

# Devanagari optical character recognition of printed text

Malathi P.<sup>1</sup>, Chandrakanth G. Pujari<sup>2</sup>

<sup>1</sup>Department of Information Science and Engineering, Dr. Ambedkar Institute of Technology Bengaluru, affiliate to Visvesvaraya Technological University, Belagavi

<sup>2</sup>Department of Master of Computer Applications, Dr. Ambedkar Institute of Technology Bengaluru, affiliate to Visvesvaraya Technological University, Belagavi

## Article Info

### Article history:

Received Dec 18, 2024

Revised Jul 16, 2025

Accepted Sep 14, 2025

### Keywords:

Convolutional neural network

Dataset complexity

Devanagari script

Diacritic

Lexicon

Low resource

## ABSTRACT

Hundreds of native languages and scripts are making their way on digital platform to sustain in multiple data formats. Optical character recognition (OCR) is one such dimension where the low resource languages are yet to find their stability. Devanagari OCR is one such low resource script problem to be dealt with, though it is the fourth widely used global script. Recent works carried on OCR have focused on word level approach and face challenges of spiraling complexity as language alphabet set size crosses hundreds. Most of these OCR works are done in constrained environment, with huge datasets and large computational resources. As a result, effective benchmark evaluation of the works against one another on defined metrics is scarce. Aim here is to explore character level Devanagari OCR with printed text images as input. Pattern recognition (PR) principles for diacritic classification and convolutional neural network (CNN) for base character classification are used. word error rate (WER) of 24.47% is attained. However, the training dataset complexity is reduced by 4.35 times. The ten multi class models, training time range from 45 minutes to 2.5 hours. Further the models can be trained in parallel to complete the training process in 3-4 hours. Thus, the approach used for text classification facilitates the Devanagari OCR solution to be offered in off-the-shelf computing devices.

*This is an open access article under the [CC BY-SA](#) license.*



## Corresponding Author:

Malathi P.

Department of Information Science and Engineering, Dr. Ambedkar Institute of Technology, affiliate to Visvesvaraya Technological University

Bengaluru- 560056, Karnataka, India

Email: malathip.is@drait.edu.in

## 1. INTRODUCTION

Hundreds of native languages and scripts are making their way on digital platform to sustain in multiple data formats like text, images, audio and video. In the present scenario, under digital text handling optical character recognition (OCR) forms the corner stone as vast amount of text to be extracted still lays in image format which can only be addressed with open-source solutions. Further, the solutions developed are more focused towards nurturing the high resource languages which have notable benchmark datasets for addressing the OCR problem. Emergence of variations in OCR like scene text OCR, online OCR have further cluttered the research works on the more notable, vital and default form, the document OCR. Natural language processing deals with semantic handling of digitized text and can be part of post processing the results of OCR. Research in natural language processing (NLP) though started late than OCR has gained better momentum in handling data both in text and audio and video format and shadowed its interests. State-of-the-art (SOTA) deep learning techniques adopted for OCR solutions thrive on large datasets and often tightly coupled with pre-processing and post-processing stages in the pipeline that effective metrics for

comparison among the contemporize works is fuzzy. Emerging technologies like large learning models that focus on end-to-end solutions are further taking to next level of sophistication. Thus, lacunae of benchmark datasets for document OCR of many regional languages lands them as low resource languages and Indic scripts are part of this category. Devanagari OCR is one among the low resource Indic scripts to be dealt with, even though it is the fourth widely used and popular global script. Works carried out on Devanagari OCR are found to be highly influenced with high resource OCRs and hence their distinguishing features are less addressed. Presently the proprietary solutions developed for Devanagari OCR are characterized with the use of large datasets for training machine learning models, consuming abundant computing resources in distributed and cloud computing environment. This impacts the affordability, customization and flexibility to adapt to applications in diverse domains and hence the sustenance of the Devanagari OCR. Aim here is to explore the use of conventional pattern recognition techniques and machine learning techniques in a balanced systemic manner to address the challenges mentioned above and contribute to building open-source Devanagari OCR for printed text. The work handles input images at character level distinct from the STOA works which address at the word, phrase or line level input images. Accuracy of 75.73% in character prediction is achieved with the dataset complexity reduced by 4.35 times. Consequentially the solution can be used in the off the shelf computing devices or incorporated in embedded systems.

## 2. PREVIOUS WORK

### 2.1. Existing system

Google provides the conventional tesseract 4 as an open-source engine based on LSTM available in standalone form. While Google cloud vision is their proprietary image analysis web platform with CUDA support for image labelling, face and landmark detection, OCR, and safe search based on deep learning principles. Thammarak *et al.* [1] in their work on “Comparative analysis of Tesseract and Google Cloud Vision for Thai vehicle registration certificate” brings out the differences between the two OCR platforms. Summary of their accuracy based on input images size is shown in the Table 1. Clearly licensed Cloud Vision is far effective than open-source Tesseract 4 and also provides additional features to facilitate automation pipeline.

Table 1. Tesseract vs Google Cloud vision for Thai plate recognition

Image Size	Accuracy (%)	
	Tesseract	Cloud Vision
Large	60.21	94.72
Standard/Medium	41.17	79.85
Small	41.21	71.17
Average	47.02	84.43

Sun *et al.* [2] have made elaborate observations about the development of the Google Cloud Vision models using pretrained ResNet-101 and the Googles internal dataset JFT-300M backed by meticulous preprocessing phase. Classification size of 18,291 labels clearly reflects the complexity of the models trained and the integration of distributed computing resources for the training process. Highlight is the use of asynchronous gradient descent training on 50 NVIDIA K80 GPUs. 17 parameter servers were used to store and update the model weights. To accommodate training of such complex architecture, models are split vertically into 50 equal sized sub-fully connected layer and distributed around different parameter servers. Searching the right set of hyper-parameters requires significant efforts. Extent of time taken for training a JFT model is hinted where 4 epochs needed 2 months on 50 K-80 GPUs. Such gigantic models are less ideal and feasible for evolving dynamic needs of the OCR use cases.

EasyOCR [3] is an open access OCR solution offered by JAIED AI community group. The architecture, options for pre-processing, post-processing phases and flexibility to extend to other languages are mentioned in their documentation. In their official paper, it is stated that F1-score reaches to 0.856. Brief insight to the effort involved is expressed as 14h to train with minimum manual supervision for 25k iteration with 8 RTX 3090 Ti system resources. Half of the GPU was assigned for training, and half of GPU assigned for supervision setting. This clearly asserts the STOA solutions offered are out of the reach of the off shelf or medium sized computing resources.

Ignat *et al.* [4] in their work have facilitated publicly available novel benchmark, OCR4MT, consisting of real and synthetic data, enriched with noise, for 60 low-resource languages in low resource scripts. Main observation of their results was that the best avail-able OCR systems work well on Latin scripts and perform significantly worse on non-Latin and non- European scripts (*e.g.*, Perso-Arabic, Khmer). Monolingual datasets thus framed serves as a valuable source of data augmentation for future research in

improving machine translation for low resource languages. It clearly highlights the lacunae of benchmark datasets till recent years.

Fan *et al.* [5], Wenzek *et al.* [6], Goyal *et al.* [7] in their findings have expressed that very large fraction of the languages spoken by the world's population have low resource support for developing the OCR solutions. Smith *et al.* [8], Wick *et al.* [9] in their observation have inferred that most of the OCR models have only been evaluated on a handful of languages. Non availability of public benchmarks for low resource scripts and languages is the bottleneck. As a result, metrics and tools for comprehensive evaluation of OCR solutions, particularly for low-resource languages and scripts, is still an open problem.

Rijhwani *et al.* [10] found that endangered language linguistic archives contain thousands of scanned documents and such language documents often contain a translation into another (usually high resource) language. They also observed that the available general purpose OCR solutions are not robust to the data-scarce setting of endangered languages. To address this problem, OCR post-correction modules are developed which are tailored to ease the training process. This enlightens the need for lexicon free OCR approaches to facilitate historical documents translation.

Ding *et al.* [11] work is based on compressing CNN-DBLSTM models for OCR using teacher-student learning and Tucker decomposition approach. Integrated convolutional neural network (CNN) and deep bidirectional long short-term memory (DBLSTM) based character models have achieved excellent recognition accuracies on OCR tasks. Models thus devised comprise of huge amount of model parameters and massive computation cost. So, to deploy CNN-DBLSTM models in products with CPU server, urgent need to compress and accelerate them as much as possible is suggested. Especially the focus on CNN part, which dominates both parameters and computation is pointed. The study clearly draws attention to the need of in depth focus on performance and efficiency factors of machine learning based models.

Gongidi and Jawahar [12] work is on handwritten text recognition for Indic scripts. Spatial transformer network (STN) with affine transformation (ATN), and thin-plate spline transformation (TPS) for rectifying variations in input images of phrases. Deep neural networks, VGG and RESNET for feature extraction. Bidirectional LSTM (BLSTM) is used for sequencing the extracted features. Connectionist temporal classification (CTC) for sequence prediction. The work is influenced by EasyOCR. The total parameters vary from 8.35 M to 17.15 M for model versions. On Bengali language of Devanagari script, CER=4.85%, WER=14.77% for the vocabulary set trained, CER=3.71%, WER=16.65% outside the vocabulary considered. In-spite of in-built word error correction principle used and standard deep learning architectures tried; error rate is noteworthy. The results indicate challenges, unique of the Indic scripts to be handled.

## 2.2. Summary

Domain specific customization is the main need for real time applications of OCR. For example, need of English numerals, selected set of special symbols to be part of Devanagari OCR alphabet set has to be considered. In such circumstances, machine learning based models fall apart as they are rigid and are not easily extensible. Customization is as good as building and training the model from the scratch, for complex models this results to be daunting task. This aspect for large alphabet sets triggers to explore the effective ways of reducing the complexity of OCR solutions. Further conventional approaches are lexicon based, which again makes the solutions ineffective when they have to accommodate new words. Thus, character level OCR suffice lexicon free approach requirement also.

## 3. PROPOSED WORK

### 3.1. Design

The design of the proposed work focuses on two main components: the alphabet set and preprocessing.

#### 3.1.1. Alphabet set

Devanagari, script used as base for Sanskrit, Prakrit, Hindi, Gujarati, Marathi, Bengali, Nepali and other regional languages, finds ancestral roots in Brahmi script, from which all the modern Indian writing systems are derived. With 750+ million global users, is the fourth most widely adopted writing system in the world. Still, Devanagari OCR is one such low resourced script short of benchmarked datasets and parametric observations. Modern Devanagari, script is composed of 46 primary characters, including 13 vowels and 33 consonants. It is an abugida, which means the writing system has consonants with an inherent vowel sound called as diacritic. One of the most recognizable features of Devanagari is the horizontal line at the top of the word called Shirorekha that groups the characters. Conjunct consonants, where two or more consonants combine to form a single character, are common in Devanagari and add to the richness and complexity of the script. The basic consonant characters are organized in groups depending on their manner of articulation, *i.e.*,

voiced, voiceless, and nasal. The pronunciation of the characters might vary slightly depending on the language using the Devanagari script. It has its own number representation that follows Hindu-Arabic numeral system.

The alphabet set  $S$  considered for the current work is the major subset of modern Devanagari scripts full-fledged alphabet set. It consists of: i) primary character set of vowels {13}, consonants {33}, commonly used consonant conjuncts {3} shown in Table 2 with their Unicode representation series and ii) Barakhadi character set is composed of conjuncts formed from combination of consonants with vowel diacritics. Sample set of Barakhadi is shown in Table 3.

Table 2. Unicode of Devanagari symbols

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+090X			०	०ः		अ	आ	इ	ई	उ	ऊ	ऋ				ए
U+091X	ऐ			औ	औ	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट
U+092X	ठ	ड	ढ	ण	त	थ	द	ध	न		प	फ	ब	भ	म	य
U+093X	र		ल	ळ		व	श	ष	स	ह					ा	ि
U+094X	ी	ु	ू	ृ				े	ै			ो	ौ			

Table 3. Barakhadi sample

Vowels	अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ	अं	अः
Diacritic		ा	ि	ी	ु	ू	ृ	े	ै	ो	ौ	ं	ः
Barakhadi Sample	क	का	कि	की	कु	कू	कृ	के	कै	को	कौ	कं	कः

Total size of the alphabet set  $S$  considered for classification aggregates to 481(432+49) in the present work. Let  $Vowels=x$ ,  $Consonants=y$ ,  $Special\ conjuncts=z$ , input size of each image for  $training=s$ , then Barakhadi contains  $(x-1) * (y+z)$  distinct characters. Theoretically with conventional approach which use word, phrase or sentence level input for training the dataset, complexity for recognition can then be represented as:

$$S = [x + y + z + (x - 1) * (y + z)] * s \quad (1)$$

With character level approach the dataset complexity for recognition can be represented as:

$$R = [x + y + z + (x - 1) + (y + z)] * s \quad (2)$$

Then reduction of dataset complexity can then be formulated as:

$$F(x, y, z, s) = \frac{[x + y + z + (x - 1) * (y + z)] * s}{[x + y + z + (x - 1) + (y + z)] * s} \quad (3)$$

For Devanagari alphabet set considered in the present work-  $x = 13, y = 33, z = 3, s = 2000$ .

Substituting in (3), reduction of dataset complexity  $F(x, y, z, s) = 7.2$  times. Also, the number of input image samples taken for each of the class for training the model can be evenly distributed. This in turn helps to figure out the classes for which the classification model efficiency is less and corrections to be taken. With conventional approach, distribution of input samples across the alphabet set is neither maintained nor assessed. In consequence they adopt to reverse engineer the dataset size for experiments based on results. Also, computational resources used, and time taken will be tuned to meet the expected efficiency by trial and error.

### 3.1.2. Preprocessing

Sample images of dataset considered for testing is shown in Figure 1(a). EasyOCR API [3] applied on the images gave totally wrong predictions. Google Cloud Vision API [13] responded as bad data. The images were then binarized, Otsu threshold and thinning operations were applied explicitly as shown in Figure 1(b). before calling the EasyOCR API and Google Cloud Vision API to respond appropriately. Thus, clarity of input images also impacts the result of classification. This highlights the role of various pre-processing operations to be used on the customer end to use the existing solutions.



Figure 1. Sample input images (a) before pre-processing and (b) after pre-processing

### 3.2. Algorithm

Modules developed in classification and recognition phases of the OCR pipeline is shown in Figure 2.

- a. Pre-processing: The input grey image is first converted into binary image using histogram-based Otsu binarization method. This helps to eliminate the outlier pixels introduced during document scanning process. The word images are next processed for skew and slant corrections. Scene text OCR has to handle colored input images. Conversion to grey images forms an additional step in that context.
- b. Thinning: It helps to focus on vital pixels that transform to features of the characters and also reduce the overall computations involved. Majority of the prints does not have characters of uniform thickness to apply global thinning operation which can result in misleading features. Refined process called skeletonization available as library is made use with further scope to fine tune it. It forms a way of normalization w.r.t thickness of character in images. It also boosts to generalize the solution, void of font dependence.
- c. Character segmentation: Challenges faced for character images segmentation is depicted in Figure 3(a). On masking Shirorekha in word image, the symbol images segregation shown in Figure 3(b) only leads to wrong predictions. But semantically the character images to be segregated is shown in Figure 3(c). even before the process of text recognition.
- d. Character listing: The characters segmented effectively has to be assembled in the word in the same order at the end of recognition process as we have used word images for testing the built classification model during experiments. Listing is important, particularly when multiple word images are passed as batch input to the OCR pipeline during experiments.
- e. Diacritic segmentation: This phase forms the focal point for effectiveness of the OCR. The possible vowel diacritic position, relative to the base consonant in Devanagari script is depicted in Figure 5. The approach used to segment the diacritic from the base consonant varies for each region. Segmenting the diacritic that lies in the bottom region is the most challenging task. Position of the diacritic relative to the base consonant used varies across the characters in the alphabet set as shown in Figure 4. Hence, this phase plays a pivotal role in the efficiency of the designed OCR solutions.
- f. Diacritic prediction: Pattern matching based technique is used to identify the diacritic. Particularly histograms were used for horizontal and vertical profiling followed by cross sect analysis done on the diacritic region of the input images. Size of the segregated diacritic is comparably smaller than the base consonant in conjunct characters. Also differences between diacritics is represented by very few pixels in the input images especially after thinning process used in step 2. So, it has to be padded on all four sides and enlarged using interpolation technique to facilitate independent diacritic recognition of the conjunct.
- g. Base character localization: Mean smoothing of the short horizontal and vertical lines that are part of vowel or consonant apart from Shirorekha and Matra facilitates fine tuning of character shapes in the images. Resurrection helps to restore the feature pixels lost during thinning in step 2.
- h. Key feature extraction: Top and bottom pendant vertices are the major key features in vowels or base consonants of the character. Their identification helps to differentiate the multi class labels during classification.
- i. Base character (vowel or consonant in conjunct) prediction: Statistical matching, convolutional neural networks (CNN) and transformer vision principles were weighed. It is this module which determines the complexity of the entire system. The main objectives to approach this task were: i) Overall OCR solution to be medium sized solution, ii) Manual effort needed to prepare data set has to be kept in bounds, iii) Computing resources required for the complete process should be implemented with off the shelf devices. Statistical matching, CNN and transformer vision principles which are based on deep learning machines were weighed. Considering the priorities of all the mentioned objectives CNN principle was found to be ideal. Basic shape features like curvatures, lines and their alignment in base character were considered to arrive at a multi-level classification approach. Accordingly the classes categorized as = [क, फ, ऋ], [त, ल], [न, ग], [च, ज, ञ, न, ब, व, त्र, ञ], [थ, ध, भ], [घ, ण, प, म, य, ष], [ख, झ, स, श, क्ष, अ, आ, ओ, औ], [द, ह, इ, ई], [र, ए, ऐ], [ड, छ, ट, ठ, ड, ढ, उ, ऊ]. This results in ten effective small sized trained models than a medium sized single classifier model for use.

- j. Character prediction: The results of diacritic classification and base vowel/consonant classification are combined thru rule-based decisions. Cases where cross check and auto correction required for valid character prediction is performed else unsuccessful prediction is indicated.
- k. Word prediction: The characters predicted are assembled into word as part of word list. The input word images were supplied in batches during testing purpose. This helped us to compare the Cross Sect OCR design with Google Cloud Vision and EasyOCR models on both CER and WER as in Table 4 instead of just CER comparison as character level OCR is our primary approach.

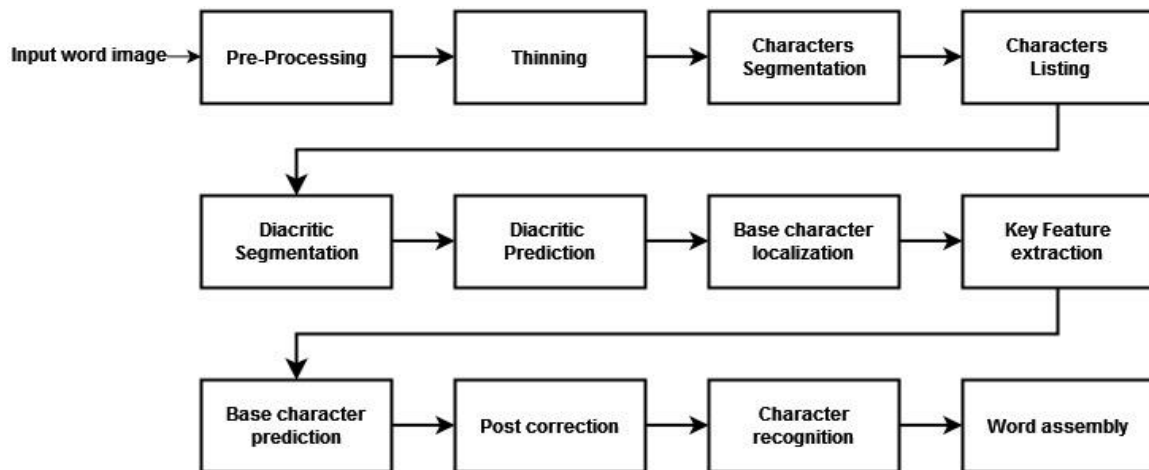


Figure 2. Cross sect OCR algorithm

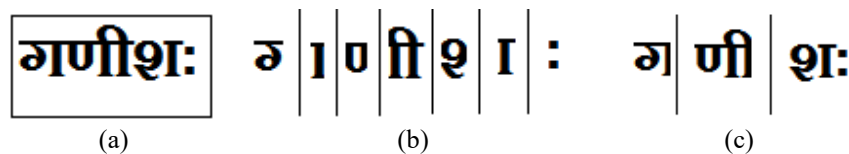


Figure 3. Character segmentation (a) word image, (b) symbol segregation, and (c) character segregation

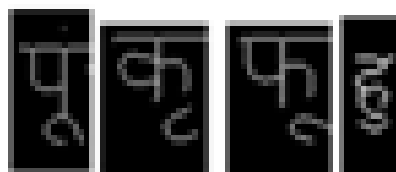


Figure 4. Diacritic positioning

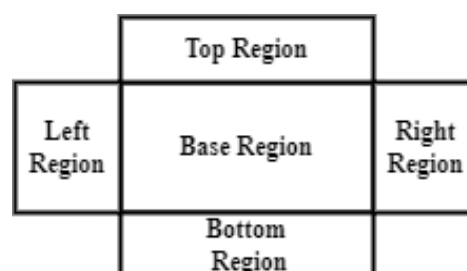


Figure 5. Devanagari character layout

#### 4. EXPERIMENT SETUP

Commonly used, Unicode based Akshara font was used to generate 2000 images for each of the base character (vowel/consonant) set of size 38 as part of the initial train dataset. Google Colab Python platform was used for dataset preparation, model training and OCR pipeline execution. Image data generators were used to generate images to complement the train dataset dynamically. This facilitates the trained classification models to be independent of font used and font size. CNN [14]–[16] was found as an optimum approach keeping MNIST [17] as reference. Accordingly, three models were designed the as shown in Figure 6 and experimented.

Efficiency of model III on training was found to be 3%-4% better than model II and 5%-6% better than model I. Also, the total training parameters of CNN in model III is greatly reduced compared to model I and II. Model III architecture is shown in Figure 7.

Model-I			Model-II			Model-III		
Layer	Output shape	Param	Layer	Output shape	Param	Layer	Output shape	Param
conv2d	(32,32,64)	640	conv2d	(32,32,32)	832	conv2d_1	(28,28,32)	832
conv2d_1	(32,32,128)	73386	conv2d_1	(32,32,32)	25632	max_pooling2d	(14,14,32)	0
conv2d_2	(32,32,256)	291568	conv2d_2	(32,32,32)	25632	conv2d_1	(10,10,64)	51264
batch_Normalization	(32,32,256)	1024	max_pooling2d	(16,16,32)	0	max_pooling2d	(2,2,64)	0
max_pooling2d	(16,16,256)	0	conv2d_3	(16,16,64)	18496	flatten	(256)	0
conv2d_3	(16,16,256)	590080	conv2d_4	(16,16,64)	36928	dense	(7)	2570
conv2d_4	(16,16,512)	1180160	conv2d_6	(16,16,64)	36928	Totalparams:	53895	
							(210.53 KB)	
batch_Normalization_1	(16,16,512)	2048	max_pooling2d_1	(8,8,64)	0			
max_pooling2d_1	(8,8,512)	0	Flatten	4096	0			
conv2d_5	(8,8,512)	2359808	Dense	256	1048832			
conv2d_6	(8,8,512)	2359808	dense_1	256	65792			
batch_Normalization_2	(8,8,512)	2048	dense_2	7	2009			
max_pooling2d_2	(4,4,512)	0	Totalparams:	1259586				
Flatten	8192	0						
Dense	256	2097408						
drop_out	256	0						
dense_1	7	1799						
Totalparams:	8963847							

Figure 6. CNN models configuration for base character experimented

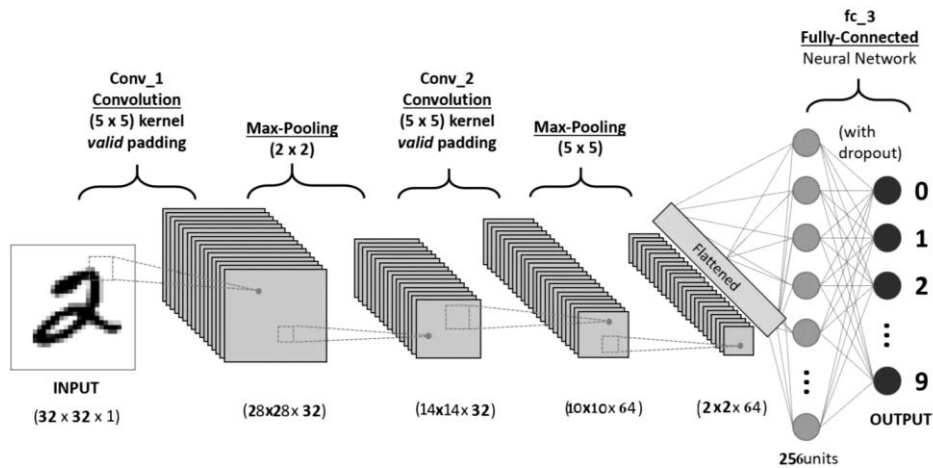


Figure 7. Model III architecture

In all the three models ReLU [18] was the activation function used for the hidden layers and SoftMax activation function in the output layer. Stochastic gradient descent (SGD) with momentum [19]–[21] was the loss evaluation function used for training data in batches. Dropout [22] mechanism was used to prevent overfitting of the dataset. The best chosen design was the CNN model III [23], [24]. Each multi class takes 10-15 iterations of batch dataset to reach accuracy of 90+ and total time of 20-150 minutes for training depending on its size. The training details of a sample multi class [च, ण, ण, ण, ण, ण] is shown in Figure 8. Lowest validation loss on training the ten different multi class models were reached at different epochs. Still, the consistency of the ten multi class models across the basic metrics was found and is depicted in Figure 9.

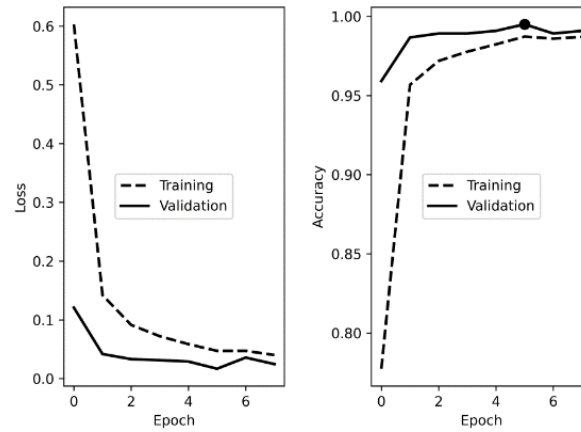


Figure 8. Multi-class training of [घ, ण, प, म, य, ष]

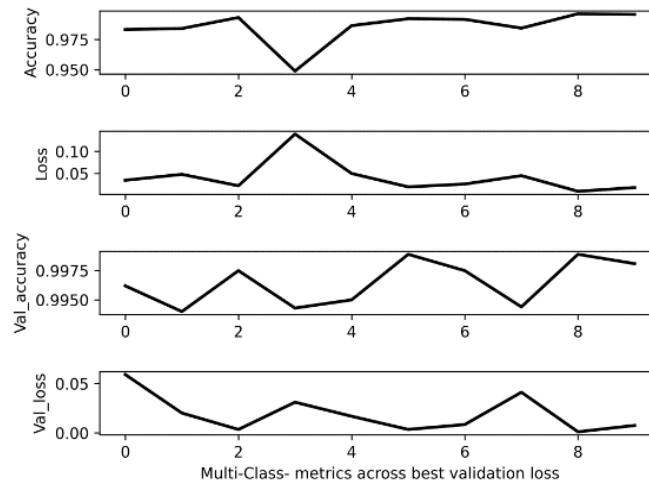


Figure 9. Multi-class training metrics convergence

## 5. RESULTS AND DISCUSSION

For testing, refined subset of Mile Devanagari dataset [25] is used. Text documents are scanned using UMAX Astra 5400, HP Scan-jet 2200c scanners at 300 dpi resolution and stored in 8-bit grey format. The images are extracted from newspapers, magazines, books and laser printed documents. Variations in printing style and sizes was ensured while selecting. About 100 pages are scanned and are segmented into words or phrase images by an automatic process. Refined subset of 10k was used for testing purpose where the words are from 2 to 8 characters in length. The two standard metrics used for assessing the performance of OCR systems are character error rate (CER).

$$(\text{Character Error Rate}) \text{ CER} = \frac{\text{No. of incorrect characters}}{\text{Total no. of characters in text}} * 100\%$$

$$(\text{Word Error Rate}) \text{ WER} = \frac{\text{No. of incorrect words}}{\text{Total no. of words in text}} * 100\%.$$

Accordingly, the results obtained from the experiments are shown in Table 4.

Table 4. Results

Error Rate	Accuracy (%)		
	Google Cloud Vision OCR	Easy OCR	Cross Sect OCR
CER	14.93	25.3	29.23
WER	9.73	20.17	24.47



Error rate observed in our work Cross Sect OCR are due to miss classification of groups of close resembling characters. In groups like [ॐ, ॐ] [ॐ ॐ ॐ:] the dot pixels can be lost during thinning stage of OCR process as on outlier or noise. In (3), new values of variables will then be  $x=11$ ,  $y=32$ ,  $z=3$  and  $s=2000$ .  $F(x, y, z, s)=4.35$ . Hence the dataset complexity reduction factor is 4.35 in place of 7.2. It suggests to explore alternatives to counter the side effects of thinning stage. Error rate is also due to the conjuncts like ॐ where the diacritic ॐ cannot be distinguished from the base consonant ॐ during classification as separate entities. This can be handled in post processing stage of OCR using n-grams in the vocabulary to reduce CER. EasyOCR classification model [26] has 8.3 M parameters. Google Cloud Vision OCR models [27], [28] which is based on vision transformers principle has 15 M - 147 M parameters. The present work model has only 210.53 kB parameters.

## 6. CONCLUSION

We observe that Cross Sect OCR developed in this work lags in terms of accuracy by 4.3%-14.7% compared with STOA solutions. At the same time dataset size used for training is reduced by 4.35 times and CNN model used for classification is much simpler. The ideal dataset complexity reduction for the dataset considered should be 7.2 times. The investigation has thus provided, insight into alternate approaches to tackle machine translation of low resource languages. Feasibility of incorporating the OCR solutions on standalone and off the shelf electronic devices for languages where the large alphabet set can be transformed to simple representation is proved. The efficiency of the proposed OCR solution can be enhanced by further working on thinning process used, by fine tuning the CNN base models, by using effective ways for diacritics separation and with adoption of post processing correction techniques. The lexicon free approach of the proposed OCR is not confined to any vocabulary is an added advantage. The approach can be used for other Indic scripts with diacritic and conjunct ligatures systems.




## REFERENCES

- [1] K. Thammarak, P. Kongkla, Y. Sirisathitkul, and S. Intakosum, "Comparative analysis of Tesseract and Google Cloud Vision for Thai vehicle registration certificate," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 2, pp. 1849–1858, 2022, doi: 10.11591/ijece.v12i2.pp1849-1858.
- [2] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 843–852, 2017, doi: 10.1109/ICCV.2017.97.
- [3] Jaided AI, "EasyOCR," *JAIDED AI*. <https://www.jaided.ai/easyocr> (accessed Dec 18, 2024).
- [4] O. Ignat, J. Maillard, V. Chaudhary, and F. Guzmán, "OCR improves machine translation for low-resource languages," *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 1164–1174, 2022, doi: 10.18653/v1/2022.findings-acl.92.
- [5] A. Fan *et al.*, "Beyond English-Centric Multilingual Machine Translation," *Preprint, arXiv:2010.11125*, 2020.
- [6] G. Wenzek *et al.*, "CCNet: extracting high quality monolingual datasets from web crawl data," *LREC 2020 - 12th International Conference on Language Resources and Evaluation, Conference Proceedings*, 2020, pp. 4003–4012.
- [7] N. Goyal *et al.*, "The FLORES-101 evaluation benchmark for low-resource and multilingual machine translation," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 522–538, 2022, doi: 10.1162/tacl\_a\_00474.
- [8] R. Smith, "An overview of the tesseract OCR engine," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, 2007*, vol. 2, pp. 629–633, doi: 10.1109/ICDAR.2007.4376991.
- [9] C. Wick, C. Reul, and F. Puppe, "Calamari – a high-performance TensorFlow-based deep learning package for optical character recognition," *Digital Humanities Quarterly*, vol. 14, no. 2, 2020.
- [10] S. Rijhwani, A. Anastasopoulos, and G. Neubig, "OCR post correction for endangered language texts," in *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2020, pp. 5931–5942, doi: 10.18653/v1/2020.emnlp-main.478.
- [11] H. Ding, K. Chen, and Q. Huo, "Compressing CNN-DBLSTM models for OCR with teacher-student learning and Tucker decomposition," *Pattern Recognition*, vol. 96, p. 106957, Dec. 2019, doi: 10.1016/j.patcog.2019.07.002.
- [12] S. Gongidi and C. V. Jawahar, "IIIT-INDIC-HW-words: a dataset for Indic handwritten text recognition," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12824 LNCS, 2021, pp. 444–459.
- [13] Google, "Google Cloud Vision documentation," *Google Cloud*. 2024, Accessed: Dec 18, 2024. [Online]. Available: <https://cloud.google.com/vision/docs/>
- [14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.
- [15] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," in *2017 IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017*, 2017, pp. 721–724, doi: 10.1109/ICBDA.2017.8078730.
- [16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, no. PART 1, Springer International Publishing, 2014, pp. 818–833.
- [17] "Papers with code - image classification on ImageNet." 2022, [Online]. Available: <https://paperswithcode.com/sota/image-classification-on-imagenet>. (accessed Dec 18, 2024).
- [18] V. Nair and G. E. Hinton, "Rectified linear units improve Restricted Boltzmann machines," in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010, pp. 807–814.




- [19] N. K. Sinha and M. P. Griscik, "A stochastic approximation method," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-1, no. 4, pp. 338–344, Oct. 1971, doi: 10.1109/TSMC.1971.4308316.
- [20] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *30th International Conference on Machine Learning, ICML 2013*, 2013, no. PART 3, pp. 2176–2184.
- [21] M. Abadi *et al.*, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 2016, pp. 265–283.
- [22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [23] P. Patro, K. Kumar, and G. Suresh Kumar, "Applications of three layer CNN in image processing," *Journal of Advanced Research in Dynamical and Control Systems*, vol. 2018, no. 1 Special Issue, pp. 510–512, 2018.
- [24] F. Lei, X. Liu, Q. Dai, and B. W. K. Ling, "Shallow convolutional neural network for image classification," *SN Applied Sciences*, vol. 2, no. 1, 2020, doi: 10.1007/s42452-019-1903-4.
- [25] MILE-Group, "MILE dataset," MILE-Group 2012, [Online]. Available: <https://sites.google.com/a/milegroup.net/www/home> (accessed Dec 18, 2024).
- [26] B. Shi, X. Bai, and C. Yao, "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*, vol. 39, no. 11, pp. 2298–2304, 2017, doi: 10.1109/TPAMI.2016.2646371.
- [27] Z. Zong *et al.*, "Self-slimmed vision transformer," *Preprint, arXiv:2111.12624*, 2021.
- [28] Z. Zong, "Self-slimmed Vision Transformer (ECCV2022)," *GitHub*, 2022. <https://github.com/Sense-X/SiT> (accessed Dec 18, 2024).

## BIOGRAPHIES OF AUTHORS



**Malathi P.**    is an assistant professor in the Department of Information Science and Engineering at Dr. Ambedkar Institute of Technology. She has more than 28 years of teaching experience and is currently pursuing her doctoral research under Visvesvaraya Technological University (VTU), Belagavi. Her research interests include computer vision and optical character recognition (OCR) for Indic scripts. She can be contacted at email: [malathip.is@drait.edu.in](mailto:malathip.is@drait.edu.in).



**Chandrakanth G. Pujari**    is a professor in the Department of Master of Computer Applications at Dr. Ambedkar Institute of Technology. With over 30 years of teaching experience, his academic and research interests cover software engineering, fuzzy logic, and artificial intelligence. He can be contacted at email: [lavbangalore6@gmail.com](mailto:lavbangalore6@gmail.com).