SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I11P117

Volume 12 Issue 11, 201-215, November 2025

Original Article

Smart Network-Based Classification of Handwritten Basic, Modified, and Complex Conjunct Characters in Devanagari Script

Pallavi Patil[1], Kevin Noronha[2]

[1,2]Department of Electronics and Telecommunication, St. Francis Institute of Technology, Maharashtra, India.

[1]Corresponding Author : pallavipatil@sfit.ac.in

Received: 17 September 2025 Revised: 17 October 2025 Accepted: 16 November 2025 Published: 29 November 2025

Abstract *- The Devanagari script is extensively used for documenting information in both the government and private sectors in India. In Devanagari documents, modified and conjunct characters frequently appear alongside consonants. Intricate forms of these characters, combined with variations in writing patterns, make the recognition process a challenging task. Most Devanagari datasets available online lack modified and conjunct characters. To address this issue, a dataset comprising basic, modified, and complex conjunct characters of the Devanagari script, which includes 580 classes, is created. Additionally, a segmentation algorithm is developed to automatically segment filled forms from different writers, thereby accelerating dataset processing. To classify these characters accurately, three convolutional neural networks, SmartNet1, SmartNet2, and SmartNet3, were designed by experimenting with various hyperparameters, such as the number of layers, filters, nodes in the fully connected layer, and kernel size. Each SmartNet was modified using a support vector machine and a k-nearest neighbors classifier, yielding a total of nine configurations. All nine configurations were tested on both self-generated and benchmark datasets. These networks are relatively shallow and have fewer parameters, enabling faster convergence and accomplishing remarkable results across different datasets.*

Keywords *- Conjunct, Devanagari, Modified, Segmentation, SFIT_Char.*

1. Introduction

The majority of Indian government entities, including courts, grampanchayats, post offices, and municipal corporations, receive thousands of documents every day that contain crucial information, both handwritten and printed forms. Manually digitizing this data is a laborious process. Automating this procedure [1, 2] will save administrative staff a great deal of time. Optical Character Recognition (OCR) has emerged as one of the most significant fields of study in pattern recognition over the past several decades [3]. The diverse applications of OCR technology in academia [4-6], the medical field [7, 8], and industry [9] are among the important reasons for active research in this field. The traditional method of data entry on a computer is via the keyboard, which is a time-consuming and cumbersome task, especially when processing large numbers of documents in government offices, banks, or courts. OCR helps alleviate this problem by automatically segmenting and recognizing characters in a text document. Compared to Roman [10] and Chinese script, OCRs on Indic scripts are rare. Lately, some researchers have developed OCR for Indian languages such as Bangla [11, 12], Malayalam, Telugu [13], Marathi [14], Hindi [15], Sanskrit [16, 17], and other multilingual [18, 19] scripts. Being an ancient script, Devanagari is used for writing various Indian languages, including Hindi, Sanskrit, and Marathi. It is widely used for official documentation, government communication, and administrative purposes in the regions where these languages are spoken. Developing OCR for the Devanagari script can not only help restore knowledge of ancient documents [20, 21] but also automate document processing in government offices. A typical OCR consists of three main modules: Preprocessing, Segmentation, and Recognition modules. The document to be digitized is first scanned to generate a PDF or converted to an image using a camera, and then fed to the preprocessing module. The preprocessing module converts the input document to a standard format by performing operations such as size normalisation, skew correction, binarisation, and more.

This preprocessed document is given to the segmentation module, which segments the entire document from lines to words to characters. These isolated characters are then fed to the recognition module, which extracts various shape-related features and assigns a class label based on them.

1.1. Devanagari Script, Characteristics, and Challenges

Devanagari script has 13 vowels, 34 consonants, and 10 numerals. The Devanagari characters, along with their corresponding International Alphabet of Sanskrit Transliteration (IAST) representation, are given below.
Vowels: अ, आ, इ, ई, उ, ऊ, ए, ऐ, ओ, औ, अं, अः, ऋ
IAST: a ā i ī u ū e ai o au ṃ ḥ ṛ
According to the origin of sound for consonants in the mouth [22], they are divided into five groups क -- Varga or Gutturals, where the origin is in the mouth, च -- Varga or Palatals, the sound is produced when the back of the tongue touches the palate, or the tongue touches the lower part of the gum area in the lower teeth. The Retroflex or ट -- Varga corresponds to the center of the palate, and the Dental or त -- Varga is produced when the tip of the tongue is between the teeth. Finally, the Labials or प -- Varga are produced using the lips.

Devanagari	**IAST**
Characters	
क, ख, ग, घ, ङ,	ka, kha, ga, gha, ṅa
च, छ, ज, झ, ञ,	ca, cha, ja, jha, ña
ट, ठ, ड, ढ, ण,	ṭa, ṭha, ḍa, ḍha, ṇa
त, थ, द, ध, न,	ta, tha, da, dha, na
प, फ, ब, भ, म,	pa pha ba bha ma
य, र, ल, व, श,	ya ra la va śa
ष, स, ह, ळ	ṣa sa ha Ḷa
Numerals ०, १, २, ३, ४, ५, ६, ७, ८, ९	0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Semi Vowels य, र, ल, व **Sibilants** श, ष, स
Aspirate ह **Conjunct Consonants** क्ष, त्र, ज्ञ

Vowels can modify these 34 consonants to generate the so-called Barakhadi.
क, का, कि, की, कु, कू, के, कै, को, कौ, कं, कः
Also, different consonants can come together to create conjunct characters. According to [22], there are specific rules under which two consonants can join to form a conjunct character. They are listed as follows.
Combining consonants with consonants:

- One point touch characters:- च, ज, ञ, त, न, त्र, ज्ञ
  Rule 1: Drop the first and the only contact or stick and attach to another
  त + य → त्य e.g., सत्य
  त+ त → त्त e.g., उत्तम
- Two point touch characters:- ख, घ, झ, थ, ध, प, भ, म, य, ष, स, क्ष
  Rule 2: Drop the second contact and attach to the other
  म + ब → म्ब e.g., अम्बा
  स + त → स्त e.g., अस्त
- Rounded bottom characters:- ङ, छ, ठ, ड, ढ
  च + छ → च्छ e.g., गच्छति
- A Combination of some other consonants
  श + व → श्व e.g., अश्व
  क + क → क्क e.g., कुक्कुट

Inherent characteristics of the Devanagari script, along with other factors, make script recognition challenging among the few challenges mentioned below.

1. Presence of modifiers and conjunct characters, which are complex in structure.
2. Variation in the writers' writing styles, such as skew, stroke thickness, and character spacing.
3. Two characters having almost the same shape, confusing during recognition, e.g., प and म or क and फ
4. Lack of an annotated dataset corresponding to modified and conjunct characters.
5. Poorly scanned documents, low quality of writing tools, and spreading of ink.

These challenges make it difficult to develop an accurate OCR for the Devanagari script. Along with basic characters, modified and conjunct characters frequently occur in the Devanagari document. Most of the research done so far focuses only on basic character recognition, lacking modified and conjunct characters. The unavailability of a dataset corresponding to modified and conjunct characters poses difficulties. The primary focus of this research is to address these challenges by developing an OCR classification module that accurately classifies not only basic but also modified and conjunct characters in the Devanagari script, along with a full-fledged dataset of 580 classes comprising basic, modified, and conjunct characters. It is anticipated that the availability of this dataset will have a positive impact on research in this direction.

The paper is structured as follows. Section 2 gives a detailed literature exploration. Section 3 discusses the methodology. Section 4 presents the results and discussion, and the conclusion and future scope are presented in Section 5.

2. Literature Exploration

An optical character recognition system enables a machine to recognize various patterns, including alphabets, numerals, and other symbols such as commas, question marks, and different characters. This type of machine learning is achieved by presenting examples of different characters from different classes to the machine and teaching it to learn distinct patterns within each class. Based on samples shown to the machine, it builds an instance, or prototype, of each class of characters. In the recognition phase, an unknown character is fed to the machine, which is then compared to previously obtained descriptions or instances. This test character is assigned to a particular class to which it matches the best. The discriminative capabilities of the extracted features affect the classification result. Especially while dealing with handwritten documents, slight intra-class variation and large inter-class variation are expected in the feature vector corresponding to each character.

In OCR systems, various features can be extracted from the individual characters, such as texture, shape, stroke, and geometry. As shown in Figure 1, these features can be broadly categorized into two types: textual and structural. Textual features are based on the general appearance and geometry of the character and extracted using different transforms, such as Fourier, Gabor [23], Scale-Invariant Feature Transform(SIFT), and wavelet, among others. Structural features are based on strokes, trajectory, constellation diagram [19], etc.

In the past few years, with the advancement of deep learning, there has been a sudden shift towards automated feature extraction using Convolutional Neural Networks (CNNs). Convolutional neural networks automatically extract features from edges, textures, and shapes in an image.

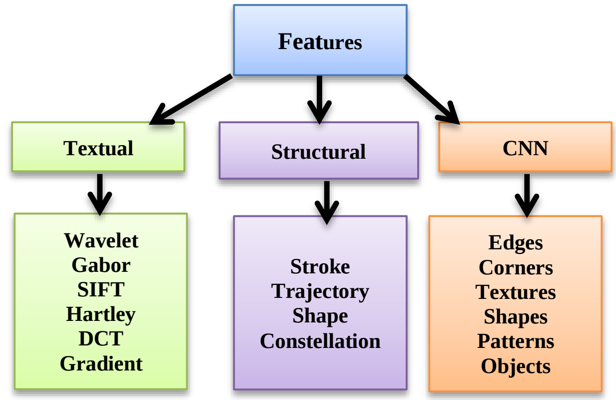


Fig. 1 Different types of features

A comparative analysis of recent literature work is given in Table 1.

Table 1. Comparative analysis of recent literature work

Year	Objective	Features	Classifier	Dataset	Accuracy	Ref
2018	Multilingual character segmentation and recognition	Fixed Center Distance based Feature (FCDF), Fixed Center Cut based Feature(FCCF), Neighborhood Count based Feature(NCF)	k-Nearest Neighbor classifier with different distances like Euclidean, city block, and cosine	For Latin Script chars74k, CVLSD, and proprietary, for Devanagari CPAR, V2DMDCHAR, and proprietary	87.42% to 99.84%	[19]
2018	Handwritten Devanagari character recognition	CNN-based features	Layer-wise trained six DCNN models	V2DMDCHAR and ISIDCHAR	98%	[24]
2018	Handwritten Devanagari character recognition	CNN-based features	5 different CNN models	DHCD dataset	96.9%	[25]
2019	Devanagari ancient document recognition	Structural/Statistical : intersection and open endpoints, centroid, horizontal and vertical peak extent	CNN,MLP, RBF-SVM	Dataset of 6152 characters(34 classes) pre-segmented from ancient manuscripts	88.95%	[26]
2020	Devanagari dataset creation and handwritten Devanagari character recognition	CNN-based features	Two-stage VGG16 model	DHCD dataset CMATERdb Bangla dataset, HMDC dataset	97.80% 99.40%, 95.83%-97.45%, 96.55%	[27]
2022	Dataset creation of Devanagari Numerals and	CNN-based features	Modified Lenet and Alexnet	Self-created Dataset	99.9%	[28]

	Vowels, and recognition of handwritten Devanagari characters					
2022	Digitization of handwritten Devanagari text	CNN-based features	12-layer CNN model including Dropout layer	DHCD dataset	99.13%	[7]
2022	Recognition of handwritten Devanagari characters	Textual: Zigzag DCT feature Structural:- intersection points, number of horizontal lines, number of vertical lines, length of vertical lines	Linear Discriminant Analysis(LDA), SVM,kNN, and Weighted kNN	3877 characters extracted from the handwritten Devanagari documents	93.6%	[29]
2023	To recognize Devanagari handwritten Numerals using a less computationally complex network	CNN-based features	Shifted Window Transformer fine-tune with VGG-16Net, ResNet-50, and DenseNet-121	DHCD Numeral Dataset	99.20%	[30]
2024	Augment the existing dataset with synthetic images to enhance handwritten Devanagari character recognition.	CNN-based features	Generative Adversarial Networks	DHCD dataset	98.33% to 98.86%	[31]
2024	Handwritten Devanagari character recognition using transfer learning	CNN-based features	Fine-tuned VGG16 model	DHCD dataset	96.58%	[32]
2025	Handwritten Devanagari character recognition	CNN-based features	Modified CapsNet	DHCD dataset	99.30%	[33]

Although there has been a significant amount of research done on the classification of basic characters that are consonant, an exhaustive study on modified and conjunct characters is still lacking. The absence of a dataset corresponding to modified and conjunct characters poses a difficulty for the study. The use of deep networks with a large number of parameters for recognizing character images with minimal background details should be questioned.

This directs research toward the problem statement: Recognition of handwritten modified and conjunct characters of the Devanagari script using convolutional neural networks with fewer parameters and faster convergence. The key contributions of this research are listed as follows:-

1. Development of the only dataset of Devanagari characters to encompass 580 classes comprising consonants, modified, and complex conjunct characters.
2. The inclusion of the character **ळ,** which is specific to the Marathi language, has not been adequately addressed by many researchers so far.
3. Conjunct characters are developed by considering the rules mentioned in the introduction section.
4. Developed a segmentation algorithm for automatically segmenting the filled forms, avoiding the need for manual cropping of each character, resulting in fast processing of the dataset. With a minimal threshold

change, the same can also be used to generate other datasets.

5. Comprehensive experimentation across different hyperparameters yielded three innovative models with two variants of each model, making a total of 9 architectures designed for classification. Innovative models developed have fewer trainable parameters, hence saving time and memory.
6. Performed exhaustive experimentation on different available standard datasets, achieving state-of-the-art results with developed intelligent networks. The networks are found to be escalating the test accuracy for different datasets.

3. Methodology

3.1. SFIT_char Dataset Development

Although a few datasets are available online, most include only vowels, consonants, and numerals. It was highly necessary to create a dataset of modified and conjunct characters, as they appear frequently alongside the basic characters in Devanagari documents. The main reason for developing this dataset is to provide data for modified and conjunct characters, as well as consonants. This dataset was developed over 1.5 years by collecting samples from more than 150 individuals of varying ages, academic backgrounds, and professions, leading to font size and style variations across the images. Writers were instructed to write within the provided box. The kind of pen, ink color, and writing hand were all unrestricted. Sufficient time was given to the writers to complete the form. All the filled forms are scanned at 200 dpi. The characters are extracted by an algorithm written explicitly for the given form, improving the time efficiency. The extracted dataset is then cleaned to remove incorrectly cropped characters, as the process was automated. It was later realized that occurrences of some characters corresponding to barakhadi are rare, so they were removed. The final dataset comprises 36 classes for consonants, 465 classes for modified characters, and 79 classes for conjunct characters. The images corresponding to each class range from approximately 140 to 220. The total number of images in the dataset is 103,115. So far, this is the only dataset with such a diverse range of character classes. The total number of images corresponding to consonants is 6767, the total number of images corresponding to modified characters is 83169, and the total number of conjunct characters is 13179.

Characters in the dataset

Complete set of consonants: क, ख, ग, घ, ङ, च, छ, ज, झ, ञ, ट, ठ, ड, ढ, ण, त, थ, द, ध, न, प, फ, ब, भ, म, य, र, ल, व, श, ष, क्ष ,स, ह, ळ, ज्ञ

Few examples of modified characters: क, का, कि, की, कु, कू, के, कै, को, कौ, कं, कः क्र, र्क, कॅ

Few examples of conjunct characters: स्वा ,स्व ,व्य ,क्त ,त्न ,स्था ,स्त ,ल्या ,च्या ,ध्ये ,त्या ,क्य ,च्य ,ध्या ,श्चि, न्ही

Developing a dataset is a cumbersome task because form processing takes significant time, especially when manually cropping characters. To expedite this process, a segmentation algorithm, as shown in Table 2, has been developed, which automatically crops the characters from the scanned document and stores them in a designated folder. It involves two stages, preprocessing and segmentation. In the preprocessing stage, scanned documents are brought into a standard format.

In the segmentation stage, horizontal and vertical projection profiles are calculated, based on which the segmentation paths are generated. Using the horizontal segmentation path (h_{sp})shown by red lines and the vertical segmentation path (v_{sp}) shown by blue lines in Figure 2, the character is cropped. This algorithm was effective in segmenting over hundreds of forms in a very short period. A few samples of segmented characters are shown in Figure 2(c). By adjusting the threshold values, the algorithm can also be used to segment other forms of data.

Table 2. Algorithm for segmenting the first and second pages of the form

Algorithm 1.	**Segmentation path calculation**
Input:	First and second page of the form in the form of an image *I*
Output:	Individual segmented characters
Steps:	
1.	Start
	Preprocessing
2.	RGB to grayscale Conversion
	$gray = 0.33R + 0.33G + 0.33B$
	Where R, G, B→ Red, green, and blue channel pixel magnitude of input image *I*
	gray→ grayscale image
	Thresholding
	Consider wd_g and ht_g Be the width and height of the gray image.
	newgray→thresholded image *i,j*→ row and column number

for i=1: ht_g do
  for j=1: wd_g do
    if gray[i,j]>=180 then
      newgray[i,j]=255
    else
      newgray[i,j]=0
    end if
  end
end
Skew Correction
Use the standard skew correction method to correct the skew of the thresholded image, yielding the final preprocessed image Ipre.
Segmentation
Plot the Horizontal (Hp) and Vertical (Vp) projection profile of preprocessed image Ipre.
Let ht_N and wd_N Be the height and width of the preprocessed image.

$$Hp_i = \sum_{j=1}^{wd\mathrm{N}} I\mathrm{pre}(\mathrm{i},\mathrm{j}) = 0 \;\; \forall\, i \in ht_N$$

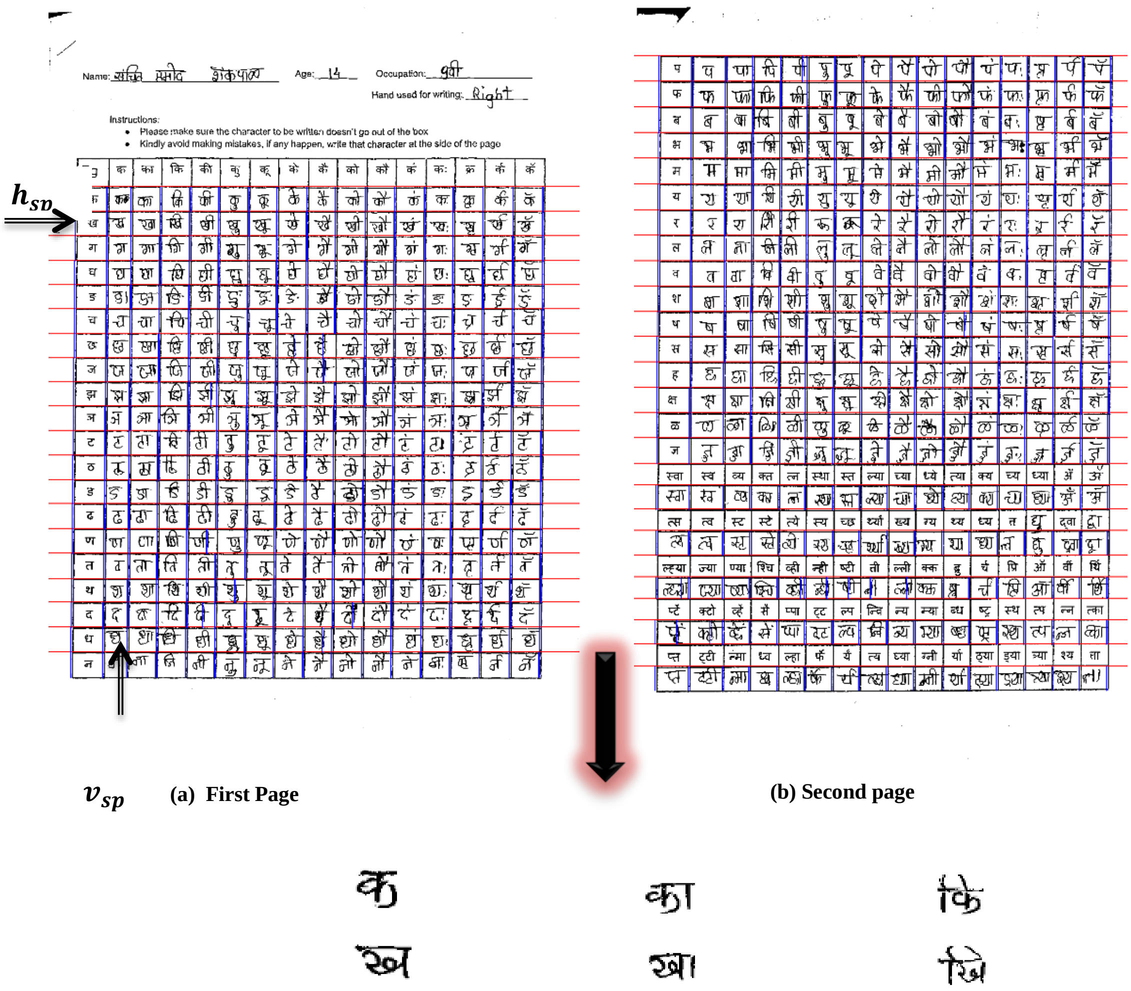
$$Vp_i = \sum_{i=1}^{ht\mathrm{N}} I\mathrm{pre}(\mathrm{i},\mathrm{j}) = 0 \;\; \forall\, j \in wd_N$$

Horizontal segmentation paths(h_{sp})
for i=1: ht_N do
  if $Hp[i] \geq \delta$wd and $i \geq f$p
    h_{sp}[i]=i
  end if
end
Where δwd→ width threshold set as 550 pixels, fp→first peak
Vertical segmentation paths(v_{sp})
for j=1: wd_N do
  if $Vp[j] \geq \delta$ht
    v_{sp}[i]=j
  end if
end
Where δht→ height threshold set as ht_N -25 pixels
Crop the characters using h_{sp} and v_{sp}
End

Table 3. Details of other standard datasets

Dataset/Author	Number of classes/Images		Modified characters	Conjunct characters
	Numerals	Characters		
DM_char /V. Dongare, V. Mankar [34]	10/5137	50/20305	Absent	Absent
CPAR /Rajiv K., Amresh K., Pervaiz A.[35]	10 / 35,000	49 / 83,300	Absent	Absent
ISI_char/U. Bhattacharya, B.Chaudhuri	10 / 22,556	49 / 30,000	Absent	Absent
DHCD/S. Acharya,P. Gyawali [36]	10 / 20,000	36/ 72,000	Absent	Absent
HMDC/S. Deore,A. Pravin	10 / 1000	48 / 4800	Absent	Absent
Proposed		580/1,03,115	Present	Present

Along with our dataset, some online available datasets are also used to test the performance of the developed network. The details of these datasets are given in Table 3.



(a) **First Page** (b) **Second page**

(c)

Fig. 2 Horizontal and vertical segmentation paths on (a) First page, (b)Second page, and (c)Sample of segmented characters.

3.2. Network Architecture

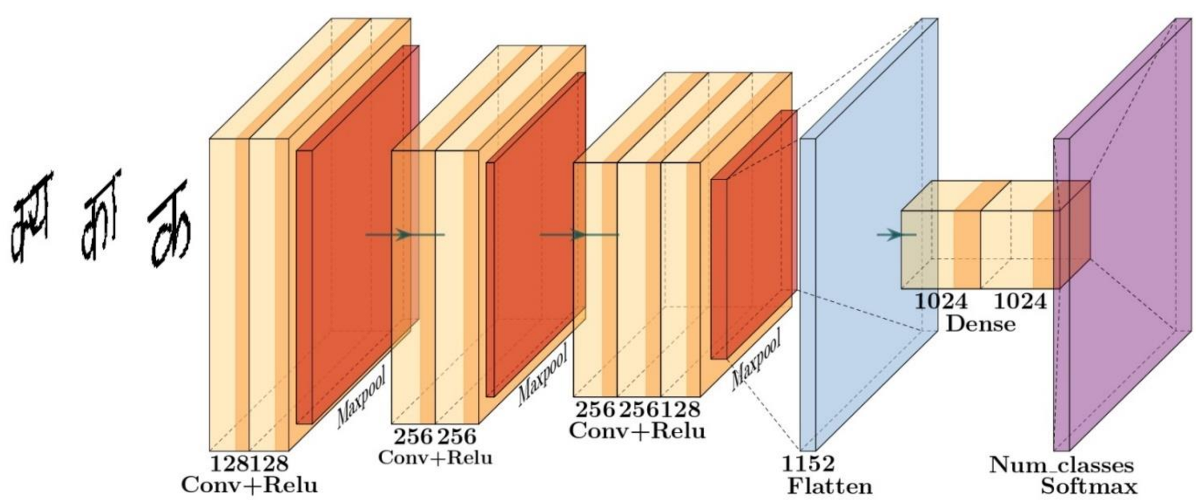
Convolutional layers are primarily responsible for feature extraction, while fully connected or dense layers handle classification tasks. Conducting experiments on the number of convolutional layers, the number of filters, the pool size, and the number of dense connections in the dense layer ultimately led to the development of three innovative models for character classification. The summary of the different network parameters for each of the three networks is presented in Table 4. The first network, referred to as SmartNet1, consists of 7 convolutional layers and 2 dense layers, each with 1024 neurons. In the second architecture, SmartNet2, the number of convolutional layers is reduced to 6, and 1x1 filters are introduced in the initial layers. Finally, SmartNet3 is developed by reducing the number of convolutional layers to 4, removing 1 dense layer, but increasing the size of the remaining dense layer to 2048 neurons. It is believed that the features extracted by SmartNet1 are more robust than those of the other two networks. However, due to its greater number of dense connections, SmartNet3 exhibits superior classification capabilities compared to SmartNet1 and SmartNet2. In addition to these three networks, two variants for each network are created by replacing the dense layers used for classification with Support Vector Machine (SVM) and k-Nearest Neighbors (kNN) models, selecting k = 7 for the kNN algorithm. This resulted in nine distinct configurations. For SVM, the Radial Basis Function (RBF) kernel, commonly known as the Gaussian kernel, is used. The reason for selecting the RBF kernel is its flexibility in handling nonlinear relationships and its effectiveness in high-dimensional spaces.

CN→ Convolutional layer
BN→ Batch Normalization layer
MP→ Max Pool layer
DP→ Dropout layer

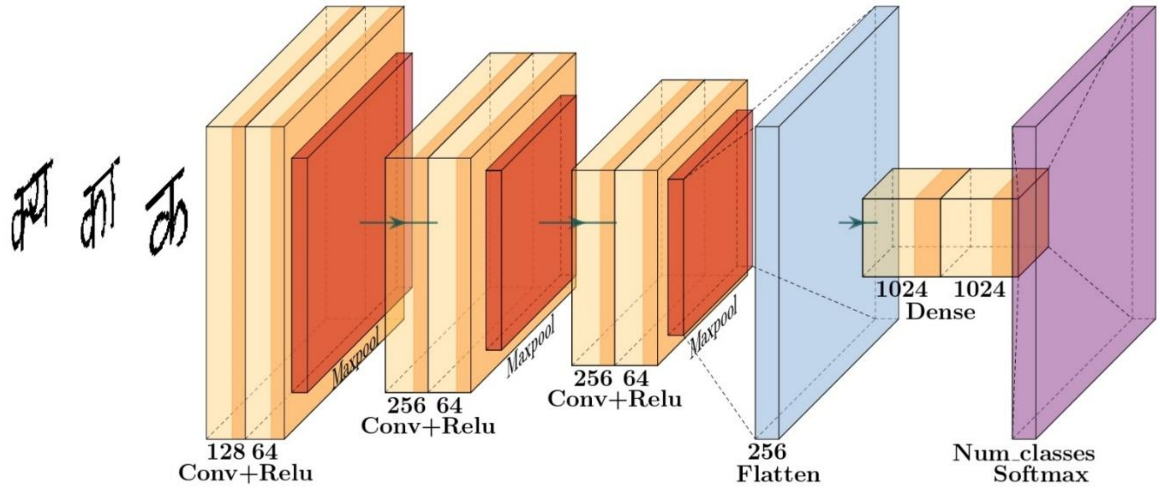
Table 4. Architectural details of 3 smartnets

	SmartNet1		SmartNet2		SmartNet3	
Layer Number	**Layer Type**	**≠ of parameters**	**Layer Type**	**≠ of parameters**	**Layer Type**	**≠ of parameters**
1	CN(7x7)	6400	CN(7x7)	6400	CN(3x3)	960
2	CN(7x7)	802944	CN(1x1)	8256	BN	384
3	BN	512	BN	256	MP(3,3)	0
4	MP(2,2)	0	MP(2,2)	0	CN(3x3)	221440
5	CN(5x5)	819456	CN(5x5)	409856	BN	1024
6	CN(5x5)	1638656	CN(5x5)	16488	MP(3,3)	0
7	BN	1024	BN	256	CN(3x3)	885120
8	MP(2,2)	0	MP(2,2)	0	BN	1536
9	CN(3x3)	590080	CN(3x3)	147712	CN(1x1)	12320
10	BN	1024	CN(1x1)	256	BN	128
11	CN(1x1)	65792	MP(2,2)	0	Flatten	0
12	BN	1024	BN	256	Dense(2048)	2361344
13	CN(1x1)	32896	Flatten	0	DP(0.5)	0
14	BN	512	Dense(1024)	263168	Softmax(49)	1188420
15	MP(2,2)	0	DP(0.5)	0		
16	Flatten	0	Dense(1024)	1049600		
17	Dense(1024)	1180672	DP(0.5)	0		
18	DP(0.5)	0	Softmax(49)	50255		
19	Dense(1024)	1049600				
20	DP(0.5)	0				
21	Softmax(49)	50255				

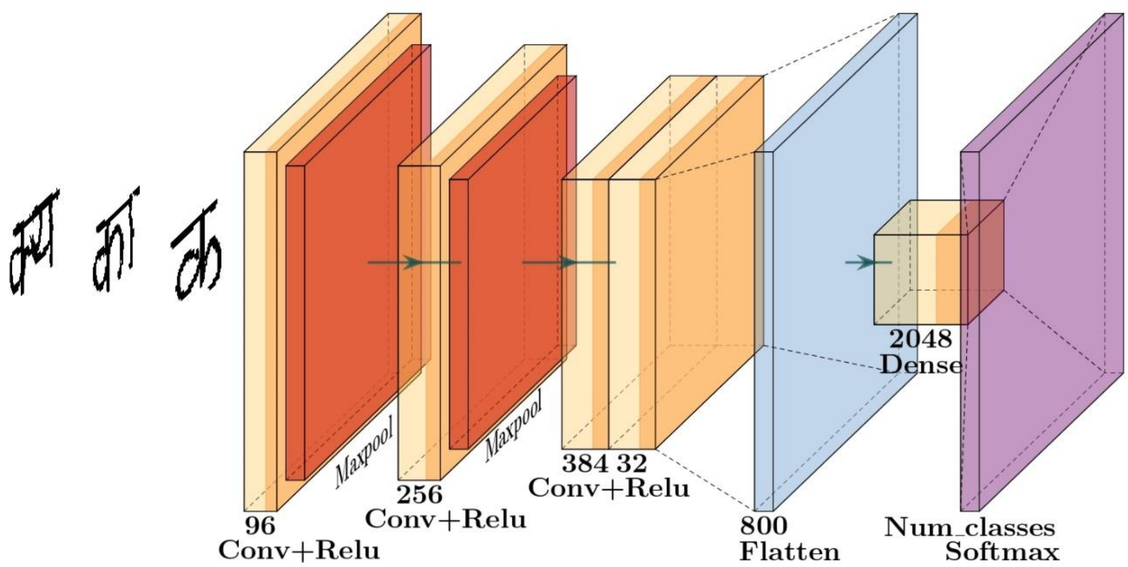
The detailed architecture of these models is illustrated in Figure 3.



(a)



(b)



(c)

Fig. 3 Different architectures of smartnet (a) SmartNet 1, (b) SmartNet 2, (c) SmartNet 3.

3.3. Experimental Setup

Exhaustive experimentation is carried out on the three developed networks and their variants. Data augmentation is applied to all datasets, and a learning rate reduction is implemented using a patience parameter. Data augmentation is used to unnaturally increase the dataset size by subjecting images to various operations, such as shifting, scaling, rotating, and flipping. This expanded dataset enables the model to be trained on various images, which helps improve generalization and reduce overfitting. The initial learning rate is fixed, but is decreased later based on a patience parameter. This helps in faster convergence and can lead to better overall performance. Additionally, the risk of overfitting during the latter part of training is mitigated due to the decreasing learning rate. More details about the experimental setup are given below.

- Train validation Test split:-60 20 20
- Optimizer used: RMSprop: Root Mean Squared propagation
- Loss function:- Categorical cross-entropy

$$Loss = -\sum_{i=1}^{num\_classes} y_i log\hat{y}_i \quad (1)$$

- Data Augmentation:-

  Rotation:-Randomly rotates the image in the range of 5°

Zoom:- 0.1, Zoom image to 10%
Width shift:- 0.1, horizontal shift in image by 10%
Height shift:- 0.1, vertical shift in image by 10%
Horizontal Flip:- Randomly flips the image

- Reduce Learning rate:-
  Monitor:- Validation loss
  Patience parameter:- 5
  Factor:-0.1

The training is carried out for 30, 50, and 100 epochs to determine the optimal number of epochs and achieve the best accuracy.

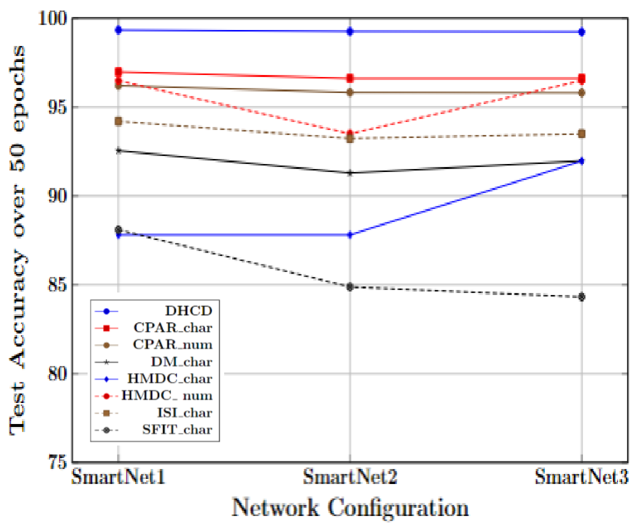
The Models are executed in GPU runtime using Keras and TensorFlow framework. Google Colaboratory Pro, a paid cloud service, is used to perform the experiments. It offers 16 GB NVIDIA Tesla with GDDR6 memory.

4. Results and Discussion

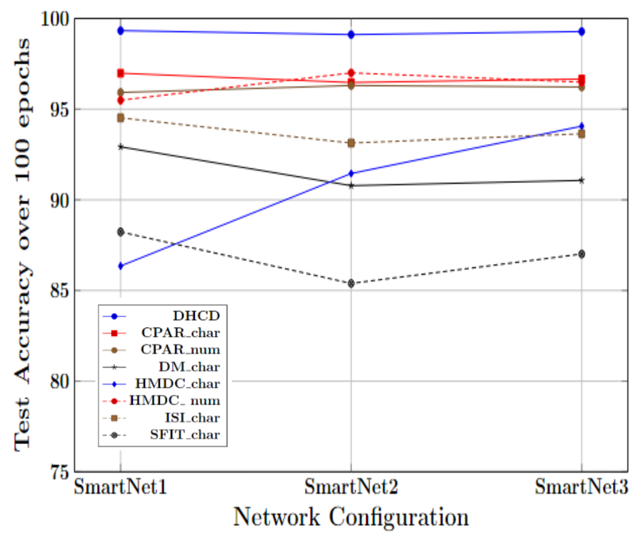
To assess network performance, Test Accuracy (TA) is used as the performance metric. The formula for the same is given below.

$$TA = \frac{Accurately\ identified\ Charaters}{Total\ Number\ of\ Characters} \times 100 \qquad (2)$$

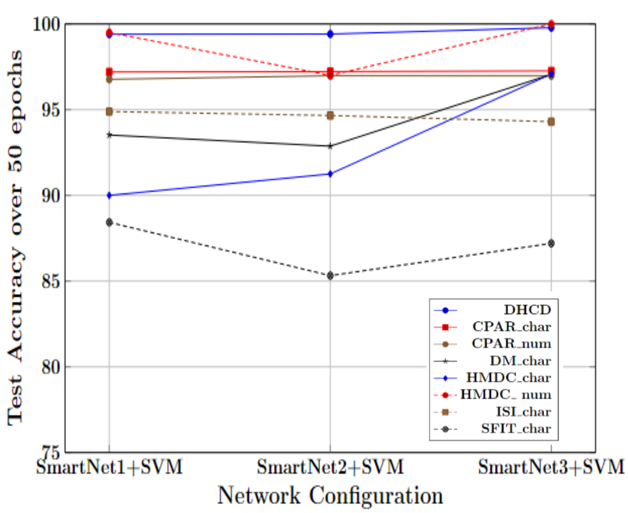
All three networks, along with 2 additional SVM and kNN variants, are trained for 30, 50, and 100 epochs. All the test accuracies are recorded, and it is observed that for each network, the maximum accuracy is typically achieved at 100 epochs. After 100 epochs, no improvement in accuracy is seen. The graphs in Figure 4 correspond to test accuracies of 50 and 100 epochs.



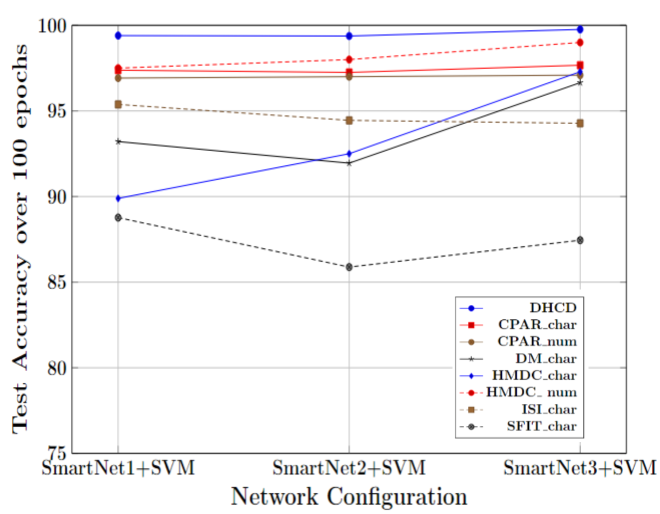
(a)



(b)



(c)



(d)

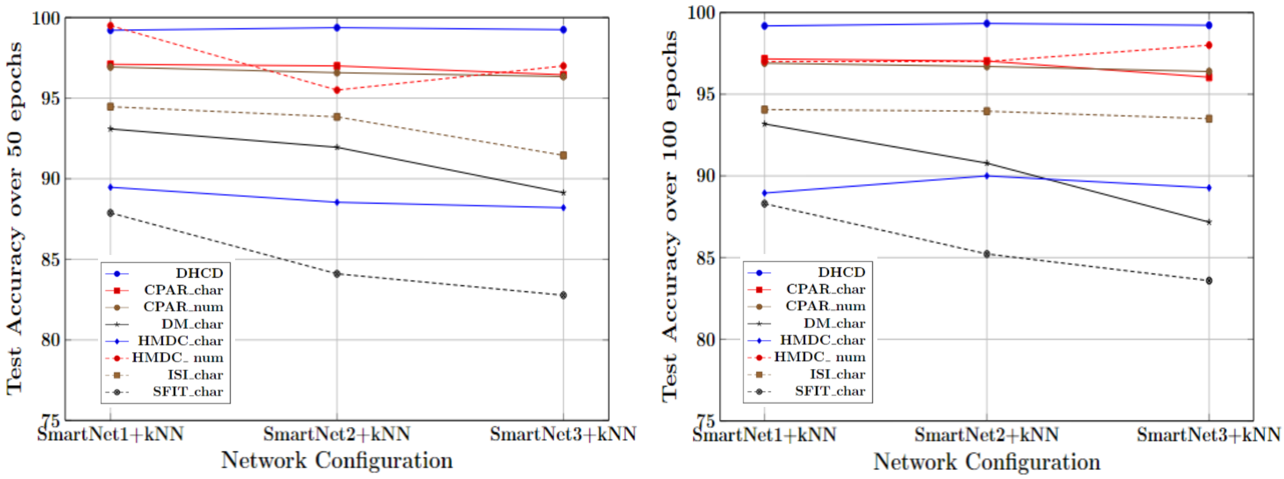


Fig. 4 (a)-(f) Test accuracies of different network configurations OVER 50 and 100 epochs

It can be observed from Figure 4(a) and (b) that SmartNet1 performance is best compared to SmartNet2 and SmartNet3 across all datasets. On the contrary, when it comes to their SVM variants, Figure 4(c) and (d) shows that SmartNet3 with its SVM variant is best compared to the SVM variants of SmartNet2 and SmartNet1. In the kNN variant corresponding to Figure 4(e) and (f), it can be seen that SmartNet1 with its kNN variant is best. The top 2 accuracies across different datasets, along with the experimental parameters, are shown in Table 5, indicating that SmartNet3+SVM and SmartNet1+SVM perform excellently compared to other configurations. Table 6 presents a comparison of three basic SmartNets based on the number of convolutional layers, total parameters, number of dense parameters, and training time for the DHCD dataset, trained over 100 epochs. It can be observed that SmartNet1 has the maximum number of convolutional layers, so its feature extraction capability is better than that of other configurations, but its percentage of dense parameters, which are responsible for classification, is lower than that of other configurations. In SmartNet2 and 3, the number of convolutional layers is fewer, but the dense parameters are more. The images to be classified have different strokes depending on the shape of the character, displayed on a plain white background, which implies fewer features. To correctly classify these features, a more robust classifier with denser connections is required. Therefore, SmartNet3 appears to be performing better than 1 and 2.

Table 5. Top 2 accuracies for different datasets

Dataset	Network	≠ of Epochs	Test Accuracy
DHCD	SmartNet3+SVM SmartNet3+SVM	50 100	99.78 % 99.76 %
CPAR_char	SmartNet3+SVM SmartNet1+SVM	100 100	97.66 % 97.38 %
CPAR_num	SmartNet3+SVM SmartNet2+SVM	100 100	97.09 % 97.00 %
DM_char	SmartNet3+SVM SmartNet3+SVM	50 100	97.06 % 96.65 %
HMDC_char	SmartNet3+SVM SmartNet3+SVM	100 50	97.29 % 97.08 %
HMDC_num	SmartNet3+SVM SmartNet3+SVM	50 100	100 % 99 %
ISI_char	SmartNet1+SVM SmartNet1+SVM	100 50	95.38% 94.89%
SFIT_char	SmartNet1+SVM SmartNet1+SVM	100 50	88.77% 88.43%

Table 6. Model complexity of three networks

Network	≠ of convolutional layers	Total parameters	Dense parameters	% of Dense parameters	Train time for 100 epochs
SmartNet1	7	6237742	2277422	36.5	1 hour
SmartNet2	6	1966062	1312768	66	43 min
SmartNet3	4	3578510	245598	68	35 min

Table 7. Comparative analysis of test accuracies of various techniques

Reference/Year	Dataset	Methodology	Experimental Setup	Test Accuracy
[19] 2018	CPAR_char	Geometrical shape-based features with kNN classifier		94.36%
Proposed Method		SmartNet1		96.98%
		SmartNet1+SVM	Number of Epochs:	97.38%
		SmartNet1+kNN	50-100	97.16%
		SmartNet2		96.62%
		SmartNet2+SVM		97.25%
		SmartNet2+kNN		97.03%
		SmartNet3		96.66%
		SmartNet3+SVM		97.67%
		SmartNet3+kNN		96.45%
[25] 2018	DHCD	CNN-based features, different CNN models with 8 and 4 convolutional layers	Number of Epochs: Not Reported	Maximum Accuracy 96.9%
[37] 2020		CNN-based features, two different CNN architectures, Model A:- 3 Convolutional layers Model B:- 2 Convolutional layers (Epochs=50)		Model A: 98.47% Model B: 98.13%
[27] 2020		CNN-based features, two-stage VGG 16 architecture, Devanagari Handwritten Character Recognition System (DHCRS) model	Train-Test split: 85:15 Two-stage training with 20 and 10 epochs	Characters: 97.80% Numerals: 99.40%
[32] 2024		CNN-based features, VGG 16 architecture	No of Epochs: 300	96.58%
[31] 2024		CNN-based features, Generator-Discriminator model	No of Epochs: 100	98.86%(Augmented)
[33] 2025		CNN-based features, Modified capsnet	Train-Test split: 85:15 No of Epochs: 15	99.30%
Proposed Method		SmartNet1	Train validation Test split:-60 20 20 Number of Epoch: 50-100	99.34%
		SmartNet1+SVM		99.40%
		SmartNet1+kNN		99.21%
		SmartNet2		99.26%
		SmartNet2+SVM		99.41%
		SmartNet2+kNN		99.38%
		SmartNet3		99.28%
		SmartNet3+SVM		99.78%
		SmartNet3+kNN		99.25%
[24] 2018	ISI_char	CNN-based features 6 Networks with 2 to 3 convolutional	No of Epochs: 100	94.27%

		layers,100 nodes in a fully connected layer		
Proposed Method		SmartNet1+SVM	Number of Epoch: 50-100	95.38%
		SmartNet1+kNN		94.47%
		SmartNet2+SVM		94.66%
		SmartNet3+SVM		94.30%
[27] 2020	HMDC	CNN-based features, two-stage VGG 16 architecture, Devanagari Handwritten Character Recognition System (DHCRS) model	Train-Test split: 80:20 Two-stage training with 20 and 10 epochs	96.55%
Proposed Method	HMDC_char	SmartNet3+SVM	Number of Epoch: 50-100	97.08 %
	HMDC_num	SmartNet3+SVM		100 %
		SmartNet2+kNN		98 %

The performance of the developed networks is compared with other state-of-the-art architectures in Table 7. In [25], two architectures are used, one with 8 and the other with 4 CNN layers. In [27] and [32], the VGG16 architecture is used, and in [27], its DHCRS variant is also used. In [31], a complex generator discriminator model is used. All the mentioned models are much more complex than the proposed networks in terms of architecture. From the design perspective, the inception of 1x1 convolutional layers in all three architectures helped in reducing the dimensionality and adding non-linearity, which helped in learning complex features. The designed networks are a result of experimentation on convolution layers, dense layers, filters, and pooling size; therefore, they inherently exhibit better feature extraction as well as classification capability. The inherent characteristics of SVM, like its effectiveness in handling high-dimensional data, lesser susceptibility to overfitting, and capability of performing well even with lesser training samples, created an impact on the overall performance of the networks. During the training phase, a reduction in the learning rate helped in fine-tuning the networks, thereby improving convergence. The developed networks are designed to provide greater accuracy with fewer trainable parameters and also converge faster. It is observed that the SmartNets developed increase test accuracy by 2 to 3% across various datasets.

5. Conclusion and Future Scope

Developing an Optical character recognizer for the Devanagari script is difficult due to variations in writing styles and the presence of modifiers and conjunct characters. Altogether, nine configurations are presented, capable of recognizing basic, modified, and complex conjunct characters. The SFIT_char dataset has been developed and is the only dataset so far available with 36 classes of consonants, 465 classes of modified characters, and 79 classes of conjunct characters. The developed networks are tested over five standard datasets. For the DHCD, CPAR_char, CPAR_num, and DM_char datasets, SmartNet3+SVM achieves maximum accuracies of 99.78%, 97.66%, 97.09%, and 97.06%, respectively. A maximum accuracy of 95.38% for ISI char by SmartNet1+SVM, 100% for HMDC_num, and 97.29% for HMDC_char by SmartNet3+SVM is achieved. On the SFIT_char dataset, SmartNet1+SVM achieves a test accuracy of 88.77 %. The developed SmartNets, though comparatively shallow, are found to perform better than other state-of-the-art deep networks and are boosting the test accuracy by 2 to 3 % across different standard datasets. In the future, the performance of the proposed networks will be tested on other Indic scripts using different ensemble approaches. It is believed that ensemble techniques improve test accuracy; therefore, work on averaging, bagging, and stacking approaches can be carried out.

Data Availability Statement

The developed Handwritten Devanagari character dataset is publicly available at https://www.kaggle.com/datasets/pallaviypatil/sfit-dataset.

References

[1] Somnath Chatterjee et al., "City Name Recognition for Indian Postal Automation: Exploring Script Dependent and Independent Approach," *Multimedia Tools and Applications*, vol. 83, pp. 22371-22394, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Ujjwal Bhattacharya, and B.B. Chaudhuri, "Handwritten Numeral Databases of Indian Scripts and Multistage Recognition of Mixed Numerals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 444-457, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[3] Jamshed Memon et al., "Handwritten Optical Character Recognition (HOCR): A Comprehensive Systematic Literature Review," *IEEE Access*, vol. 8, pp. 142642-142668, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Ismet Zeki Yalniz, and R. Manmatha, "Dependence Models for Searching Text in Document Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 49-63, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[5] Chetan Ralekar et al., "Intelligent Identification of Ornamental Devanagari Characters Inspired by Visual Fixations," *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, Sydney, NSW, Australia, pp. 14-19, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Diptee Vishwanath Chikmurge, and Shriram Raghunathan, "Enhancing Marathi Handwritten Character Recognition Using Ensemble Learning," *Signal Processing*, vol. 40, no. 1, pp. 327-334, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Sandeep Dwarkanath Pande et al., "Digitization of Handwritten Devanagari Text using CNN Transfer Learning – A Better Customer Service Support," *Neuroscience Informatics*, vol. 2, no. 3, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Srinidhi Karthikeyan et al., "An OCR Post- Correction Approach Using Deep Learning for Processing Medical Reports," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 5, pp. 2574-2581, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Hubert et al., "Classifying Promotion Images Using Optical Character Recognition and Naïve Bayes Classifier," *Procedia Computer Science*, vol. 179, pp. 498-506, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Senka Drobac, and Krister Lindén, "Optical Character Recognition with Neural Networks and Post-Correction with Finite State Methods," *International Journal on Document Analysis and Recognition*, vol. 23, pp. 279-295, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Noor Masrur et al., "Dissimilarity Matrix: Bridging the Gap of Bangla OCR Error Correction through a Novel Approach," *2023 5th International Conference on Sustainable Technologies for Industry 5.0 (STI)*, Dhaka, Bangladesh, pp. 1-6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Zhenrong Zhang et al., "Bengali Handwritten Grapheme Recognition Using Cutmix-Based Data Augmentation," *Proceedings of the 3rd World Symposium on Software Engineering*, pp. 145-149, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] Minesh Mathew, Mohit Jain, and C.V. Jawahar, "Benchmarking Scene Text Recognition in Devanagari, Telugu and Malayalam," *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Kyoto, Japan, pp. 42-46, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[14] Vrushali. T. Lanjewar, and R.N. Khobragade, "Transfer Learning using Pre-Trained AlexNet for Marathi Handwritten Compound Character Image Classification," *2021 International Conference on Intelligent Technologies (CONIT)*, Hubli, India, pp. 1-7, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Madhuri Yadav, Ravindra Kumar Purwar, and Mamta Mittal, "Handwritten Hindi Character Recognition: A Review," *IET Image Processing*, vol. 12, no. 11, pp. 1919-1933, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[16] Meduri Avadesh, and Navneet Goyal, "Optical Character Recognition for Sanskrit Using Convolution Neural Networks," *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, Vienna, Austria, pp. 447-452, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[17] Arti Khaparde, Vaidehi Deshmukh, and Manisha Kowdiki, "Enhanced Nature-Inspired Algorithm-Based Hybrid Deep Learning for Character Recognition in Sanskrit Language," *Sensing and Imaging*, vol. 24, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Dhruv Kudale et al., "Textron: Weakly Supervised Multilingual Text Detection through Data Programming," *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2871-2880, 2024. [Google Scholar] [Publisher Link]

[19] Parul Sahare, and Sanjay B. Dhok, "Multilingual Character Segmentation and Recognition Schemes for Indian Document Images," *IEEE Access*, vol. 6, pp. 10603-10617, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[20] Sonika Rani Narang, M.K. Jindal, and Munish Kumar, "Ancient Text Recognition: A Review," *Artificial Intelligence Review*, vol. 53, pp. 5517-5558, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Amar Jindal, and Rajib Ghosh, "A Hybrid Deep Learning Model to Recognize Handwritten Characters in Ancient Documents in Devanagari and Maithili Scripts," *Multimedia Tools and Applications*, vol. 83, pp. 8389-8412, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[22] Anuradha Choudry, Intermediate Level of Spoken Sanskrit, Swayam - NPTEL, 2018. [Online] Available: https://onlinecourses.nptel.ac.in/noc19_hs53/preview

[23] Partha Pratim Roy et al., "HMM-based Indic Handwritten Word Recognition using Zone Segmentation," *Pattern Recognition*, vol. 60, pp. 1057-1075, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[24] Mahesh Jangid, and Sumit Srivastava, "Handwritten Devanagari Character Recognition Using Layer-wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods," *Journal of Imaging*, vol. 4, no. 2, pp. 1-14, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[25] Shrawan Ram, Shloak Gupta, and Basant Agarwal, "Devanagri Character Recognition Model using Deep Convolution Neural Network," *Journal of Statistics and Management Systems*, vol. 21, no. 4, pp. 593-599, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[26] Sonika Narang, M.K Jindal, and Munish Kumar, "Devanagari Ancient Documents Recognition Using Statistical Feature Extraction Techniques," *Sādhanā*, vol. 44, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[27] Shalaka Prasad Deore, and Albert Pravin, "Devanagari Handwritten Character Recognition using Fine- Tuned Deep Convolutional Neural Network on Trivial Dataset," *Sādhanā*, vol. 45, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[28] Duddela Sai Prashanth et al., "Handwritten Devanagari Character Recognition Using Modified Lenet and Alexnet Convolution Neural Networks," *Wireless Personal Communications*, vol. 122, pp. 349-378, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[29] Snehal S. Gaikwad, S.L Nalbalwar, and A.B. Nandgaonkar, "Devanagari Handwritten Characters Recognition using DCT, Geometric and Hue Moments Feature Extraction Techniques," *Sādhanā*, vol. 47, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[30] Danveer Rajpal, and Akhil Ranjan Garg, "Deep Learning Model for Recognition of Handwritten Devanagari Numerals with Low Computational Complexity and Space Requirements," *IEEE Access*, vol. 11, pp. 49530-49539, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[31] Khushi Sinha, and Richa Gupta, "Enhancing Handwritten Devanagari Character Recognition via GAN-Generated Synthetic Data," *Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing*, pp. 484-489, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[32] Chetan Sharma et al., "Advancements in Handwritten Devanagari Character Recognition: A Study on Transfer Learning and VGG16 Algorithm," *Discover Applied Sciences*, vol. 6, pp. 1-15, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[33] Bharati Yadav, Ajay Indian, and Chandra Prakash Gupta, "Offline Handwritten Devanagari Character Recognition Using Modified Capsule Neural Network," *Procedia Computer Science*, vol. 258, pp. 2878-2887, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[34] Vikas J. Dongre, and Vijay H. Mankar, "Development of Comprehensive Devnagari Numeral and Character Database for Offline Handwritten Character Recognition," *Applied Computational Intelligence and Soft Computing*, vol. 2012, no. 1, pp. 1-5, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[35] R. Kumar, A. Kumar, and P. Ahmed, "A Benchmark Dataset for Devnagari Document Recognition," *Recent Advances in Electrical Engineering Series*, pp. 258-263, 2013. [Google Scholar]

[36] Shailesh Acharya, Ashok Kumar Pant, and Prashnna Kumar Gyawali, "Deep Learning based Large Scale Handwritten Devanagari Character Recognition," *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, Kathmandu, Nepal, pp. 1-6, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[37] Anuj Bhardwaj, and Ravendra Singh, "Handwritten Devanagari Character Recognition Using Deep Learning - Convolutional Neural Network (CNN) Model," *PalArch's Journal of Archaeology of Egypt / Egyptology*, vol. 17, no. 6, pp. 7965-7984, 2020. [Google Scholar] [Publisher Link]