ही वड्रयाईं रिदाखी ।।

यूहंने उस पर साखी दिती अते हकका पाई
करी गलया जिसेरी वावत मौ गलया थिआ कि
जे मेरे पिचो इंदा है से मौ कछा उतम है
इसकरी कि से मौ कछा अगे थिअ से एइयोइ

Figure 4: Applying OCR at page level.

Table 5: Evaluation across other scripts. For Gujarati we experimented with two scenarios, (a) Gujarati LRL- Like low resource language and (b) Gujarati HRL- like high resource language

Script Name	Script type	Language	SOTA Accuracy	VOLTAGE Accuracy	Dataset size	Label count	Glyph features
Takri	Abugida	Multiple	NA	95%	14,051	59	9
Modi	Abugida	Marathi	(84-94)%	93%	7,221	46	11
Ol Chiki	Alphabet	Santali	(83-92)%	91%	8,873	30	5
Gujarati LRL	Abugida	Gujarati	(86-96)%	93%	7,643	42	9
Gujarati HRL	Abugida	Gujarati	(86-96)%	96%	200K+	42	9
Wancho	Alphabet	Wancho	NA	91%	6,500	42	8

ing 18 lines with 162 words and we pass this via VOLTAGE. We observe that for this sample, 19 words are mis-recognized in recognised output (illustrated in red in the figure), thereby getting E2E error of 12%.

5.2 Applying the methodology to other scripts

We apply our methodology to four other diverse Indic scripts (*Modi, Ol Chiki, Gujarati and Wancho*) to validate the overarching effectiveness of our work. While Modi and Gujarati belong to abugidas family of scripts with ancient history, Ol Chiki and Wancho are more modern alphabetic scripts. Modi, Ol Chiki and Wancho are very low on resources and less explored (Barman et al., 2022; Chandankhede and Sachdeo, 2023b; Das). Gujarati is more popular and resourceful but we use only limited data for our experiment, treating this as low resource experiment (Goswami and Mitra, 2017, 2016) and also as HRL.

The purpose of using a mix of Indic scripts is to validate the all-inclusive application of our design across multiple types of scripts. It is evident from our results in Table 5 that VOLTAGE generalizes well as it provides consistent results across scripts and can be very useful for scripts where labelled data is a scarce.

5.3 Baseline studies

We use the available annotated data-sets for multiple high resource Indic script and fine tune for Takri (Jawahar et al., 2010). As illustrated in Table 6 we observe the following issues with this approach, (a) The choice of what foundation script to choose is very important. In case of Takri we observed using Gujarati gives best results. The choice of foundation model has lot of manual intervention hence we did not include this in the overall process. (b) We restricted our few shot experiment till 100K, since moving to higher numbers would need lot of data which is not feasible for ultra low scripts, also leads to catastrophic interference thereby defying the purpose of using a foundation model. We also see that as the number of samples go up, most models converge to similar results, and far from that of obtained by VOLTAGE. Appendix G further illustrates the size and source of individual data sets across multiple scripts. We also illustrate the SOTA accuracy for these data rich scripts and compare with the results we have achieved.

5.4 Ablation studies

We conduct ablation studies for VOLTAGE to substantiate the contributions of individual elements along with improved model understanding. We already illustrated in Section 4.2 that the application of GFRS improves annotation accuracy by 27%. We conduct more experiments to substantiate the importance of each step in the whole pipeline.

We use basic CNN-LSTM models on each zone separately and test them. We train three separate sets of models for each zone. Within each zone there are separate models subject to the source of training data used, thereby resulting in total of nine models (see Table 7). These models can be clas-

Table 6: Applying annotated Takri dataset to other script OCR, and evaluating accuracy.

Foundation Script	Training samples (Takri)			
	10K	30K	50K	100K
Devanagiri	59%	81%	83%	86%
Gurumukhi	54%	79%	81%	86%
Gujarati	61%	83%	84%	87%
Oriya	56%	79%	82%	87%
Bangali	58%	74%	81%	85%
Tamil	39%	61%	71%	78%

Table 7: Symbol counts in Takri dataset and models used

Sr. No.	**Transformations**	**UZ**		**MZ**		**BZ**	
		count	**model**	**count**	**model**	**count**	**model**
1	None (Actual images)	2981	B^{uz}	9702	B^{mz}	1368	B^{bz}
2	Image Transformations	28,563	E_1^{uz}	110,012	E_1^{mz}	15,931	E_1^{bz}
3	Generative Images	40,427	E_2^{uz}	164,162	E_2^{mz}	21,403	E_2^{bz}

sified as (i) three Base models one for each zone (B^{uz}, B^{mz}, B^{bz}) which include only the actual images extracted from source documents for training, (ii) three Enriched$_1$ models (E_1^{uz}, E_1^{mz}, E_1^{bz}) which include data generated by image transformations along with data used in base models, and (iii) three Enriched$_2$ models (E_2^{uz}, E_2^{mz}, E_2^{bz}) which include images generated by GAN along with the data used in E_1 models. We compare these nine models with three models (one for each zone) using contrastive learning as described in our work (V).

As illustrated in Table 8 we observe that VOLTAGE models outperforms the base models for both machine printed and handwritten evaluation.

Table 8: Character error rates (CER) for Machine Printed (MP) and Handwritten (HW) symbols for CNN-LSTM models and compare the results with VOLTAGE models (V)

	Model	**UZ**	**MZ**	**BZ**
Composition		16%	79%	5%
CER-MP	B	06%	08%	06%
	E_1	04%	07%	03%
	E_2	08%	09%	10%
	V	04%	06%	03%
CER-HW	B	21%	27%	21%
	E_1	19%	25%	19%
	E_2	14%	17%	15%
	V	12%	15%	11%

5.5 Use Cases: Takri for the digital world

We have observed that there is dearth of printed books on Takri, hence it is important to facilitate Takri in printed form. We facilitate NLP for Takri by developing two ready to use tools, (a) Transliteration to Takri facilitating digitization of folk literature (via development of standardised individual symbol images, and creating rule based engine to amalgamation) and (b) Synthetic generative models for each symbol in Takri.

Appendix D further illustrates with an example, how our transliteration engine converts text in other languages to Takri in digital format. This can be very instrumental to publish small stories, news headlines etc. and shared in interested community to facilitate the use of the script. We also share our GAN models to be used by fellow researchers for furthering research[2].

6 Conclusion and Future directions

The paper presents a comprehensive unsupervised OCR methodology, VOLTAGE, which includes a novel Glyph feature recommendation system (GFRS) for effective symbol labelling. We developed VOLTAGE using a very low resource language, Takri, and validated its effectiveness and generalization on various Indian scripts. We achieve accuracy at par with SOTA of respective test scripts. We also build use cases for Takri to demonstrate the usefulness of the work. Our work can facilitate the digitization of ultra low resource scripts thereby save them from extension.

As part of our future work, we shall use our method to build more comprehensive datasets along with building the vocabulary for Indic languages to help in error correction during post processing. We also plan to use the method as described to digitize more languages within India partnering with local governments.

Limitations

Glyph feature store is designed keeping in mind, the stroke characteristics of Indic scripts. Hence it can work for all Indic scripts without any modifications, but may need changes for other family of scripts. It is possible to use same design principles and extend the feature stores for any other family of scripts and apply the method as described in our paper.

[2]https://github.com/prawaal/Takri

References

Md Abedin, Tapotosh Ghosh, Tazqia Mehrub, Mohammad Abu Yousuf, et al. 2022. Bangla printed character generation from handwritten character using gan. In *Soft Computing for Data Analytics, Classification Model, and Control*, pages 153–165. Springer.

Alankrita Aggarwal, Mamta Mittal, and Gopi Battineni. 2021. Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, 1(1):100004.

Debaditya Barman, Tuheli Bhattacharya, and Nirmalya Chowdhury. 2022. A deep learning framework for handwritten ol chiki character recognition. *Journal of Scientific Research*.

Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. 2013. Unsupervised transcription of historical documents. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 207–217.

Thomas M Breuel. 2008. The ocropus open source ocr system. In *Document recognition and retrieval XV*, volume 6815, pages 120–134. SPIE.

Syed Saqib Bukhari, Ahmad Kadi, Mohammad Ayman Jouneh, Fahim Mahmood Mir, and Andreas Dengel. 2017. anyocr: An open-source ocr system for historical archives. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 305–310. IEEE.

Chaitali Chandankhede and Rajneeshkaur Sachdeo. 2023a. Handwritten modi lipi barakhadi dataset.

Chaitali Chandankhede and Rajneeshkaur Sachdeo. 2023b. Offline modi script character recognition using deep learning techniques. *Multimedia Tools and Applications*.

Arindam Chaudhuri, Krupa Mandaviya, Pratixa Badelia, Soumya K Ghosh, Arindam Chaudhuri, Krupa Mandaviya, Pratixa Badelia, and Soumya K Ghosh. 2017. *Optical character recognition systems*. Springer.

BB Chaudhuri. 2009. On ocr of major indian scripts: Bangla and devanagari. In *Guide to OCR for Indic Scripts*, pages 27–42. Springer.

BB Chaudhuri. 2010. On ocr of major indian scripts: Bangla and devanagari. *Guide to OCR for Indic Scripts: Document Recognition and Retrieval*, pages 27–42.

BB Chaudhuri, U Pal, and Mandar Mitra. 2002. Automatic recognition of printed oriya script. *Sadhana*, 27:23–34.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Peter T Daniels. 2017. Writing systems. *The handbook of linguistics*, pages 75–94.

Bishakha Das. Creation of scripts in indigenous language learning: The present scenario of arunachal pradesh.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jignesh Dholakia, Atul Negi, and S Rama Mohan. 2010. Progress in gujarati document processing and character recognition. *Guide to OCR for Indic Scripts: Document Recognition and Retrieval*, pages 73–95.

Pasi Fränti and Sami Sieranoja. 2019. How much can k-means be improved by using better initialization and repeats? *Pattern Recognition*, 93:95–112.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Mukesh Goswami and Suman Mitra. 2014. Dataset set for machine printed gujarati character symbols collected from machine printed books.

Mukesh M Goswami and Suman K Mitra. 2016. Classification of printed gujarati characters using low-level stroke features. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 15(4):1–26.

Mukesh M Goswami and Suman K Mitra. 2017. High level shape representation in printed gujarati character. In *ICPRAM*, pages 418–425.

VK Govindan and AP Shivaprasad. 1990. Character recognition—a review. *Pattern recognition*, 23(7):671–683.

George A Grierson. Linguistic survey of india.. vol. ix, part i: Western hindi and panjabi. edited by sir.

George A Grierson. 1909. The languages of the northern himalayas, being studies in the grammar of twenty-six himalayan dialects. by the revt. grahame bailey, bd, ma, mras asiatic society monographs, vol. xii. london, 1908. *Journal of the Royal Asiatic Society*, 41(1):184–189.

Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, and Yuyan Chao. 2017. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70:25–43.

Rose Holley. 2009. How good can it get? analysing and improving ocr accuracy in large scale historic newspaper digitisation programs. *D-Lib Magazine*, 15(3/4).

CV Jawahar, Anand Kumar, A Phaneendra, and KJ Jinesh. 2010. Building data sets for indian language ocr research. *Guide to OCR for Indic Scripts: Document Recognition and Retrieval*, pages 3–25.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

Benjamin Kiessling, Matthew Thomas Miller, Maxim Romanov, and Sarah Bowen Savant. 2017. Important new developments in arabographic optical character recognition (ocr). *Al-Uṣūr al-Wusṭā*, 25(1):1.

Aparna Kokku and Srinivasa Chakravarthy. 2010. A complete ocr system for tamil magazine documents. *Guide to OCR for Indic Scripts: Document Recognition and Retrieval*, pages 147–162.

Vinay Kukreja, Deepak Kumar, Amandeep Kaur, et al. 2020. Gan-based synthetic data augmentation for increased cnn performance in vehicle number plate recognition. In *2020 4th international conference on electronics, communication and aerospace technology (ICECA)*, pages 1190–1195. IEEE.

Mohinder Kumar, MK Jindal, and Munish Kumar. 2022. Distortion, rotation and scale invariant recognition of hollow hindi characters. *Sādhanā*, 47(2):1–6.

Gurpreet S Lehal. 2010. A complete machine-printed gurmukhi ocr system. *Guide to OCR for Indic Scripts: Document Recognition and Retrieval*, pages 43–71.

Laurence Likforman-Sulem, Abderrazak Zahour, and Bruno Taconet. 2007. Text line segmentation of historical documents: a survey. *International Journal of Document Analysis and Recognition (IJDAR)*, 9(2):123–138.

Daniel Lopresti. 2008. Optical character recognition errors and their effects on natural language processing. In *Proceedings of the second workshop on Analytics for Noisy Unstructured Text Data*, pages 9–16.

Shikha Magotra, Baijnath Kaushik, and Ajay Kaul. 2019. A database for printed takri class of north-west indian regional scripts. In *International Conference on Futuristic Trends in Networks and Computing Technologies*, pages 508–520. Springer.

Shikha Magotra, Baijnath Kaushik, and Ajay Kaul. 2021. Use of classification approaches for takri text challenges. In *Information and Communication Technology for Competitive Strategies (ICTCS 2020)*, pages 403–410. Springer.

Xiao Pan, Mingxuan Wang, Liwei Wu, and Lei Li. 2021. Contrastive learning for many-to-many multilingual neural machine translation. *arXiv preprint arXiv:2105.09501*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Maurice Roux. 2018. A comparative study of divisive and agglomerative hierarchical clustering algorithms. *Journal of Classification*, 35(2):345–366.

Punam K Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. 2016. A survey on skeletonization algorithms and their applications. *Pattern recognition letters*, 76:3–12.

Naresh Saini, Promodh Pinto, Aravinth Bheemaraj, Deepak Kumar, Dhiraj Daga, Saurabh Yadav, and Srihari Nagaraj. 2022. Ocr synthetic benchmark dataset for indic languages. *arXiv preprint arXiv:2205.02543*.

Noushath Shaffi and Faizal Hajamohideen. 2021. uthcd: a new benchmarking for tamil handwritten ocr. *IEEE Access*, 9:101469–101493.

Archana A Shinde and DG Chougule. 2012. Text preprocessing and text segmentation for ocr. *International Journal of Computer Science Engineering and Technology*, 2(1):810–812.

Chang Shu, Yusen Zhang, Xiangyu Dong, Peng Shi, Tao Yu, and Rui Zhang. 2021. Logic-consistency text generation from semantic parses. *arXiv preprint arXiv:2108.00577*.

Ray Smith. 2007. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29.

Andreas Spruck, Maximiliane Hawesch, Anatol Maier, Christian Riess, Jürgen Seiler, and André Kau. 2021. 3d rendering framework for data augmentation in optical character recognition. In *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*, pages 1–4. IEEE.

Ralf C Staudemeyer and Eric Rothstein Morris. 2019. Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer.

Martin Tomaschek. 2018. Evaluation of off-the-shelf ocr technologies. *Bachelor thesis Masaryk University, Brno, Czech Republic*.

Ciprian Tomoiaga, Paul Feng, Mathieu Salzmann, and Patrick Jayet. 2019. Field typing for improved recognition on heterogeneous handwritten forms. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 487–493. IEEE.

UNESCO. 2021. Unesco project: Atlas of the world's languages in danger. *https://unesdoc.unesco.org/ark:/48223/pf0000192416*.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. 2020. Scan: Learning to classify images without labels. In *European conference on computer vision*, pages 268–285. Springer.

Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2).

Christoph Wick, Christian Reul, and Frank Puppe. 2018. Calamari-a high-performance tensorflow-based deep learning package for optical character recognition. *arXiv preprint arXiv:1807.02004*.

Kai Wu, Dingjiang Yan, Hongcheng Liao, Xiang Zhang, Qilin Huang, Qian Zhang, and Min Fu. 2021. Application of data augmentation of rare words based on font cyclegan in character recognition. In *2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT)*, pages 477–479. IEEE.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Rui Zhang, Yangfeng Ji, Yue Zhang, and Rebecca J Passonneau. 2022. Contrastive data and learning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 39–47.

Acknowledgements

We would like to thank Shikha Magotra for compiling the digital database of *Takri* script from multiple museums in north-western Indian geography and making it available for our research.

Appendix A: Glyph feature store

Feature ID	Feature Description	Type	Range
F1	**Presence of Headline**: Checks for the presence of a horizontal line on the top of the sub-symbol.	Boolean	0/1
F2	**Number of Loops**: Counts the number of loops in the symbol including loops with headline.	Number	0-N
F3	**Number of Loops with headline**: Counts the number of loops the symbol makes with the headline (F1).	Number	0-N
F4	**Presence of left-sidebar**: Check for the presence of a vertical line on the left-most side of sub-symbol.	Boolean	0/1
F5	**Presence of right-sidebar**: Check for the presence of a vertical line on the right-most side of sub-symbol.	Boolean	0/1
F6	**Number of connected components**: Counts the number of sub-symbols which are connected.	Number	0-N
F7	**Number of endpoints**: Counts the number of points, which have only one black pixel in its 3x3 neighbourhood.	Number	0-N
F8	**Number of junctions**: Counts the number of points, which have more than two black pixel in its 3x3 neighbourhood.	Number	0-N
F9	**Number of junctions with headline**: Counts the number of junctions (F8) which touch the headline (F1).	Number	0-N
F10	**Number of bend points clockwise**: Counts the number of points which makes 90 degree turn towards right direction.	Number	0-N
F11	**Number of bend points anti-clockwise**: Counts the number of points which makes 90 degree turn towards left direction.	Number	0-N
F12	**Aspect Ratio**: Ratio of symbol height and width on 0-100 scale.	Number	0-100
F13	**Horizontal Symmetry**: Flip the image on Y axis (mirror the image in left-right perspective). Find the similarity with original image using threshold value.	Boolean	0/1
F14	**Vertical Symmetry**: Flip the image on X axis (mirror the image in top-down perspective). Find the similarity with original image using threshold value.	Boolean	0/1
F15	**Number of Dots**: Counts the number of points, which have zero black pixels in its 3x3 neighbourhood.	Number	0-N
F16	**Number of left-right layers**: Counts the number of layers of black pixels on x axis. This relates to the maximum isolated black pixels in horizontal cross section.	Number	0-N
F17	**Number of top-down layers**: Counts the number of layers of black pixels on y axis. This relates to the maximum isolated black pixels in vertical cross section.	Number	0-N
F18	**Minimum horizontal projection**: Compute the horizontal projection (count the number of black pixels across y axis for every x axis) and find the minimum value. Scale this by taking ratio with height of image, and multiple by 100,	Number	0-100
F19	**Minimum vertical projection**: Compute the vertical projection (count the number of black pixels across x axis for every y axis) and find the minimum value. Scale this by taking ratio with width of image, and multiple by 100,	Number	0-100

F20	**Maximum horizontal projection**: Compute the horizontal projection (count the number of black pixels across y axis for every x axis) and find the maximum value. Scale this by taking ratio with height of image, and multiple by 100,	Number	0-100
F21	**Maximum vertical projection**: Compute the vertical projection (count the number of black pixels across x axis for every y axis) and find the maximum value. Scale this by taking ratio with width of image, and multiple by 100,	Number	0-100
F22	**Maximum left depth**: The maximum depth of the left profile calculated as a percentage with respect to total width of the box enclosing the symbol.	Number	0-100
F23	**Maximum right depth**: The maximum depth of the right profile calculated as a percentage with respect to total width of the box enclosing the symbol.	Number	0-100
F24	**Maximum top depth**: The maximum depth of the top profile calculated as a percentage with respect to total width of the box enclosing the symbol.	Number	0-100
F25	**Maximum bottom depth**: The maximum depth of the bottom profile calculated as a percentage with respect to total width of the box enclosing the symbol.	Number	0-100
F26	**Minimum left depth**: The minimum depth of the left profile calculated as a percentage with respect to total width of the box enclosing the symbol.	Number	0-100
F27	**Minimum right depth**: The minimum depth of the right profile calculated as a percentage with respect to total width of the box enclosing the symbol.	Number	0-100
F28	**Minimum top depth**: The minimum depth of the top profile calculated as a percentage with respect to total width of the box enclosing the symbol.	Number	0-100
F29	**Minimum bottom depth**: The minimum depth of the bottom profile calculated as a percentage with respect to total width of the box enclosing the symbol.	Number	0-100
F30	**Stroke length**: Count of total black pixels as a percentage with total area (height * width) of the symbol box.	Number	0-100
F31	**Epicenter top down**: Find the mean of all black pixels, and compute the location of Y axis from center as a percentage from mid point on y axis .	Number	0-100
F32	**Epicenter left right**: Find the mean of all black pixels, and compute the location of X axis from center as a percentage from mid point on x axis .	Number	0-100

Recommended Feature set for languages:

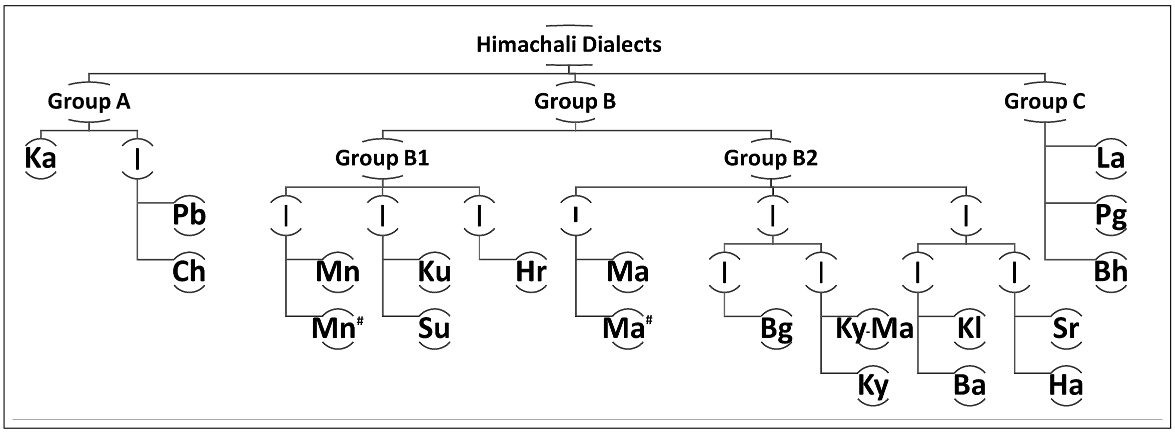
- Takri - F1, F2, F5, F7, F8, F12, F13, F14, F15.
- Modi - F1, F2, F4, F5, F7, F9, F12, F15, F16, F23, F30 .
- Ol Chiki - F2, F4, F8, F12, F16.
- Gujarati - F1, F2, F4, F5, F7, F12, F13, F15, F16.
- Wancho - F2, F5, F8, F12, F13, F15, F16, F21.

Appendix B: Post processing rule inventory

In our work for Takri OCR, we have applied rules from R1 to R7. We could not apply R8, R9, R10 due to non availability of dictionary for Takri.

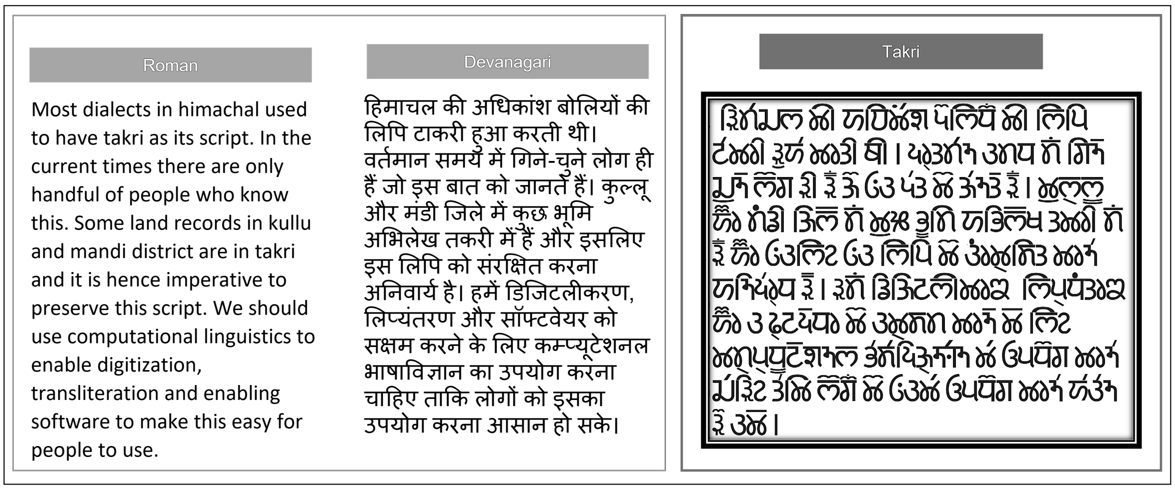
Rule ID	Rule description	Remarks
R1	One consonant should not contain more than one vowel modifier.	Generic rule for Indic scripts. However some fine tuning may be needed from script to script basis.
R2	Sentence delimiter should not happen in between of a word.	
R3	Sentence delimiters should not repeat consecutively.	
R4	Numbers should not merge with consonants or vowels within the word boundary.	
R5	Independent vowels symbols (excluding the modifiers) can occur only at the start of end of the word in most Indic scripts.	
R6	*Panchamkshar* Characters (like 𑚓 ("Nya") and 𑚎 ("Nga") in case of Takri) should not occur in the end of the word.	
R7	Unicode sequence of symbols identified may not follow the same sequence as required by computing systems, so sequence of unicodes may need to be changed. For example 𑚟𑚮𑚤 ("Fira") identifies vowel "i" before consonant "f" and sequence needs to be changed.	
R8	Shape characteristics of some symbols (glyph features) may follow Whole-Part design. Aggregation of some symbols (parts) may together form another symbol (whole). For example in Takri the "parts" - 𑚤 ("Ra") and । ("Viram") can form "whole" - 𑚌 ("Ga").	Validation with prebuilt vocabulary/dictionary to generate/shortlist candidate word.
R9	Shape characteristics of some symbols may be very similar and create mis-classification. For example 𑚎 ("Nga") and 𑛃 ("Number 3") are very similar	
R10	It is possible that in case of short word partitions, the boundaries of word are not marked correctly. In case the word spans across two lines (with not sufficient space at the end of the line) word partition error can happen as well.	

Appendix C: Languages using Takri as a script and their relationships



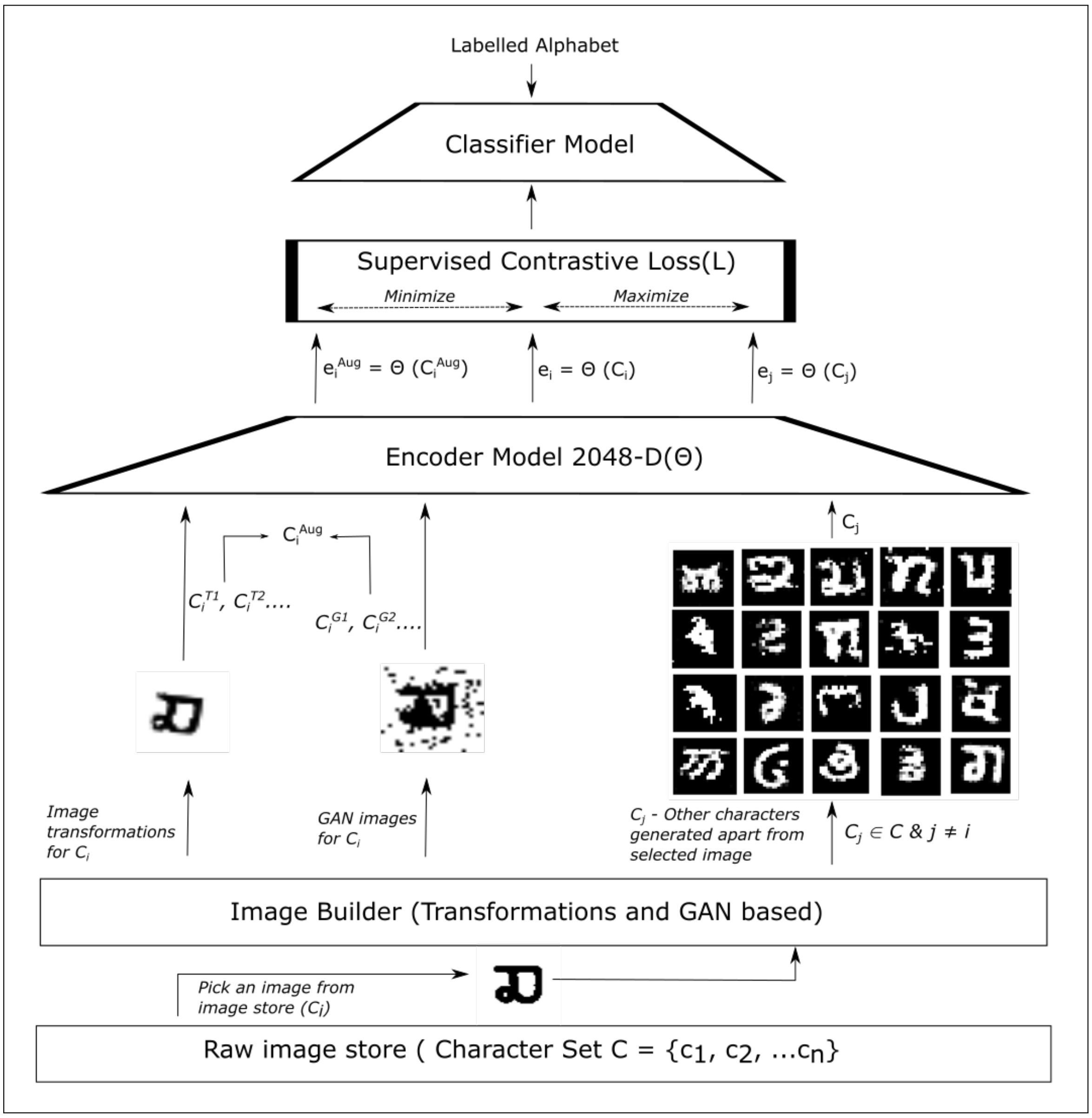
Dialects with inflections are represented with superscript ('). (Baagli (Ba), Bhagati (Bg), Bharmauri (Bh), Chambiali (Ch), Hamirpuri (Hr), Hatti (Ha), Kalhuri (Kl), Kangri (Ka), Kinnauri (Kn), Kulluvi (Ku), Kyunthali (Ky), Lahauli (La), Mahasuvi (Ma), Mandiali (Mn), Pangwali (Pg), Punjabi (Pb), Sirmauri (Sr), Suketi (Su))

Appendix D: Transliteration Sample to Takri using custm made glyhs.



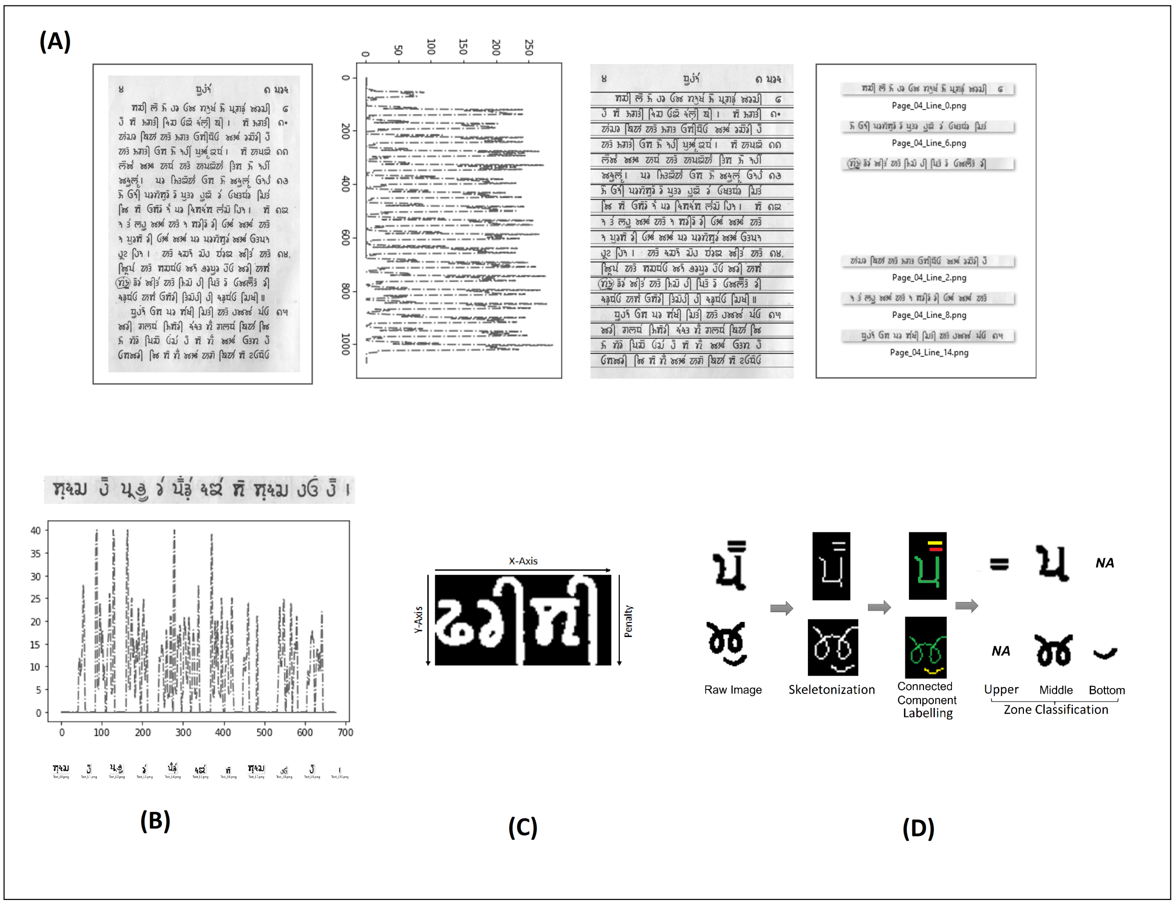
Roman	Devanagari	Takri
Most dialects in himachal used to have takri as its script. In the current times there are only handful of people who know this. Some land records in kullu and mandi district are in takri and it is hence imperative to preserve this script. We should use computational linguistics to enable digitization, transliteration and enabling software to make this easy for people to use.	हिमाचल की अधिकांश बोलियों की लिपि टाकरी हुआ करती थी। वर्तमान समयें में गिने-चुने लोग ही हैं जो इस बात को जानते हैं। कुल्लू और मंडी जिले में कुछ भूमि अभिलेख तकरी में हैं और इसलिए इस लिपि को संरक्षित करना अनिवार्य है। हमें डिजिटलीकरण, लिप्यंतरण और सॉफ्टवेयर को सक्षम करने के लिए कम्प्यूटेशनल भाषाविज्ञान का उपयोग करना चाहिए ताकि लोगों को इसका उपयोग करना आसान हो सके।	[illegible]

Appendix E: Use of supervised contrastive learning for character identification


Labelled Alphabet
Classifier Model
Supervised Contrastive Loss(L)
Minimize
Maximize
$e_i^{Aug} = \Theta(C_i^{Aug})$
$e_i = \Theta(C_i)$
$e_j = \Theta(C_j)$
Encoder Model 2048-D(Θ)
C_j
C_i^{Aug}
$C_i^{T1}, C_i^{T2}....$
$C_i^{G1}, C_i^{G2}....$
Image transformations for C_i
GAN images for C_i
C_j - Other characters generated apart from selected image
$C_j \in C$ & $j \neq i$
Image Builder (Transformations and GAN based)
Pick an image from image store (C_i)
Raw image store (Character Set C = $\{c_1, c_2, \ldots c_n\}$

We use Google colab for running our experiment. For GAN images we train for 150 epochs, 4 layer generator and network, adam optimiser and learning rate of 0.0002.

Appendix F: Extraction of source raw data into lines, words, characters and symbols.



- A - Segmentation of page into lines.
- B - Segmentation of line into words.
- C - Segmentation of word into characters.
- D - Segmentation of character into symbols.

Appendix G: Baseline studies.

TABLE A -

Script Name	Annotated Symbols
Devanagiri	121K
Gurumukhi	111K
Gujarati	121K
Oriya	115K
Bangali	122K
Tamil	116K

The annotated dataset is taken from (Jawahar et al., 2010) which is the largest corpus of annotated Indic scripts. We have limited the size of the corpus to approx (110-120)K symbols to have all scripts in similar size. We used this initial dataset to build a base model for that script and later fine tune it for Takri.

TABLE B -

Script Name	SOTA Accuracy on base script	Accuracy on Takri with 100K labelled dataset
Devanagiri	97.9% (Chaudhuri, 2010)	86%
Gurumukhi	96% (Lehal, 2010)	86%
Gujarati	96% (Dholakia et al., 2010)	87%
Oriya	96% (Chaudhuri et al., 2002)	87%
Bangali	97% (Chaudhuri, 2010)	85%
Tamil	(94-97)% (Kokku and Chakravarthy, 2010)	78%

We used our baseline models created using annotated symbol (from Table A) and used 100K annotated Takri tokens to fine tune for our purpose. SOTA accuracy (in second column) is for base script (as mentioned in column one) and Accuracy on Takri (in last column) is for Takri script.