DevChar: An Extensive Dataset for Optical Character Recognition of Devanagari Characters

Akshara Subramaniasivam, Azhar Shaik, Kaushik Ravichandran, and Manu George

Abstract The advent of cameras has only accelerated the need to digitize content as it helps prevent data corruption by natural processes and enables faster transfer of the data across communities. Handwritten documents and ancient manuscripts form a large part of this data as they call for a need to be translated from the local languages they were written in. The first step into solving this problem is the recognition of handwritten text. Existing handwritten datasets for the Devanagari script can be used for the recognition of individual characters, but they fail to perform well when the text contains matras and conjuncts created by joining character modifiers. This also introduces a dependency between the model and the data source due to required pre-processing for extracting characters recognized by the model from the word itself. These datasets also lack variation in their penmanship which is essential to encompass diversity in the writing style. We present an extensive dataset that addresses these issues. Our dataset has around 4 million characters of varying handwriting styles, complex characters and matras. Training a simple CNN on our data, to detect characters with matras, gave accuracies exceeding 98%. We also show that using this dataset allows a separation of the input data format from the model design, thus allowing researchers to focus on the latter. This dataset is made publicly available at DevChar2020.

Keywords Pattern Recognition (PR) · Optical Character Recognition (OCR) · Handwritten dataset · Devanagari script · Hindi dataset

1 Introduction

Since its inception in 1914, the first optical character recognition (OCR) model [1] was a mechanical statistical machine, and its intricacy has only exponentiated over time and so has its compactness. It is now a prime area of study in conjunction with complex deep learning models, which can easily be deployed on any home computer.

A. Subramaniasivam · A. Shaik · K. Ravichandran (✉) · M. George
PES University, Bengaluru, India

S. Dhar et al. (eds.), *Advances in Communication, Devices and Networking*,
Lecture Notes in Electrical Engineering 776,
https://doi.org/10.1007/978-981-16-2911-2_13

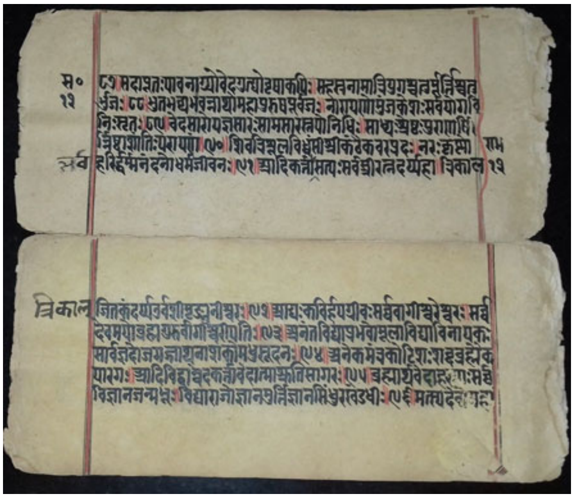


Fig. 1 Ancient manuscripts written in Devanagari script [5]

To simplify this process for the common man, advancements have been made with the release of open-source applications such as the Google Lens. The primary focus of these models is to convert handwritten text into digital documents, the importance of which is maintained by Nagy [2].

The study of OCR for printed text is not much of our concern as there are many novel methods proposed for the same, as mentioned by Impedovo et al. [3]. OCR for printed text is a fairly straightforward problem as the text is well defined in its parameters, such as the font used, which is specified and the size, which is kept uniform. The challenge lies in building a model for OCR for handwritten text. Writing any text by hand introduces a large variation in the text, as there is no clear definition of font, size and other such features. There can be a large variation within the font itself, and a constant size cannot be guaranteed due to anthropogenic factors.

One of the key elements required for the advancement of existing OCR technology is a good dataset. There are many large and extensive datasets for the English language such as NIST [4], but very few exist for Indian languages. It is known that most Indian and a huge number of Southeast Asian languages use the Devanagari script. In addition to this, the script is one of the oldest known scripts, which is being used since the first century CE. Some of the world's oldest manuscripts and scriptures, as shown in Fig. 1, are written in Sanskrit, which also uses the Devanagari script. This calls for a requirement for a large and extensive dataset for this script. The Devanagari script is used by over 120 Indo-Aryan languages and is a logical composition of its constituent symbols in a horizontal fashion from left to right, connected by a horizontal line, also known as the sirorekha. It has 14 vowels and 33 simple consonants. Besides the consonants and vowels, other constituent symbols in Devanagari are the set of vowel modifiers called Matras (placed to the left, right, above, or at the bottom of a character or conjunct) and pure consonants (also called half letters) which when combined with other consonants yield conjuncts. The presence of these vowel modifiers and pure consonants (half letters) often makes character recognition for the Devanagari script quite complex.

Some of the most widely used handwritten datasets in the Devanagari script include a printed and written dataset which was created by Yadav et al. [6]. ISM office fonts were used for the creation of printed characters, and the written dataset was collected from people of various age groups and professions. Holambe et al. [7] prepared a database of Hindi characters, which also follows the Devanagari script, totaling 77 different characters in 5 fonts, giving 385 characters which were used in training. An accuracy of 98.5% was obtained for the character-level recognition. The fallback of this dataset is that it is varied in terms of fonts, but not in terms of handwriting. Acharya et al. [8, 9] introduce a public image dataset for Devanagari script, containing 92 thousand segmented images of handwritten characters, of 46 different classes of which 72 thousand images were of consonants and 20 thousand were numerals. The characters were handwritten and then manually cropped. The processing stages included converting the image to grayscale followed by color inversion to produce white characters on a black background. One contribution of this paper is that the images were pre-processed well, but the dataset still lacks variation and is small in size (Fig. 2).

In this paper, we present an extensive dataset for the characters of the Devanagari script. Our large dataset comprises almost 4 million images with the following novelties:

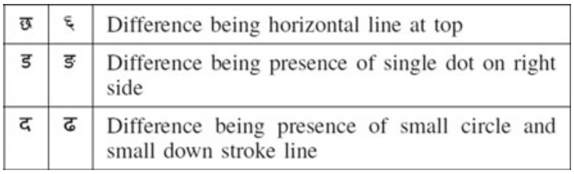
- Our dataset contains 378 different character classes.
- Not only pure but also character-vowel and character-character combinations were considered.
- Our dataset has over five different handwriting styles.
- The dataset is pre-processed and ready to train.
- This dataset allows using simpler models for character recognition, which was not so earlier, as they had to accommodate combined characters within the model.
- The dataset is available in two sizes—DevChar-small and DevChar-large, with the same amount of variance in parameters.

The rest of this paper is structured as follows. The next two sections talk about the shortcomings of existing datasets in the Devanagari script and our contributions to overcome them, respectively. We then talk about the dataset creation in detail followed by the evaluation and results of the dataset.

2 Shortcomings of Existing Datasets

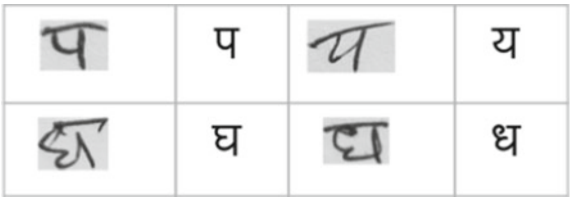
The base form of the consonants can be coalesced with vowels to form additional characters. There are 33 consonants and 14 vowels giving us a total of 462 different characters to detect and identify. So far, most available datasets address only recognition of individual characters, and not conjuncts created by joining two characters

Fig. 2 Problems with OCR for Devanagiri text



छ	६	Difference being horizontal line at top
ड	ङ	Difference being presence of single dot on right side
द	ढ	Difference being presence of small circle and small down stroke line

(a) Structural differences in characters



(b) Difference in writing styles

or a character and a vowel modifier, which is a common occurrence. Creating such an extensive dataset for handwritten text involves a lot of manual work. Neglecting these conjunct characters might seem like a simple solution, but comes with a giant overhead to deal with during the image processing stage, since separating the vowel from the consonant is not simple due to the cursive nature of the written text. This nature of the script makes the segmentation process of the script into its characters for recognition a lot more complex.

Since we are dealing with images of handwritten text collected from various different sources, each image needs to be pre-processed efficiently such that factors such as the ink used while writing, the paper it was written on and the thickness of the pen nib do not become considerably significant. All these factors make it hard for us to separate the foreground and the background in the image. Pre-processing is an often overlooked stage in the creation of a dataset, but one which carries significant weight in deciding the merit of a good dataset.

Segmenting sentences from a paragraph of text is fairly easy as there exists a punctuation mark to signify the end of a sentence, a vertical line. Segmenting the sentence into words is also a simple task as the characters combined to form a single word are all joined together by a continuous horizontal line connecting the tops of every character. Segmenting the words into individual characters is a complex undertaking. Existing methods identify the characters joined by the horizontal line and then remove the line to try and segment the characters, which gives rise to another issue where the few structural differences that distinguish the characters are lost.

3 Our Contributions

Ancient scripts have an irregular nature which is also an inherent characteristic of characters in the Devanagari script. Regardless of the language, segmenting a word using the horizontal line connecting the literals is not accurate since it is seldom present in ancient scripts and carvings. Due to the age of the Devanagari carvings and manuscripts, the characters, words and lines are ill-formed and require in-depth analysis and segmentation at the character level. By creating a huge variety of combinations of characters and vowels, rather than just the characters alone, we address the need to further segment characters into consonants and vowels for identification, which would otherwise be required.

The performance of a machine learning model depends heavily on the quality of the training dataset. The performance is the best when the dataset is diverse enough to truly represent the target concept. We have created a varied dataset representing the many circumstances of alignment, spacing, slope and shape of characters. There are many acceptable ways of writing the same character which may depend on the region, the font, the era of the script and most importantly, the author. A combination of all these varied factors gives rise to a heavily varied target distribution. The dataset was created by collecting the manually written and annotated characters from various people to encompass diversity in the writing style. In addition to this, we ensure that a large variation is captured by artificially augmenting the images by using a tool that allows us to perform elastic distortions.

One of the key stages during the development of OCR techniques is the pre-processing of images before training the model. Every image in our dataset is subjected to various phases of pre-processing in order to optimize and normalize each image. This saves time and resources as researchers can now focus on the architecture of the model rather than the uniformity of the input. The stages and details of the pre-processing steps are mentioned further in Sect. 4.2 of the paper.

4 Dataset Creation

The Devanagari script is composed of 47 primary characters, including 33 consonants and 14 vowels. The vowels are mostly used along with characters, and sometimes standalone, but never combined with each other. Compound characters can also be used, depending on the rules of the language. The dataset was created with the main focus being combinations of consonants and vowels. Since there are over 120 languages using this script, making a dataset with all combinations of primary characters is not easily feasible and hence has been avoided. The digits in Devanagari script have also been avoided as they are hardly seen in texts. This dataset includes 378 classes of 10000 images each, with the following classes being covered:

- Individual vowels;
- Individual characters;

- Characters combined with vowels;
- Characters combined with other characters.

This section lists the steps involved in creating the dataset. To give a thorough insight into the dataset creation, we have divided it into four subsections. The first subsection briefly goes over the collection of the images, followed by a subsection on the pre-processing involved to make the data simpler to train models. The following subsection covers the steps used to augment the dataset to increase its size. We finally conclude this section by presenting the metrics of our dataset compactly.

4.1 Data Collection

This dataset was collaborated by five people, to include a large variance in the handwriting. Each individual contributed to writing about 63 classes. This gave an equal distribution for each handwriting, thus increasing the variance in the dataset. We started with one class in one data sheet, which is a plain A4 size sheet of paper. Each data sheet was divided into 8 columns with 11 rows each, thus giving 88 cells per class, where each cell corresponds to one image, as shown in Fig. 3. This whole sheet was scanned at 300 dpi and then cut into 88 images, programmatically. We then organized this dataset into one folder per class, which makes it easier for generating more data, and training the model. The next few sections cover the pre-processing done and the steps involved in augmenting the images to generate a larger quantity of images.

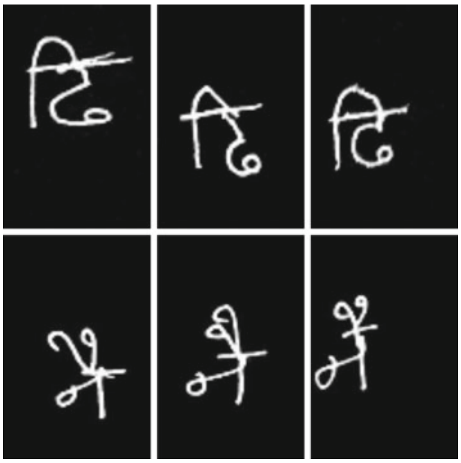


Fig. 3 Dataset images

4.2 Pre-Processing

Working with raw images adversely affects the training process as the foreground character blends into the background burdening the network to learn to differentiate between foreground and background first. To mitigate this undesirable overload, we first obtain the grayscale of the image and then perform image inversion. This results in an image where the white foreground character is placed upon a black background marking a clear distinction between the foreground and background aiding in the character recognition process. In addition to this, each character image is resized to 100 × 100 resolution as the model takes this resolution as input. We restrict the resolution of the input images to ensure that the training of the model happens in a reasonable time.

4.3 Data Generation

Once we obtain the dataset and perform the necessary pre-processing, we use Augmentor [10] to generate more data. Augmentor is a software package used for image augmentation, which provides support for many commonly used operations in image augmentation for machine learning problems. After loading the image, we add each operation we would like to perform on the image into a pipeline, where the operations are executed sequentially. Once all the operations are loaded in the pipeline, we sample the images to generate the specified number of output images. The output images may or may not be augmented, based on the probability.

We applied a random distortion to all our images, which means that any elastic distortion is done, with a specific probability. Two other parameters we provided were grid size and magnitude. The grid size controlled how fine and granulated the distortion will be, whereas the magnitude controlled the size of the distortion. An elastic distortion means that the aspect ratio of the image is maintained, but the horizontal or vertical lines in the images are distorted from their original position. Figure 4 shows augmented images for different distortion parameters for each image. Augmented images along with the original are shown in Fig. 4. The simple algorithm for augmenting is given in Algorithm 1.

Algorithm 1: Augmentor

```
Result: Set of Augmented Images
1 for classes i=0 to n do
2     initialize pipeline;
3     set random distortions;
4     sample images for class i;
5 end
```

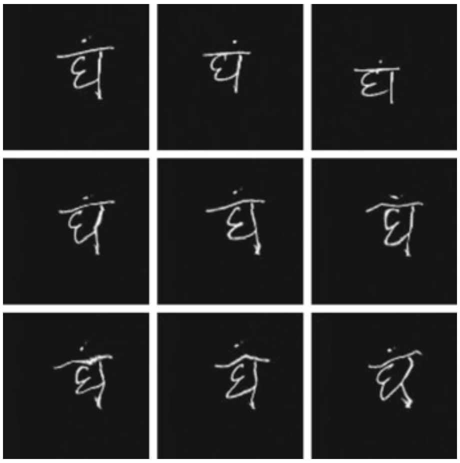


Fig. 4 Augmented Images Row 1: Original Images; Row 2: Grid size 6 × 6 and magnitude 3; Row 3: Grid size 8 × 8 and magnitude 4

Table 1 Dataset metrics

Characters	Images per class	Number of classes	DevChar-small	DevChar-large
Consonants	88	33	14850	330000
Vowels	88	14	6300	140000
Consonants + Vowels	88	329	148050	3290000
Consonants + Consonants	88	2	900	20000
Total	–	378	170100	3780000

4.4 Dataset Metrics

The different classes in the dataset have been covered in the first subsection of this section. The list of vowels, characters and combinations in the dataset are listed in Table 1. Each class has 450 images in the small dataset, and 10000 images in the large dataset.

5 Evaluation and Results

The goal of our evaluation is twofold. We first evaluate the performance of our dataset when trained on a simple CNN. We also compare this with the performance of the same model on the dataset obtained from the UCI Machine Learning Repository [8, 9].

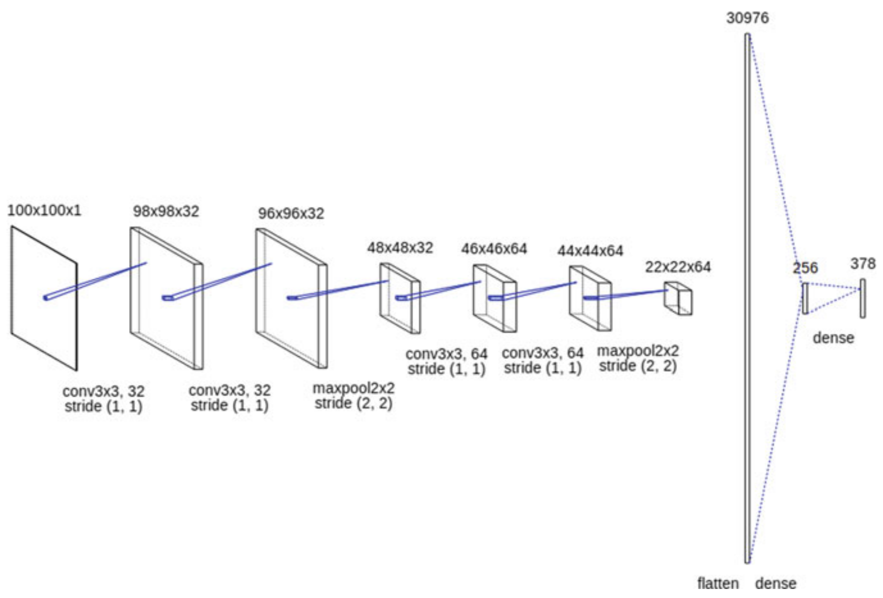


Fig. 5 CNN model used

The first subsection discusses the model we used in brief along with a presentation of the results obtained. In the second subsection, we discuss how this dataset makes building a model easier, in comparison with previously used models.

$$p = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{1}$$

$$r = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{2}$$

$$F_1 = 2 * \frac{p * r}{p + r} \tag{3}$$

The metrics [11] that were used are given below. Precision is the probability of false predictions actually being false, as given in Eq. (1). Recall is the probability of false being predicted false, and is calculated according to the formula in Eq. (2). F_1 is the harmonic mean of Precision and Recall, which is given by Eq. (3). We also report the accuracy of the model.

Table 2 Evaluation results

Dataset	Precision	Recall	F1	Accuracy
UCI	0.992	0.992	0.992	99.23
DevChar-small	0.929	0.949	0.939	92.19
DevChar-large	0.994	0.992	0.993	99.25

5.1 Model

The architecture of our Convolutional Neural Network is shown in Fig. 5 and is similar to VGG16 [12]. Instead of having 16 blocks of 2 Convolutional and 1 max-pooling layer, we have two such blocks followed by 2 dense layers. The final layer corresponds to the number of output classes which in our case is 378. We also use a dropout with p = 0.25 and p = 0.5 for the last layer. All activations are ReLU except for the last layer which uses the softmax activation function to give a probability distribution over the classes. We ran the model for 40 epochs. The batch size was kept constant at 64 for training and testing. The training was done at 1200 steps per epoch, and the testing was done at 215 steps per epoch. We ran our model on our datasets for 6 epochs, and used a batch size of 256, with 10000 steps per epoch for the first three epochs, and 2000 steps per epoch for the last three epochs. The evaluation was done on an Apple MacBook Pro 2017 Intel(R) Core(TM) i5-7360U CPU @ 2.30GHz, 8 GB RAM using Python v3.7.3. Our model, obtained an accuracy of 99.23%, on the UCI Machine Learning Repository Dataset and an accuracy of 99.25% on our large dataset, as shown in Table 2.

5.2 Inference of Results

Recognition of characters with matras and compound characters formed by the combination of two characters is a challenging task. Recognition of these characters often requires complex models to isolate the base character from its complex form and then use models like a classifier or a finite state automaton to detect the character. For example, [13] uses Zernike moment feature descriptor and proposes SVM and k-NN-based classification systems for the extraction of features from handwritten compound characters. Using our dataset, the same can be detected with a much simpler CNN model as mentioned in the previous subsection as the dataset itself contains characters along with their matras and compound forms. This helps us avoid the process of segmentation to isolate the base form of the character. Furthermore, simplifying the model also makes the process of optical character recognition much faster and also provides us with accuracies exceeding 98%. Hence, this dataset helps make character detection faster, simpler and more accurate.

6 Conclusion and Future Work

In this paper, we present a new dataset which is made publicly available for any research scholar to build models for Optical Character Recognition of characters of the Devanagari script. The experimental analysis illustrates that the dataset is robust and can be applied to solve the difficulty in character recognition of all Devanagari script-based languages. The characters were handwritten, manually cropped and scanned. Augmentation of the characters provided variations that would have otherwise required more time-consuming manual labor in writing and annotating. The addition of conjunct characters, i.e. consonants with all relevant vowels makes the dataset more detailed, extensive and allows us to achieve thorough and accurate results.

The dataset we created is extensive in that it covers most conjunct characters, but a lot of the images were augmented. Each class can be enlarged by writing more samples per class by hand, rather than augmenting. We have focused mostly on vowel-consonant combinations. There are many more allowed combinations of consonants with other consonants, depending on the language and its dialect. The dataset could also be enlarged in this direction as well. The network used to test robustness only provided a first look into what could be used for the process of recognition. A much more detailed study can be done to build our own network. We only show how character recognition is done, and avoid word-level and sentence-level recognition, which is where our dataset would perform the best. These two additional steps are essential to digitize a document, which is the end goal of our study. There are also efforts being made by the authors to donate the dataset to be a part of the UCI Machine Learning Repository [9].

References

1. Tauschek G (1929) Reading machine, United States Patent 2026329, May 27, 1929
2. Nagy G, 29 optical character recognition–Theory and practice. Classification pattern recognition and reduction of dimensionality, 621–649
3. Impedovo S, Ottaviano L, Occhinegro S, Optical character recognition: a survey. Int J Pattern Recogn Artif Intell 05(1–2):1–24
4. Grother PJ (1995) NIST special database 19 handprinted forms and characters database. National Institute of Standards and Technology
5. Research Group on Manuscript Evidence, Hindu Manuscript on Paper http://manuscriptevidence.org/wpme/sanskrit-and-prakrit-manuscripts/
6. Yadav D, Sánchez-Cuadrado S, Morato J (2013) Optical character recognition for hindi language using a neural-network approach. J Inf Process Syst 9(1)
7. Holambe AN, Thool RC, Jagade SM (2010) Printed and handwritten character & number recognition of Devanagari script using gradient features. Int J Comput Appl (0975- 8887) 2(9)
8. Acharya S, Pant AK, Gyawali PK (2015) "Deep Learning based large scale handwritten Devanagari character recognition. 9th International conference on software. Knowledge, Information Management and Applications (SKIMA)
9. Dua D, Graff C, UCI machine learning repository. http://archive.ics.uci.edu/ml

10. Bloice MD, Roth PM, Holzinger A (2019) Biomedical image augmentation using Augmentor. Bioinformatics 35(21):4522–4524
11. Chinchor N (1992) MUC-4 evaluation metrics. In: Proceedings of the fourth message understanding conference (MUC-4)
12. Simonyan K, Zisserman A, Very deep convolutional networks for large-scale image recognition. International conference on learning representations (ICLR)
13. Kale KV (2014) Zernike moment feature extraction for handwritten Devanagari (Marathi) compound character recognition