Wireless Personal Communications (2022) 122:349–378

https://doi.org/10.1007/s11277-021-08903-4

**Handwritten Devanagari Character Recognition Using Modifed Lenet and Alexnet Convolution Neural Networks**

**Duddela Sai Prashanth1,2 · R. Vasanth Kumar Mehta1 · Kadiyala Ramana3 · Vidhyacharan Bhaskar4**

Accepted: 8 August 2021 / Published online: 24 September 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

**Abstract**

Despite many advances, Handwritten Devanagari Character Recognition (HDCR) remains unsolved due to the presence of complex characters. For HDCR, the traditional feature extraction and classifcation techniques are limited to the datasets developed in the respec tive laboratory that are not available publicly. A standard benchmarking dataset is not avail able for HDCR that helps to develop deep learning models. To progress the performance of HDCR, in this study, we produced a dataset of 38,750 images of Devanagari numerals, and vowels are generated and made publicly available for fellow researchers in this domain. This data is collected from more than 3000 subjects of diferent age groups. Each character is extracted by a segmentation technique proposed here, which is limited to this applica tion. Experiments are conducted on the dataset; three diferent Convolution Neural Net works (CNN) architecture is developed. 1. CNN, 2. Modifed Lenet CNN (MLCNN) and 3. Alexnet CNN (ACNN). A Modifed LCNN is proposed by changing the architecture of Lenet 5 CNN. Regular Lenet 5 has tanh(*x*) as its activation function. Since the Devangari characters are nonlinear, non-linearity is introduced in the Networks by using Rectifed Lin ear Unit. This solves the problem of vanishing gradient problem by tanh(*x*). We achieved a recognition rate of 96% on training data and 94% on unseen data using CNN. MLCNN obtained an accuracy rate of 99% and 94% with less computational cost. Whereas, ACNN attained a recognition rate of 99% and 98% on unseen data. A series of experiments were conducted on the data with diferent combination splits of data and found a minimum loss of 0.001%. Such developments fll a signifcant percentage of the huge gap between real world requirements and the actual performance of Devanagari recognizers.

**Keywords** Devanagari character recognition · Convolution neural network · Computer vision · Pattern recognition

🖂 Vidhyacharan Bhaskar

vcharan@gmail.com

Extended author information available on the last page of the article

1 3

Vol.:(0123456789)

350 D. S. Prashanth et al.

**1 Introduction**

In the digital world, the demand for accessing multimedia is emerging. It involves a pro cess of adapting the ofine entities into the digital domain. The analysis of the scanned documents deals with the images. In real-time, character recognition on those images is a complex process. The automation of character segmentation and recognition of characters in real-time is a complex process—it includes character segmentation and recognition of characters with the features extracted [1]. Optical Character Recognition (OCR) is catego rized into two methods, i.e., online and ofine.

Character recognition is easier for the documents with Latin script with the form of the English language. A lot of research is carried for handwritten character recognition in various scenarios like English, Chinese, and Arabic scripts [2]. In any case, for Indian Languages, OCR framework work is as yet slacking when compared with others due to its complicated structure and computations. Devanagari script is base for many languages like Hindi, Marathi, Sanskrit, and many more [3]. It contains 10 numerals, 13 vowels, and 33 consonants. In this research, majorly concentrate on the recognition of ofine handwritten Devanagari numeral and vowels.

Ofine handwritten Devanagari character recognition (HDCR) is an emerging research feld over decades because of the challenges of variant character classes. Despite many advances, HDCR remains unsolved due to the presence of complex characters. Most of the traditional recognizers have not to lead better performance, or it is limited to their data

set, because of not having big datasets as a benchmark. The variety of handwriting styles makes it more difcult due to the similarities between the characters [4]. The progress in this research is improved recognition accuracies with the help of traditional approaches. Though, the number of studies increased—crafted features and robust classifers still rec

ognition accuracy is far behind the human ability. The reason is the lack of benchmark datasets [5]. Many researchers have developed their dataset in their laboratory and clas sifers developed, accuracies are limited to the respective dataset, and most of the data sets are not available publicly [6]. Implementing HDCR systems is much difcult than printed characters, especially for the Indian scripts [7]. The algorithms used for the printed Devanagari characters can often be used on the handwritten characters. Most of the time, it fails to work on the handwritten characters because of the variation in writing style and sizes of diferent writers.

Most of the OCR techniques developed on HDCR includes a time-consuming pre-pro cessing step. After noise removal and character segmentation, a process that extracts the header line and separates the upper part and lower part of the line for every character. Clas sifers are built on the basis of features extracted from these two parts of the character.

The primary objective of this study is to develop a CNN-based model with a lower com putational complexity and memory space while maintaining the required accuracy. As a result, the paper is divided into two parts: (a) We investigate and compare/analyze several diferent state-of-the-art CNN models; and (b) We develop a new CNN model, which we refer to as the "Modifed LeNet Convolutional Neural Network (MLCNN)," that requires less time and memory space than existing CNN models. On some publicly available data sets, “MLCNN” is compared to other popular CNN models for validation purposes.

In this paper, a new dataset is presented for Devanagari numerals and vowels. A total of 38,750 images are collected from 2400 subjects of diferent age groups. This dataset is publicly available [8]. A segmentation algorithm is proposed for the dataset collected, and Convolution Neural Network (CNN) architectures are proposed for the HDCR. First, CNN

1 3

Handwritten Devanagari Character Recognition Using Modifed… 351

obtained 96% accuracy and 94% for unseen data. Secondly, LeNet CNN architecture is modifed by introducing max-pooling instead of average pooling to reduce the parameters of fully connected layers then added non-linearity by using Rectifed Linear Unit. Modi fed LeNet Convolution Neural Network (MLCNN) obtained an accuracy of 99% accuracy on Training data and 94% on unseen data. Thirdly, AlexNet CNN (ACNN) architecture achieved an accuracy of 99% and 98% on the unseen dataset and made a detailed compari son between these architectures by separating the data into diferent ratios of training and test (i.e., 80:20, 70:30, 60:40) with varying epochs of count (i.e., 20, 30, 40 and 50 epochs).

The rest of this paper is organized as follows. Section 2 reviews the related works about Ofine HDCR. Section  3 provides the method: the basic theory of CNN, ACNN, and MLCNN and details about the proposed architecture. Section  4 presents the experiment results and discussion that includes the experimental data, settings for the experiment, and training strategy along with the comparison of the results. Finally, the conclusion is given in Sect. 5.

**2 Related Work**

Ofine Handwritten Devanagari Character Recognition (HDCR) has received intense attention from the past few decades. The traditional OCR method includes three stages: pre-processing, feature extraction, and classifcation. Pre-processing consists of the steps of size normalization and resampling [6]. The variations in the input may adversely impact recognition. Features characterize the best representation of the shape of a character.

Character segmentation is classifed into word, line, and character segmentation [1, 42]. It is more accessible to segment printed characters when compared to handwritten char acters. There are many standard segmentation techniques developed for printed text. For HDCR, segmentation-based classifers are popular before pattern-based neural implemen tation. Diferent approaches of segmentation as per the dataset is implemented.

Feature extraction techniques developed are of shape-based or non-shape based [10, 41]. These techniques generate a feature vector. These vectors are used as input for any classifers, unlike deep learning methods, feed a pre-processed image as input along with its label for supervised learning. Hybrid features—a combination of shape-based and non

shape-based features classify the input image more accurately. Still, the automatic features generated by CNN’s are better.

In the recent review article [11], structured based and statistical features and their accu racies are compared. Chain code and gradient features achieved an accuracy of 98.51% by the SVM classifer for the printed Devanagari characters [12]. Few Structural features with diferent classifers like syntactic pattern analysis-based classifer [13] (90% accuracy), Binary tree classifer [14] (95% accuracy), Distance-based classifer [15] (93% accuracy) for printed text. Diferent feature extraction techniques implemented for HDCR, quadratic based classifer on the histogram of directional chain code features [16] achieved 98.86% accuracy with the dataset size of 11,270 handwritten Devanagari Characters. The most commonly used classifers are MLP based classifers [17–19], SVM [18, 20, 21] Neural Networks [22, 23] or Modifed Neural Networks for feature-based classifcation.

There is no standard dataset that is implemented nor used for Devanagari characters, unlike MNIST for Latin numbers. The dataset size used to develop feature-based classifers vary. Few datasets developed are 11,270 [16] and achieved an accuracy of 98.86%, 25,000 [18] characters with an accuracy of 90.116% by SVM and 87.56% by MLP classifer.

1 3

352 D. S. Prashanth et al.

Zernike Moment based feature [21] on 27,000 handwritten characters with an accuracy of 98.37% by SVM.

Deep learning models has increased its acceptance in the feld of Computer Vision and Pattern Recognition due to CNN [24]. CNN includes deep layers of convolutions, subsam pling layers, feature extraction over pooling layers, stride, and padding operations to con trol the size of image representation. It also comprises diferent activation functions like sigmoid, softmax, to introduce non-linearity—Rectifed Linear units, and more. To avoid overftting, many regularization techniques are included in CNN’s. The classifcation meth ods are categorized into Kernal based, statistical based and Neural network based [25]. In [37], diferent language scripts numerals data is used and classifed using CNN and able to achieve 99% accuracy. The numerals dataset used in [37] is Arabic (MADBase), MNIST, and Persian. In [38], a comparison of CNN-based classifcation and feature-based classi fcation is presented. The maximum accuracy able to reach by HoG based features and MLP classifer is 82.66% where as using CNN, 94.91% accuracy, and 96.09% accuracy by Bidirectional Long Short Term Memory (BLSTM). In [39, 40], CNN with linear func tion—tanh(*x*), accuracy achieved is 93.8% and 85.1%. When nonlinearity is introduced to CNN 96.9% accuracy is achieved.

This brief inre survey reveals that there is a lack of standard datasets for Devanagari Scripts. With larger datasets, CNN models can be designed with high accuracies than the classifcation techniques. Hence, in the present work, we present a dataset and developed a unique segmentation algorithm that is limited to this collected data. Three types of CNN models are developed and compared, CNN, ACNN, and MLCNN.

**3 Proposed System**

**3.1 Preprocessing**

The traditional method of recognizing individual characters consists of three steps, pre processing, feature extraction, and classifcation. In deep learning models, features are extracted by the Neural model automatically [43]. There is no exclusive process involved, but a few techniques are required in pre-processing. The below-mentioned steps are imple mented on the data collected by diferent subjects, as shown in Figs. 1, and 2. that extracts the images into a folder. Those images are manually identifed and separated into respec tive folders for classifcation.

1 3

Handwritten Devanagari Character Recognition Using Modifed… 353 

**Fig. 1** Sample sheet used to collect Devanagari numeral data

1 3

354 D. S. Prashanth et al. 

**Fig. 2** Sample sheet used to collect Devanagari vowel data

1 3

Handwritten Devanagari Character Recognition Using Modifed… 355

The size of each original character varies from one image to another. Since the bounding box is plotted on the recognized character and crops it by the original coordinates. Each subject writes in their own ways leads to the variation of the size of characters on the paper. All the characters extracted are resized to 28×28. Following the standards of one of the most standard and benchmark datasets is MNIST for numerals, the size of the character is chosen as 28×28.

1 3

356 D. S. Prashanth et al.

**3.2 Deep Learning Models**

From the past few decades, research in Deep Learning models (DLM) has delivered remark able outcomes in the feld of computer vision, machine learning, and pattern recognition. DLM’s are more efcient because of their deep architecture. Unlike any conventional classif ers, DLM’s accept raw or pre-processed images as inputs. These models do not entail any par ticular feature extraction. The deep layers, extract features from the input at the lower and mid dle levels and high-level layers perform classifcation. These models integrate everything into a single network that can be optimized as per the objective function. These type of integrated models results better than traditional classifers. The following concepts are being utilized in this research to build such models.

**3.3 Convolutions**

Convolutions were employed for automatic feature extraction [26]. For an image *I* in size of (*p*, *q*), the convolution is defned as

(3) *C*(*p*, *q*) = (*I* ∗ *K*)(*p*, *q*) = ∑*k*∑*lI*(*p* − *m*, *q* − *n*)*K*(*m*, *n*)

where *K* is the kernel is the size of (*m*, *n*). Convolution is a mathematical function to describe the process of combining two functions to produce a third function. The output is called a feature map, and by using a flter or kernel to the input image, extract features for training.

**3.4 Dropouts and L2 Regularization**

Devanagari vowels and numerals are highly sensitive and complex. Characters in Table 5 out of 13 diferent vowels six vowels are identical with minor changes. Similarly, numerals in Table 6 out of 10 numerals, two couple of numerals, and three numerals are identical with changes. This Features extracted vary with small diferences. It leads to over tune of weights to our dataset used for training that may lead to the problem of overftting in the data, and classifer performs poorly on the unseen data. Regularization methods reduce overftting [27]. Diferent regularization methods are L1 & L2 regularization, cross-validation, early stopping, drop out, dataset augmentation. In this research, dropout is adopted and well performed on the data.

Dropout refers to dropping nodes to reduce overftting. This regularization ignores some neurons in the network during forwarding and backward propagation. This approach helps in reducing interdependent learning among the neurons. CNN extracts the most robust features.

In dropout a parameter ‘*p*’ that sets the probability of which nodes are kept or (1 − *p*) for those that are dropped.

(

*t* −∑*n*

)2

+∑*n*

*pi*(1 − *pi*)*w*2*i I*2*i*

*E* = (4) 12 *piwiIi*

*i*=1

*i*=1

L2 regularization is also called Ridge regression. It helps to penalize weights. Any larger weights are gradients with abrupt changes in the model are manifest with a decision boundary by penalizing them it becomes smaller. *𝜆* controls the degree of penality.

1 3

Handwritten Devanagari Character Recognition Using Modifed… 357

*E* = (5) 12(targetO1 − outO1)2 +*𝜆*2*w*2*i*

**3.5 Subsampling**

Subsampling reduces the parameters or the dimensions of the feature map. It helps to dis card the spatial information. For input to the further layers in CNN, the parameters are reduced by taking a collection of neurons. In this research, max pooling is used by taking the whole image and is partitioned into rectangles. From each box, a maximum value is taken and forms a new box [28]. Only one value is considered from a group of values. It helps to avoid further overftting of data.

(

max 0≤*u*,*v*≤*Mn*−1*Xn*−1

)

*X* (6) *n*(*p*,*i*,*j*) = *f*

(*p*,*iSn*+*u*,*jSn*+*v*)

**3.6 Activation Functions**

The role of activation functions is crucial in CNN to segregate the useful and not much helpful information from the neurons. Non-linearity is introduced into the network using activation functions. Over the choices of activation functions like logistic function, tanh(*x*) function, arctan function, and more, these functions lead to vanishing the gradients [4]. Since the input to the neurons is a range of 0, whereas the gradients are signifcant values. Rectifed Linear Unit (ReLU) is used to overcome this problem.

*ReLU*(*x*) = max(*x*, 0) (7)

**4 Optimizer**

**4.1 Stochastic Gradient Descent (SGD)**

SGD optimizers update the parameter for every training example and its label [30]. Lets *xi* is the input and its labels*yi*. *J*(*𝜃*;*xi*;*yi*) in the equation represents the objective function. Gradient Descent is a way to minimize the cost. *J*(*𝜃*;*xi*;*yi*) is parameterized by the model’s parameters *𝜃* ∈ ℝd. *𝜃* represents the weights, that gets updated for every iteration. *𝜂* repre

sents the learning rate that refects the size of the steps that take to reach a local minimum. *𝜃* = *𝜃* − *𝜂* (8) .∇*𝜃J*(*𝜃*;*xi*;*yi*)

Batch Gradient Descent (BGD) performs excess calculations for massive datasets, as it recomputes gradients for similar examples before every parameter gets updated. SGD gets rid of this repetition by performing each update at a time. It is in this manner typically a lot quicker and can likewise be utilized to learn online.

SGD performs updates frequently with a large diference that causes the target function to change vigorously as in image. SGD enables a better local minimum than BGD.

1 3

358 D. S. Prashanth et al.

**4.1.1 Adadelta**

Adagrad algorithm adapts the lower learning rate of the parameters with lower learning rates where features occur frequently and the higher learning rate for the parameters with infrequent features. Adelta is an extension of Adagrad that optimizes the learning rate. Rather than aggregating all past squared gradients, it limits the window of col

lected past gradients to some fxed size *w*

*E* (9) [*g*2]*t* = *𝛾E*[*g*2]*t*−1 + (1 − *𝛾*)*g*2*t*

where *E*[*g*2]*t* is moving average, *t* is the time step, and *𝛾* is the momentum. Here, *𝛾* value is set to 0.9.

**4.2 Fully Connected Layers**

All the nodes in one layer are connected to the output of the other layers is the Fully Connected (FC) layer. The FC layer outputs the class probabilities, where each class is assigned a probability number. All probabilities must sum to 1. Softmax functions produce the probability values for all the categories. All the outputs of a fully connected layer are probabilities. Applying softmax squashes the real value numbers into prob

abilities that sum to 1. Softmax comprises the output range of (0,1) (10) *softmax*(*x*)*i* = exp(*xi*)

~~∑~~*~~n~~*

*j*=1 exp(*xj*)

**4.3 Need of ReLU (Non‑linearity) and Modifcation in LeNet 5**

Non-linear functions are required for universal approximation. Adding more layers into the neural network doesn’t increase the approximation power. Nonlinearity in the deci sion function increases with the ReLU activation function. As per the universal approxi mation algorithm, using only linear activation function in the Neural Networks is less valid. When more layers are added into the network, ReLU is faster to train than CNN with tanh(*x*). The reason to make the network train slower is—Gradient Descent based optimizers are used for weight optimization and to achieve the local minima. Use of tanh(*x*) may lead to vanish the Gradient Descent of the network. Whereas in the tanh(*x*), x is the independent variable and it has the problem of reaching ±∞ and tanh(*x*) deriv ative goes to 0. The gradients become smaller and smaller which means the network cannot be trained anymore, whereas in ReLU, the derivative is a constant if x>0. The relationship between the Network and input is non-linear with the use of non-linear activation function, which makes trained networks take complex decisions. In a regular LeNet 5 the activation function used is tanh(*x*) which is linear in nature, where a modi fcation in LeNet 5 by updating Linear function to ReLU a non-linear function gives a better classifcation accuracy and achieved max local minima (discussed in the results session). in the below Table 6.

1 3

Handwritten Devanagari Character Recognition Using Modifed… 359

**5 Experiment Results and Discussion**

**5.1 Experimental Data**

**5.1.1 Dataset**

The data contains handwritten samples of Devanagari numerals and vowels (i.e., 10 numer als and 13 consonants). Thus, the dataset includes a total of 23 diferent Devanagari char acters, as shown in Tables 1 and 2. The data is collected on a regular A4 sheet which was distributed to the subjects. The subjects wrote numerals and vowels in Devenagari Scripts. Targeted subjects are from schools and colleges. These sheets were scanned at 300 dpi using Epson DS – 150 is shown in Figs. 1, and 2. The numerals and vowels were collected

from 2400 and 1400 subjects of diferent age groups (Figs. 3, 4, Tables 3, 4). Characters are extracted from the scanned images using the segmentation algorithm mentioned in Sect. 2. Further, segmented data is pre-processed. This characters are manu ally segregated since noise in the scanned images are also obtained as characters. All the images were resized to 28×28 pixels, verifed manually and converted into black and white. Where background was converted to black and character was converted to white, as shown in Tables 1 and 2. These are stored in a publicly accessible location. By removing the occluded images and scribbles. the fnal data set contains a total of 38,750 digitized images where 22,500 Devanagari Numerals (2250 each) and 16,250 Vowels (1250 each). Extracted images of Devanagari characters were arranged into diferent folders—a total of 23 folders where each folder for each character (Figs. 5, 6).

**Table 1** Devanagari Numerals Representation

0123456789 10 12345678 9

**Table 2** Devanagari Vowels Representation

a aa e ee u uu ru ye i o au am aha 11 12 13 14 15 16 17 18 19 20 21 22 23

**Table 3** Devanagari Characters that looks similar

Vowels

(pronunciation) a aa u uu o au am aha

Vowels Sample 

Image

Class Number 11 12 15 16 20 21 22 23

1 3

360 D. S. Prashanth et al.

**Table 4** Devanagari Numerals that looks similar

Numerals 123567 8

Numerals Sample 

Image

Class Number 123567 8

The success rate of research on the recognition of handwritten English characters is high when compared to the Indian script like Devanagari. The state-of-the-art techniques in deep learning are efcient in automatic recognition of Devanagari handwritten charac ters, but this requires large samples of data with labels. This data will fll that data-gap for Devanagari numerals and vowels. This data is publicly available at https://data.mendeley. com/datasets/pxrnvp4yy8/1 (Figs. 7, 8).

In this research, CNN, MLCNN, and ACNN is used for implementing HDCR. Dataset used in this work is collected from 3,800 subjects of diferent ages from schools, colleges, and other professions. The number of images fltered after pre-processing is 38,750. The data is split into diferent ratios of 80:20, 70:30, and 60:40 for training and testing, as men

tioned in Table 10 (Figs. 9, 10).

**5.2 Experimental Settings**

In this research, the input image characters are resized to 28×28, following the standards of the MNIST dataset. If the size of the character has increased, the accuracy may also increase, which is directly proportional to the computational cost. So, to avoid more com putational cost, the size is fxed to 28×28. For training the network, the batch size is set to 32. Both the numerals and alphabets are trained separately as per Table 8.

**5.2.1 For CNN**

Along with the above settings, SGD is used to train the model at the learning rate of 0.01. In the frst layer, the network is initialized with 36 kernels and increased to 64 kernels. Dropouts are added to avoid overftting with the ratio of 25%, and 50% of neurons are dropped before the fnal layer. The architecture of CNN is as mentioned in the Table 5.

**5.2.2 For MLCNN**

Convolution kernels used in the MLCNN network is 6 and 16 in diferent layers, which are very useful. Strides are used with values 1 and 2 to control the size of the convolution layer in subsampling. The padding of [1 1 1 1] is implemented in the second and fourth layers. In this NN, SGD is used with a learning rate of 0.01. It is prepared to update the value of the learning rate if the error rate is compromised. The architecture of MLCNN is as mentioned in the Table 6.

1 3

Handwritten Devanagari Character Recognition Using Modifed… 361

CNN Architecture **Table 5**

Activation function No. of params

Param

Image input Layer (type)

ReLU

320 32 kernel 3×3 and input shape

Convolution ‘conv1’

1

28×28

–

–

2×2 Avg Pooling, input shape

Max pooling ‘pool1’

2

26×26×32

ReLU

18,496 64 kernel 3×3 and input shape

Convolution ‘conv2’

2

13×13×32

–

–

2×2 Avg Pooling, input shape

Max pooling ‘pool1’

3

11×11×64

–

–

1600, input shape 5×5×64

Flatten fatten

4

ReLU 1,179,776 Fully connected with dense 128

Fully connected ‘fc1’

5

–

–

50% dropout

Dropout ‘drop2’

6

Softmax

1290 Fully connected with dense 10

Fully connected ‘fc2’

7

Non-trainable

Trainable params: 1,199,882 Total param:

params: 01,199,882

1 3

362 D. S. Prashanth et al.

Modifed Lenet CNN Architecture **Table 6**

Activation function No. of params

Param

Image input Layer (type)

ReLU

156 6 kernel 5×5 and input shape

Convolution ‘conv1’

1

28×28

2×2 Avg Pooling, padding [1 1 1 1],

Avg pooling ‘pool1’

2

Stride 2, input shape 27×27×6

ReLU

2416 16 kernel 5×5, Stride 2, input shape

Convolution ‘conv2’

3

23×23×6

2×2 Avg pooling, padding [1 1 1 1],

Average pooling ‘pool2’

4

Stride 1, input shape 11×11×16

ReLU

232,440 Fully connected with dense 120,

Fully connected ‘fc1’

5

input shape 5×5×16

ReLU

10,164 Fully connected with dense 84

Fully connected ‘fc2’

6

Softmax

1105 Fully connected with dense 13 / 10

Fully connected ‘fc3’

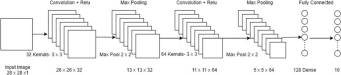
7

Non-trainable

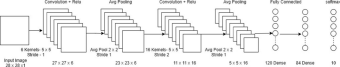
Trainable params: 246,281 Total param:

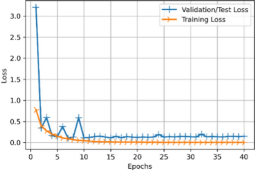
params: 0246,281

1 3

Handwritten Devanagari Character Recognition Using Modifed… 363 

**Fig. 3** CNN Architecture

**Fig. 4** Modifed LeNet CNN Architecture

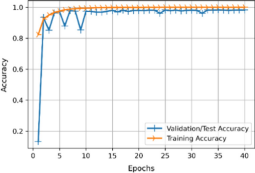


**Fig. 5** Loss for 40 Epochs for ACNN for 80:20 Vowels Dataset

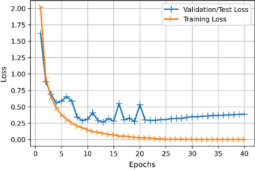
**5.2.3 For ACNN**

ACNN has its frst layers with 28 kernels that are regularized with the L2 regularization technique. L2 regularization helps to avoid the overftting of data from the frst layer itself. Batch normalizations are added to all the convolution layers with activation function ReLU that introduces non-linearity to the data. Kernels that are used in each layer are 56, 112, 224, 448 with the window size of (3,3) (5,5) (7,7) (3,3), and (3,3), respectively. Since there are essential parameters in Alexnet CNN – to avoid further overftting, dropout is added at the ratio of 50% for the frst two fully connected layers. The fnal output layer is softmax. The architecture of MLCNN is as mentioned in the Table 7.

1 3

364 D. S. Prashanth et al. 

**Fig. 6** Accuracy for 40 Epochs for ACNN for 80:20 Vowels Dataset



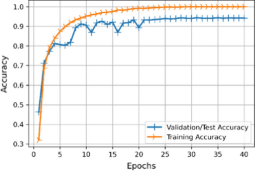
**Fig. 7** Loss for 40 Epochs for MLCNN for 80:20 Vowels Dataset

**5.3 Results Comparison and Training strategy of CNN, MLCNN and ACNN**

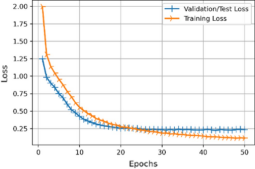
Three diferent CNN architectures are designed before performing classifcation and trained as per the ratios in Table  8. Table  11 shows the training accuracies achieved by CNN, MLCNN, and ACNN for alphabets and numerals (Fig. 11, Tables 9, 10).

A simple CNN architecture is used in this research for both alphabets and numer als. Alphabets achieved an accuracy of 96.88% of efciency for 70:30 split ratio at 50 epochs and 96.29%, 96.00% for 80:20 and 60:40 as shown in the Fig.  12 split ratio. ACNN architecture obtained an accuracy of 99.99% is collected for 40 epochs training of alphabets at 80:20 ratio split as shown in the Fig. 8. The other accuracies obtained

1 3

Handwritten Devanagari Character Recognition Using Modifed… 365 

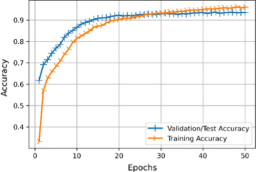
**Fig. 8** Accuracy for 40 Epochs for MLCNN for 80:20 Vowels Dataset



**Fig. 9** Loss for 50 Epochs for CNN for 60:40 Vowels Dataset

at 70:30 and 60:40 is 99.98% and 99.97%, respectively. The training of alphabets is stopped at 40 epochs; the reason for not recording the 50 epochs training is—loss values beyond 40 epochs are increasing and crossing the local minima in the opposite direc tion. A similar problem was raised for MLCNN. The training strategy used in MLCNN is, where SGD is using as the optimizer, the learning rate is updated to 0.001 from 0.01. The advantage of using SGD over the Adadelta optimizer is tuning the learning rate. By slowing down the learning rate, the accuracy is compromised. In MLCNN, at 33rd epoch loss is increasing so, the learning rate is tuned to 0.001. Later, it is observed that the cost value is not reversed when it is trained for 50 epochs. When the network is trained for 40 epochs as shown in the Fig. 10, the accuracy obtained is 99.98%, 99.98%,

1 3

366 D. S. Prashanth et al. 

**Fig. 10** Accuracy for 50 Epochs for CNN for 60:40 Vowels Dataset

and 99.97% for 80:20, 70:30 and 60:40 at 33rd, 35th, and 35th epoch simultaneously. The problem of overftting was raised for alphabets, because of 13 classes in HDCR of alphabets. Eight categories of characters look similar to minor changes, as shown in Table 3.

{1,2}, {3, 6}, and {5, 7, 8} sets of numbers in Devanagari have structural similarity as shown in Table 4. CNN architecture obtained training accuracy of 97.22%, 96.86%, and 96.77% at diferent splits. ACNN obtained top accuracy of 99.87% for 80:20 ratio at 40 epochs as show in the Fig. 12 and Fig. 13. MLCNN has shown promising results with the least loss for both training and testing, as shown in Table 12. It obtained an accuracy of 99.84%, 99.83%, and 99.82% at diferent splits and epochs (Fig. 14).

A trained model efciency can be calculated not only based on training data accu racy. But, based on accuracy, it gives, when it encounters unseen data. Table 12 indi cates the accuracy achieved by the sensed data by the neural network model. Overall, the training accuracies of all the three models are similar when it is compared to test ing accuracy/ validation accuracy or accuracy on the unseen dataset; there are some diferences in the values. ACNN for alphabets obtained an accuracy of 98.25% accu racy for 80:20 split at 40 epochs. The regularization techniques L2 and dropouts are used in this ACNN helped to give better accuracy. 98.25% and 98.32% for 70:30 and 60:40 split of data. Whereas, numerals are considered 97.22% of accuracy is achieved at 70:30 for 40 epochs. The loss calculated over epochs for vowels dataset are shown in the Figs. 7, 9 and 11 for ACNN, MLCNN and CNN respectively. The loss calculated for MLCNN and CNN for numerals dataset is shown in the Figs. 11 and 14.

Each Deep Learning Network is trained and tested for 12 number of times for alpha bets data and 12 number of times for numbers data. This process is categorized into 4,

1 3

Handwritten Devanagari Character Recognition Using Modifed… 367

Alexnet CNN Architecture **Table 7**

No. of params

Param

Image input Layer (type)

280 28 kernel 3×3 and input shape 28×28

Convolution ‘conv1’

1

112 Batch normalization

Batch normalization ‘norml’

2

–

ReLU

ReLU ‘activation1’

3

–

2×2 max pooling

Max pooling ‘pool1’

4

39,256 56 kernel 5×5 and padding [1 1 1 1]

Convolution ‘conv2’

5

224 Batch normalization

Batch normalization ‘norm2’

6

–

ReLU

ReLU ‘activation2’

7

–

2×2 max pooling

Max pooling ‘pool2’

8

–

Zero Padding [1 1 1 1]

Zero padding ‘zeropadding3’

9

307,440 112 kernel 7×7 and padding [1 1 1 1]

Convolution ‘conv3’

10

448 Batch normalization

Batch normalization ‘norm3’

11

–

ReLU

ReLU ‘activation3’

12

–

2×2 max pooling

Max pooling ‘pool3’

13

–

Zero Padding [1 1 1 1]

Zero padding ‘zeropadding4’

14

226,016 224 kernel 3×3 and padding [1 1 1 1]

Convolution ‘conv4’

15

896 Batch normalization

Batch normalization ‘norm4’

16

–

ReLU

ReLU ‘activation4’

17

–

Zero padding [1 1 1 1]

Zero padding ‘zeropadding5’

18

903,616 448 kernel 3×3 and padding [1 1 1 1]

Convolution ‘conv5’

19

1792 Batch normalization

Batch normalization ‘norm5’

20

–

ReLU

ReLU ‘activation5’

21

–

2×2 max pooling

Max pooling ‘pool5’

22

5,620,496 Fully connected Layer with dense 786

Fully connected ‘fc6’

23

3136 Batch normalization

Batch normalization ‘norm6’

24

–

ReLU

ReLU ‘activation6’

25

–

50% dropout

Dropout ‘drop6’

26

945,140Fully connected layer with dense 1204

Fully connected ‘fc7’

27

1 3

368 D. S. Prashanth et al.

(continued) **Table 7**

No. of params

Param

Image input Layer (type)

4816 Batch normalization

Batch normalization ‘norm7’

28

–

ReLU

ReLU ‘activation7’

29

–

50% dropout

Dropout ‘drop7’

30

15,665 Fully connected layer with dense 10

Fully connected ‘fc8’

31

52 Batch normalization

Batch normalization ‘norm8’

32

–

Softmax

Softmax ‘prob’

33

–

Crossentropy with Adadelta optimizer

Classifcation output ‘Output’

34

Non-trainable params: 5738Trainable params: 8,063,647

Total param:

8,069,385

1 3

Handwritten Devanagari Character Recognition Using Modifed… 369

**Table 8** Data split for training

and testing Training: testing Numerals Alphabets Training Testing Training Testing

80:20 18,000 4500 13,000 3250

70:30 15,750 6750 11,375 4875

60:40 13,500 9000 9750 6500

**Table 9** Accuracy on training data

Training Accuracies for Alphabets (13 Classes) Training Accuracies for Numerals (10 Classes) Train: Test ACNN MLCNN CNN Train: Test ACNN MLCNN CNN

20 Epochs 20 Epochs

80:20 99.92 99.10 91.95 80:20 99.69 99.82 95.54 70:30 99.89 99.31 90.63 70:30 99.65 99.82 95.37 60:40 99.96 97.81 91.37 60:40 99.61 99.81 94.36 30 Epochs 30 Epochs

80:20 99.99 99.98 94.01 80:20 99.79 99.82 96.10 70:30 97.91 99.96 93.83 70:30 99.86 99.84 96.01 60:40 99.22 99.91 93.22 60:40 99.84 99.83 95.42 40 Epochs 40 Epochs

80:20(39) 99.99 99.98 95.32 80:20 99.87 99.82 96.66 70:30 99.98 99.98 95.50 70:30 99.85 99.81 96.56 60:40 99.97 99.97 94.91 60:40 99.87 99.81 96.04 50 Epochs 50 Epochs

80:20 – 95.32 96.29 80:20 99.88 99.83 97.22 70:30 – 94.28 96.68 70:30 99.83 99.81 96.86 60:40 – 92.53 96.00 60:40 99.81 99.81 96.77

i.e., 20 epochs, 30 epochs, 40 epochs, and 50 epochs. Every category is further trained and test based on the split of data into diferent ratios, i.e., 80:20, 70:30, 60:40. The proposed deep learning architectures (CNN, MLCNN, and ACNN) are developed and tested on the dataset for 12 times. The dataset is tested for 72 times. For every instance, four parameters are taken to check the model stability. Training and testing accuracy and loss values are collected. Among the 72 diferent sets of values, Table 11 contains the optimal values. A study measures it, maximum accuracy of both training and test

ing, along with the minimum costs of training and testing loss.

1 3

370 D. S. Prashanth et al.

**Table 10** Accuracies calculated on the unseen data

Testing Accuracies for Alphabets (13 Classes) Testing Accuracies for Numerals (10 Classes) Train: Test ACNN MLCNN CNN Train: Test ACNN MLCNN CNN

20 Epochs 20 Epochs

80:20 98.03 91.41 92.46 80:20 97.07 96.56 96.0 70:30 97.74 88.86 91.85 70:30 96.17 96.49 95.44 60:40 97.40 88.40 91.88 60:40 97.30 96.57 95.79 30 Epochs 30 Epochs

80:20 98.25 91.56 93.85 80:20 97.20 96.64 96.24 70:30 97.91 91.87 93.05 70:30 96.74 96.41 96.04 60:40 98.26 90.88 92.08 60:40 97.27 96.41 95.86 40 Epochs 40 Epochs

80:20(39) 98.25 94.09 93.54 80:20 97.16 96.71 96.18 70:30 98.24 91.73 93.62 70:30 97.22 96.40 96.21 60:40 98.32 91.11 93.05 60:40 97.27 96.49 96.20 50 Epochs 50 Epochs

80:20 – 88.86 94.31 80:20 97.27 96.58 96.44 70:30 – 87.53 94.21 70:30 97.28 96.55 96.44 60:40 – 85.83 93.52 60:40 97.26 96.46 96.28

**5.4 Comparison of Diferent Methods HDCR**

To show the pre-eminence of the proposed architecture and dataset efciency, we com pare the performances of diferent models from the recent review articles [2, 6, 8]. The parameter considered for comparison is the size of the dataset, because, lower the size of the dataset leads to cover fewer variant styles and may achieve higher accuracies but when the dataset size and more variants of writing styles are introduced in the dataset may lead to a drop of accuracies. In the Table 12, the highest accuracy achieved is 99% in [32], whereas the size of the dataset is 5424. Similarly, the dataset of sized 10,000 [31], 25,000 [18], 22,556 [23], and 32,413 [32], got accuracy of 97.18%, 90.11%, 93.4% and 61.8% respectively. It is observed that the accuracy is reduced when more variants of data or data size increase. In this article, the dataset size of 22,500 images could able to achieve 99.9% training data and 98.25% for unseen datasets for numbers. For the dataset size of 16,250 images, could able to achieve 99.87% for training data and 97.16 for unseen data. our model has signifcant improvement.

1 3

Handwritten Devanagari Character Recognition Using Modifed… 371

Top Accuracies **Table 11**

ACNN for Numerals (10 Classes) ACNN for Alphabets (13 Classes)

40 Epochs 40 Epochs

Test Loss Train Loss Testing Accuracy Training Accuracy Train: Test Test Loss Train Loss Testing Accuracy Training Accuracy Train: Test

0.20 0.003

97.16

99.87 80:20 0.14 0.001

98.25

99.99 80:20

0171 0.004

97.22

99.85 70:30 0.13 0.001

98.24

99.98 70:30

0.17 0.003

97.27

99.87 60:40 0.11 0.001

98.32

99.98 60:40

MLCNN for Numerals (10 Classes) MLCNN for Alphabets (13 Classes)

30 Epochs 40 Epochs

Test Loss Train Loss Testing Accuracy Training Accuracy Train: Test Test Loss Train Loss Testing Accuracy Training Accuracy Train: Test

0.21 0.002

96.64

99.82 80:20 0.34 0.001

94.09

99.98 80:20 (33)

0.21 0.002

96.41

99.84 70:30 0.50 0.0010

91.73

99.98 70:30 (35)

0.20 0.002

96.41

99.83 60:40 0.50 0.001

91.11

99.97 60:40(35)

CNN for Numerals (10 Classes) CNN for Alphabets (13 Classes)

50 Epochs 50 Epochs

Test Loss Train Loss Testing Accuracy Training Accuracy Train: Test Test Loss Train Loss Testing Accuracy Training Accuracy Train: Test

0.12 0.07

96.44

97.22 80:20 0.21 0.10

94.31

96.29 80:20

0.12 0.08

96.44

96.86 70:30 0.23 0.09

94.21

96.68 70:30

0.120.09

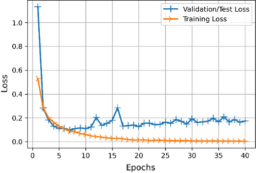
96.28

96.77 60:40 0.23 0.11

93.52

96.00 60:40

1 3

372 D. S. Prashanth et al. 

**Fig. 11** Loss for 50 Epochs for MLCNN for 80:20 Numerals Dataset



**Fig. 12** Accuracy for 50 Epochs for MLCNN for 80:20 Numerals Dataset

**6 Conclusion**

In this paper, we provide a dataset of 38,750 images of Devanagari numerals and vow els collected from 2400 subjects of diferent age groups. We also developed a segmenta tion algorithm suitable for the dataset. A modifed CNN is proposed to solve the HDCR

1 3

Handwritten Devanagari Character Recognition Using Modifed… 373

**Table 12** Diferent methods and accuracy of HDCR

Method Feature Accuracy (%) Dataset

Tree based classifer and Tem plate matching based approach [31]

Structural features 97.18 10,000

KNN and neural network [32] Gradient features 61.8 32,413

MLP and SVM classifer [18] Chain code, Kirsch Directional edges, Gradient, distance transform

90.116 by SVM and 87.56 by MLP

25,000

Neural network classifer [23] Combination of structural and statistical features

93.4 22,556

MQDF [32] Directional features 99.0 5424 MLCR [34] Geometric distance based 98.1 3100 CNN [35] Convolution 95 4282

KNN [36] Hu’s seven variants and zernike moment features

MLCNN Modifed LeNet convolutional neural network

89 2600 99.9 38,750



**Fig. 13** Loss for 50 Epochs for CNN for 80:20 Numerals Dataset

problem. The dataset developed followed the standards of MNIST and served as a bench mark dataset for Devanagari numerals and vowels. Experiments are conducted by using three diferent deep learning architectures, 1. CNN, 2. MLCNN, and 3. ACNN. It achieved an error rate of 0.001% and 99.98% accuracy. Through a series of experiments, we have shown that MLCNN and ACNN are highly efcient (99.9% and 99.8%). ACNN’s compu tational cost is high when compared to MLCNN but can overcome by GPU based systems.

1 3

374 D. S. Prashanth et al. 

**Fig. 14** Loss for 50 Epochs for CNN for 80:20 Numerals Dataset

We implemented a CNN by modifying LeNet 5 CNN, which reduces the time signifcantly, where these models can be used in online HDCR. The performance of ACNN can be fur ther improved through the fne-tuning of its structure and its parameters. In future work, we plan to develop a combined model for all the Devanagari characters of more than 50 classes that include numeral (10), vowels (13), and consonant (33).

**Authors’ contributions** Duddela Sai Prashanth: Conceptualization, Methodology, Software, Visualization R Vasanth Kumar Mehta: Data curation, Writing—original draft, Data Analysis, Investigation. Kadiyala Ramana: Software, Validation, Editing. Vidya Charan Bhaskar: Supervision, Writing—review & editing.

**Funding** This research did not receive any specifc grant from funding agencies in the public, commercial, or not-for-proft sectors.

**Availability of data and material** The datasets generated during and/or analysed during the current study are available in the Mendeley repository, https://data.mendeley.com/datasets/pxrnvp4yy8/1

**Code availability** The code that supports the fndings of this study are available on request from the cor responding author. The code is not publicly available due to containing information that could compromise the privacy of research participants.

**Declarations**

**Confict of interests** The authors declare that they have no known competing fnancial interests or personal relationships that could have appeared to infuence the work reported in this paper.

**References**

1. Sahare, P., & Dhok, S. B. (2019). Robust character segmentation and recognition schemes for mul tilingual indian document Images. *IETE Technical Review, 36*(2), 209–222.

2. Parekh, K. A., Goswami, M. M., & Mitra, S. K. (2020). Handwritten numeral recognition using polar histogram of low-level stroke features. *In Proceedings of 3rd international conference on computer vision and image processing* (pp. 169–181). Springer.

1 3

Handwritten Devanagari Character Recognition Using Modifed… 375

3. Bansal, V., & Sinha, R. M. K. (2002). Segmentation of touching and fused Devanagari characters. *Pattern recognition, 35*(4), 875–893.

4. Li, Z., Teng, N., Jin, M., & Lu, H. (2018). Building efcient CNN architecture for ofine handwritten Chinese character recognition. *International Journal on Document Analysis and Recognition (IJDAR), 21*(4), 233–240.

5. Kaur, S., & Sagar, B. B. (2019). Brahmi character recognition based on SVM (support vector machine) classifer using image gradient features. *Journal of Discrete Mathematical Sciences and Cryptography, 22*(8), 1365–1381.

6. Setlur, S. (2009). Guide to OCR for indic scripts. In V. Govindaraju (Ed.). Springer. 7. Bathla, A. K., Gupta, S. K., & Jindal, M. K. (2016, March). Challenges in recognition of Devanagari Scripts due to segmentation of handwritten text. *In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 2711–2715). IEEE.

8. https://data.mendeley.com/datasets/pxrnvp4yy8/1

9. Bharath, A., & Madhvanath, S. (2011). HMM-based lexicon-driven and lexicon-free word recognition for online handwritten Indic scripts. *IEEE transactions on pattern analysis and machine intelligence, 34*(4), 670–682.

10. Soora, N. R., & Deshpande, P. S. (2018). Review of feature extraction techniques for character recogni tion. *IETE Journal of Research, 64*(2), 280–295.

11. Sharma, A. K., Adhyaru, D. M., & Zaveri, T. H. (2020). A survey on Devanagari character recognition. *In Smart systems and IoT: Innovations in computing* (pp. 429–437). Springer.

12. Goswami, M. M., Prajapati, H. B., & Dabhi, V. K. (2011). Classifcation of printed Gujarati characters using SOM based k-Nearest Neighbor Classifer. In *2011 International conference on image informa tion processing* (pp. 1–5). IEEE.

13. Sinha, R. M. K., & Mahabala, H. N. (1979). Machine recognition of Devanagari script. *IEEE Transac tions on Systems, Man and Cybernetics, 9*(8), 435–441.

14. Jayanthi, K., Suzuki, A., Kanai, H., Kawazoe, Y., Kimura, M., & Kido, K. (1989, November). Devana gari character recognition using structure analysis. In *Fourth IEEE Region 10 international conference TENCON* (pp. 363–366). IEEE.

15. Bansal, V., & Sinha, M. K. (2001, September). A complete OCR for printed Hindi text in Devanagari script*. In Proceedings of Sixth International Conference on Document Analysis and Recognition* (pp. 0800–0800). IEEE Computer Society.

16. Sharma, N., Pal, U., Kimura, F., & Pal, S. (2006). Recognition of ofine handwritten devnagari char acters using quadratic classifer. In *Computer vision, graphics and image processing* (pp. 805–816). Springer.

17. Arora, S., Bhattacharjee, D., Nasipuri, M., Basu, D. K., Kundu, M., & Malik, L. (2009, December). Study of diferent features on handwritten Devnagari character. In *2009 Second international confer ence on emerging trends in engineering & technology* (pp. 929–933). IEEE.

18. Kumar, S. (2009). Performance comparison of features on Devanagari hand-printed dataset. *Interna tional Journal of Recent Trends in Engineering, 1*(2), 33.

19. Arora, S., Bhattacharjee, D., Nasipuri, M., Basu, D. K., & Kundu, M. (2010). Recognition of non compound handwritten devnagari characters using a combination of mlp and minimum edit dis tance. arXiv preprint arXiv:1006.5908.

20. Pal, U., Chanda, S., Wakabayashi, T., & Kimura, F. (2008). Accuracy improvement of Devnagari character recognition combining SVM and MQDF. In *Proceedings of 11th International Con ferenece on Frontiers Handwrit. Recognition* (pp. 367–372).

21. Kale, K. V., Deshmukh, P. D., Chavan, S. V., Kazi, M. M., & Rode, Y. S. (2013). Zernike moment feature extraction for handwritten Devanagari compound character recognition. In *2013 Science and information conference* (pp. 459–466). IEEE.

22. Arora, S., Bhatcharjee, D., Nasipuri, M., & Malik, L. (2007). A two stage classifcation approach for handwritten Devnagari characters. In *International conference on computational intelligence and multimedia applications (ICCIMA 2007)* (Vol. 2, pp. 399–403). IEEE.

23. Hanmandlu, M., Murthy, O. R., & Madasu, V. K. (2007, December). Fuzzy model based recogni tion of handwritten Hindi characters. In *9th Biennial Conference of the Australian Pattern Recogni tion Society on Digital Image Computing Techniques and Applications (DICTA 2007)* (pp. 454– 461). IEEE.

24. Aneja, N., & Aneja, S. (2019, July). Transfer learning using CNN for handwritten Devanagari char acter recognition. In *2019 1st International Conference on Advances in Information Technology (ICAIT)* (pp. 293–296). IEEE.

1 3

376 D. S. Prashanth et al.

25. Arora, S., Bhattacharjee, D., Nasipuri, M., Malik, L., Kundu, M., & Basu, D. K. (2010). Perfor mance comparison of SVM and ANN for handwritten devnagari character recognition. arXiv pre print arXiv:1006.5902

26. Lu, S., Lu, Z., & Zhang, Y. D. (2019). Pathological brain detection based on AlexNet and transfer learning. *Journal of computational science, 30*, 41–47.

27. Shima, Y., Nakashima, Y., & Yasuda, M. (2018, April). Handwritten Digits Recognition by Using CNN Alex-Net Pre-trained for Large-scale Object Image Dataset. In *Proceedings of the 3rd inter national conference on multimedia systems and signal processing* (pp. 36–40).

28. Kumar, M., Jindal, M. K., Sharma, R. K., & Jindal, S. R. (2019). Character and numeral recogni tion for non-Indic and Indic scripts: A survey. *Artifcial Intelligence Review, 52*(4), 2235–2261. 29. Niu, X. X., & Suen, C. Y. (2012). A novel hybrid CNN–SVM classifer for recognizing handwritten digits. *Pattern Recognition, 45*(4), 1318–1325.

30. Jiang, B., He, J., Yang, S., Fu, H., Li, T., Song, H., & He, D. (2019). Fusion of machine vision tech nology and AlexNet-CNNs deep learning network for the detection of postharvest apple pesticide residues. *Artifcial Intelligence in Agriculture, 1*, 1–8.

31. Chaudhuri, B. B., & Pal, U. (1997, August). An OCR system to read two Indian language scripts: Bangla and Devnagari (Hindi). In *Proceedings of the fourth international conference on document analysis and recognition* (Vol. 2, pp. 1011–1015). IEEE.

32. Kompalli, S., Nayak, S., Setlur, S., & Govindaraju, V. (2005, August). Challenges in OCR of Devanagari documents. In *Eighth International Conference on Document Analysis and Recognition (ICDAR’05)* (pp. 327–331). IEEE.

33. Pal, U., Sharma, N., Wakabayashi, T., & Kimura, F. (2007, September). Handwritten numeral rec ognition of six popular Indian scripts. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* (Vol. 2, pp. 749–753). IEEE.

34. Soora, N. R., & Deshpande, P. S. (2017). Novel geometrical shape feature extraction techniques for multilingual character recognition. *IETE Technical Review, 34*(6), 612–621.

35. Prashanth, D. S., Mehta, R. V. K., & Sharma, N. (2020). Classifcation of Handwritten Devanagari number-an analysis of pattern recognition tool using neural network and CNN. *Procedia Computer Science, 167*, 2445–2457.

36. Prashanth, D. S., & Panini, C. N. (2017). KNN classifcation of Kannada Characters using Hu’s Seven Variants and Zernike Moment. *Nagendra Panini Challa* on Aug, 10.

37. Latif, G., et  al. (2018). Deep convolutional neural network for recognition of unifed multi-lan guage handwritten numerals. *2018 IEEE 2nd International workshop on Arabic and derived script analysis and recognition (ASAR)*. IEEE.

38. Chakraborty, B., et al. (2018). Does deeper network lead to better accuracy: a case study on hand written Devanagari characters. *2018 13th IAPR International Workshop on Document Analysis Sys tems (DAS)*. IEEE.

39. Rojatkar, D. V., Chinchkhede, K. D., & Sarate, G. G. (2013). Design and analysis of LRTB feature based classifer applied to handwritten Devnagari characters: A neural network approach. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE,.

40. Ashokkumar, P., Siva Shankar, G., Gautam Srivastava, Praveen Kumar Reddy Maddikunta, & Thippa Reddy Gadekallu. 2021. “A Two-stage Text Feature Selection Algorithm for Improving Text Clas sifcation”. *ACM Trans. Asian Low-Resour. Lang. Inf. Process*. 20, 3, Article 49 (July 2021), 19 pages. DOI:https://doi.org/10.1145/3425781

41. Dhanamjayulu, C., Nizhal, U. N., Kumar Reddy Maddikunta, P., Thippa Reddy Gadekallu, Celestine Iwendi, Chuliang Wei, & Qin Xin. (2021). Identifcation of malnutrition and prediction of BMI from facial images using real-time image processing and machine learning. *IET IMAGE PROCESSING*.

42. Gadekallu, T. R., Alazab, M., Kaluri, R., Reddy Maddikunta, P. K., Bhattacharya, S., Lakshmanna, K., & Parimala, M. (2021). Hand gesture classifcation using a novel CNN-crow search algorithm. *Com plex & Intelligent Systems, 7*, 1–14.

43. Kothai, G., Poovammal, E., Dhiman, G., Ramana, K., Sharma, A., AlZain, M. A., & Masud, M. (2021). A new hybrid deep learning algorithm for prediction of wide trafc congestion in smart cities. *Wireless Communications and Mobile Computing, 2021*, 1.

**Publisher’s Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional afliations.

1 3

Handwritten Devanagari Character Recognition Using Modifed… 377

**Duddela Sai Prashanth,** received M.Tech from Indian Institute of Information Technology, Allahabad and pursuing his PhD from SCS VMV University. He is serving as Assistant Professor from 2015 in Sahyadri College of Engineering & Management, Mangaluru. He is Co-Principal Investigator for the project sanctioned 30,00,000 INR by Vision Group of Science and Technology to establish Center of Excel lence in Artifcial Intelligent and Machine Learning. 

**Dr R Vasanth Kumar Mehta** received M.S from BITS Pilani and the PhD Degree from SCSVMV University. He Joined SCSVMV Univer sity as Assistant Professor in the year 2001, where he is now Associate Professor. He is serving the University with the diferent responsibili ties, as Board of Studies – Chairman, enabling of-campus internships and placements, Interaction with industries and academia in India and abroad for enhancing teaching, placements, industry grade projects and research. His Area of Interest lies in Data Mining, Distributed Computing Domain, Pattern Recognition. He is professional member of IEEE, CSI, ACM, IAENG. He is a member in Editorial Boards and Reviewer of reputed Journals. 

**Dr. Kadiyala Ramana** is currently working as Associate Professor in Department of Artifcial Intelligence & Data Science, AITS, Rajam pet, India. He obtained his Bachelor of Technology degree in Informa tion Technology from JNTUH, Hyderabad, India, M. Tech. from Saty abhama University, Chennai, India and completed his Ph.D. from SRM University, Chennai, India. He has 14 years of experience in teaching and research. He authored more than 30 international publica tions in Hindawi, Springer, IEEE etc. He is editorial board member and reviewer of several journals of international repute. His research interests Wireless Sensor Networks, Software-Defned Networking with Machine Learning and Data Analytics.

1 3

378 D. S. Prashanth et al.

**Dr. Vidhyacharan Bhaskar** received the B.Sc. degree in Mathematics 

from the University of Madras, Chennai, India in 1992, M.E. degree in

Electrical & Communication Engineering from the Indian Institute of

Science, Bangalore in 1997, and the M.S.E. and Ph.D. degrees in Elec

trical Engineering from the University of Alabama in Huntsville in

2001 and 2002, respectively. During 2002-2003, he was a Post Doc

toral fellow with the Communications research group at the University

of Toronto, Canada, where he worked on the applications of space

time coding for wireless communication systems. During 2003-2006,

he was an Associate Professor in the Department of Information Sys

tems and Telecommunications at the University of Technology of

Troyes, France. From Jan. 2007 to May 2014, he was a Full Professor

in the Department of Electronics and Communication Engineering at

S.R.M. University, Kattankulathur, India. Since 2015, he is a Professor

in the Department of Electrical and Computer Engineering at San

Francisco State University, San Francisco, California, USA. His

research interests include MIMO wireless communications, signal pro

cessing, error control coding and queuing theory. He has published 130 Refereed Journal papers, presented around 72 Conference papers in various International Conferences over the past 20 years, published a book on “Adaptive Rate Coding for A-CDMA Systems” in Jan 2011, a book on “Higher-Order s-to-z mapping functions for digital flters,” in March 2013, and has also co-authored a book on MATLAB in 1998. He has 868 Google scholar citations. He has advised 50 Masters students, 11 Doctoral students, and 3 Senior design projects. He is an IEEE Senior member (SM-IEEE) and is a member of IET (M-IET, UK). He is a Fellow of Institute of Electronics and Telecommunication Engineers (F-IETE), and a Fellow of Institute of Engineers (F-IE), Kolkata, India. He is also a Life member of the Indian Society of Technical Education (LM-ISTE) and a member of the Indian Science Congress (M-ISC).

**Authors and Afliations**

**Duddela Sai Prashanth1,2 · R. Vasanth Kumar Mehta1 · Kadiyala Ramana3 · Vidhyacharan Bhaskar4**

Duddela Sai Prashanth

dsaip13@gmail.com

R. Vasanth Kumar Mehta

vasanthmehta@gmail.com

Kadiyala Ramana

ramana.it01@gmail.com

1 SCSVMV University, Kanchipuram, India

2 Sahyadri College of Engineering and Management, Mangaluru, India

3 Department of Artifcial Intelligence and Data Science, Annamacharya Institute of Technology and Sciences, Rajampet, Andhra Pradesh, India

4 Department of Electrical and Computer Engineering, San Francisco State University, San Francisco, CA 94132, USA

1 3