

SHALAKA PRASAD DEORE 

## A DHCR\_SMARTNET: A SMART DEVANAGARI HANDWRITTEN CHARACTER RECOGNITION USING LEVEL-WISED CNN ARCHITECTURE

**Abstract** *Handwritten script recognition is a vital application of the machine-learning domain. Applications like automatic license plate detection, pin-code detection, and historical document management increases attention toward handwritten script recognition. English is the most widely spoken language in India; hence, there has been a lot of research into identifying a script using a machine. Devanagari is a popular script that is used by a large number of people on the Indian subcontinent. In this paper, a level-wised efficient transfer-learning approach is presented on the VGG16 model of a convolutional neural network (CNN) for identifying isolated Devanagari handwritten characters. In this work, a new dataset of Devanagari characters is presented and made accessible to the public. This newly created dataset is comprised of 5800 samples for 12 vowels, 36 consonants, and 10 digits. Initially, a simple CNN is implemented and trained on this new small dataset. During the next stage, a transfer-learning approach is implemented on the VGG16 model, and during the last stage, the efficient fine-tuned VGG16 model is implemented. The obtained accuracy of the fine-tuned model's training and testing came to 98.16% and 96.47%, respectively.*

**Keywords** convolutional neural network, VGG16, fine-tuned, handwritten script, Devanagari characters

**Citation** Computer Science 23(3) 2022: 301–320

**Copyright** © 2022 Author(s). This is an open access publication, which can be used, distributed and reproduced in any medium according to the Creative Commons CC-BY 4.0 License.

## 1. Introduction

India is a land of diversity, with several languages and cultures in one place. In India, Devanagari is the most commonly used script; it consists of 120 languages, which makes it the most adapted writing system [21]. Devanagari serves as the base language to many languages, such as Marathi, Hindi, Sanskrit, Gujarati, and Nepali. Devanagari has evolved over a period of 2000 years and is highly continuous in nature. Speaking about technology in the computational genre the introduction of GPUs has made substantial improvement. There has been considerable growth in data sources and improvement in computational power, creating a scope for new methodologies that recognize handwriting in different languages. This gives us an opportunity to develop language-processing models for Devanagari (being a compound script-based language). Word recognition is one of the applications that deal with recognizing handwritten or printed words in text. The text is scanned and then converted to a machine-editable format. With the printed word, handwritten word recognition is also gaining more interest nowadays. Many research works that are related to handwritten word recognition have been noted in various scripts like Japanese [16], Chinese [36], etc. The field is comprised of both artificial intelligence and image processing. The recognition process is comprised of two categories: online recognition, and offline recognition [22]. Online recognition deals with recognizing real-time acquisition; it typically uses an optical pen for drawing on a screen, and a recognized word is displayed on the screen. This approach is mainly used in real-time environments for handwriting recognition. The second approach is offline recognition; this deals with recognizing text that is written on a sheet. The sheet is scanned through a digital device like a camera or scanner, and the image is stored in the system and, hence, recognized. The main advantage of the offline method is that it can be done at any time, as the images are scanned and stored.

For any text recognition, an input image goes through different phases to produce the output [11]; Figure 1 depicts these different phases. The first phase is data generation/collection; this phase involves the collection of data from various sources. This data can be collected in two ways: online, or offline. In this work, a Devanagari handwritten character data set has been newly created and is publicly available for research. The second stage is data pre-processing, where noise is eliminated from the data. All of the necessary operations are performed on the input data (binarization, sampling, noise reduction, thinning, smoothing, etc.) in order to produce clean data.

The segmentation phase deals with dividing a text into parts or zones in order to recognize each part individually. The segmentation task is very difficult for handwritten characters that are touching each other. In Devanagari, the script consists of various character sets that include vowels and consonants as well as compound and composite characters. One horizontal line is present in each character (called *shirorekha*); this produces a problem in the segmentation phase and decreases the identification accuracy (as explained by Sonkusare et al. [32]). These authors also presented various segmentation techniques along with their efficiency. Based on the

seed pixel that is chosen among the candidate pixels, the segmentation facilitate feature technique was presented by Kohli et al. [13] in order to discover a connective way to separate any touched components. In this proposed method, three neighboring pixels are considered, and the connection link between the touching characters are identified for separating them. Containing two touching consonant words achieved the highest accuracy of 96.2%. The importance of a zonal-based segmentation methodology that can be used for Indic language recognition was explained in [3, 25]. Here, the image is split into several zones; then, each zone is recognized individually. The authors found that the zonal methodology was more efficient than the traditional one. Feature extraction is a very important stage and plays an important role in classification. Extracting good features always helps to gain better accuracy. Features that are extracted manually result in a very time-consuming task. In our work, a deep-learning approach is explored to extract features automatically, which is a faster method that works directly on raw pixel data. The last phase consists of classification, recognizing an input image, and mapping it to an output class.



**Figure 1.** Text recognition phases

More than 300 million individuals use the Devanagari script for maintaining their records on the Indian subcontinent [9]. Being an important script in India, the work for digitizing Devanagari-based languages is comparatively minimal. There is considerably less available research on Indian languages when compared to the number of people that interact in them. Recognizing a handwritten script comes under the applications of character recognition, which deals with recognizing various kinds of handwritten/printed characters (such as digits, cursive scripts, symbols, and touch characters).

In [24], the author presented a Chinese text-recognition model that consisted of three layers. A new layer was introduced as a feature extractor that was combined with the residual network to gain the advantages of more-accurate identification. In [12], the authors developed a dataset of Malayalam handwritten words and trained their model using a deep convolutional neural network (CNN) architecture; here, the hybrid approach was implemented where CNN was used for extracting features, and a support vector machine (SVM) was implemented for their identification. In [18], research was performed on the recognition of ancient Devanagari documents; statistical features like open endpoints, centroid, horizontal, and vertical peak levels, and intersection points were extracted from the images. A random forest multi-layer

perceptron (MLP) neural network, SVM with an RBF kernel, and CNN classifiers were explored in this work, and all of the results were compared. A feature of data plays an important role in recognition. Histogram of oriented gradient-based local features were presented in [4, 5]; this HOG approach was demonstrated to be an efficient method for extracting the curvature features of Devanagari characters. Dutta et al. [6] presented a lexicon-free Bangla and Devanagari word-recognition system. The performance of the system was evaluated using a hybrid model that was based on CNN-RNN. The results were evaluated on both lexicon-free and lexicon-based datasets. The model included various layers: a spatial transformer, residual convolutional blocks, and bi-directional LSTM. In [34], the author proposed the word-level recognition that was present in scenes; basically, the characters of a word in a scene touch each other, making their recognition difficult. So, the author used a segmentation-free approach; in this, an image is converted into a sequential signal, then it is passed to a recurrent neural network (RNN) for recognition. The RNN technique is then combined with the LSTM technique to gain a good result; however, the system provides very poor results for curvy or distorted images. For classification, identifying the features of an image is crucial. Selecting the appropriate features would improve the results; however, this is lengthy and complex method. In a deep-learning approach, this works directly on raw images and automatically extracts the required information from an image; this helps us improve the accuracy while devoting less time [14].

Deep neural networks have received significant attention because of their applicability for developing various applications in different areas (such as recommendation systems, text identification, stock prediction, and audio recognition) [15]. Alom et al. [2] evaluated the application of a different CNN for recognizing handwritten Bangla characters. A layer-wise deep-CNN approach was proposed in [10] in order to improve the recognition of Devanagari characters over a standard CNN. Based on the permutations of the convolutional-pooling layers and neurons, the author presented six architectures of DCNN here; this resulted in an accuracy of 96.45%. In paper [8], author represented the use of various convolutional blocks for designing a residual network with feedback. Using many convolutional blocks, it is very easy to design deep convolutional models. With this concept, the author achieved vast improvements in segmentation, detection, and classification. Guha et al. [7] compared various CNN models with respect to the parameters that were required for the training, memory space, and execution time. The authors primarily concentrated on developing parts of the model in order to create a model that was effective in terms of space and time. The proposed model was evaluated on publicly available datasets in this paper, and the findings were positive. The model was easy in design. Since it had fewer layers, it took less time to train. A handwritten page-level dataset (PHDIndic.11) of 11 popular Indic scripts was created by the authors of [20]. With script identification, the authors stated that the generated dataset could also be useful for a number of applications, including understanding scripts, recognizing a page writer, word recognition, etc. Santosh and Wendling [29] presented a novel algorithm called dynamic time warping (DTW). Here, the features were extracted

for each projection using Radon transformation, and the feature pairs were matched using the DTW algorithm. The DTW algorithm avoids a loss of data because there is no need to convert the feature vector into one vector. The proposed method can handle any type of image that includes defects, but its computational cost is high.

Based on a literature study, deep learning stands out as the most effective technique for obtaining encouraging results on image classification. Even scripts such as Bangla [23], Roman (MNIST) [19], and Arabic [37] have been effectively recognized using deep-learning approaches. In a deep-learning approach, hyper-parameter selection is the most difficult problem to solve and necessitates much research. This raises a number of questions concerning CNN's architecture for the problem of handwriting recognition: How could CNN improve its ability to extract numerous features from handwritten characters? What impact do different hyper-parameters have on CNN's success? What effect do design parameters have in CNN efficiency optimization? Hence, this research attempts to answer these questions based of the research gap that was identified in the literature study. The contributions of this research are as follows:

1. corpus creation: constructed isolated Devanagari handwritten character dataset;
2. level-wised model: implemented level-wised Devanagari handwritten character-recognition (DHCR) model (first stage – simple CNN model is implemented; second stage – model is implemented using pre-trained VGG16 network; finally fine-tuning done on several layers (top) of pre-trained network);
3. several deep-learning models explored on our newly created dataset, and our fine-tuned model also executed on standard datasets like UCI, CMATERdb, etc.

Furthermore, Section 2 describes the features of the Devanagari script. The creation of the dataset and the proposed work are discussed in Sections 3 and 4, respectively. The experimental protocol and results are explained in Section 5. The conclusion of this paper is given in Section 6.

## **2. Features of Devanagari script**

The Devanagari script was initially developed to write the Sanskrit language; later on, it was extended to develop other Indian languages (Hindi, Marathi, Gujarati, etc.). Devanagari represents a phonetic script; this means that each character is pronounced in exactly the way that it is written. The left-to-right direction is followed for writing Devanagari. Each word is separated by a line that exists at the top; this is known as "Shirorekha." Devanagari is an alphasyllabary writing system; such systems are based on consonants in which the vowels are requisite yet secondary. It is a syllabic script; this means that Devanagari characters are written as combinations of vowels and consonants where a vowel can be added to a consonant in the form of character or in the form of a modifier.

The Devanagari script is comprised of 36 consonants, 12 vowels (with 12 modifiers), and 10 digits (as depicted in Figure 2). In Figure 2, a) describes a handwritten

sample of the vowels and consonants, and b) depicts the digits. The vowels can be written as individual characters or can be combined with consonants in the form of modifiers. For instance, the vowel "अ" is in its independent form; with the consonant "प," it can be written in a combined form ("पा") where the vertical line at the end is a modifier. Being a phonetic script, Devanagari allows for the continuation of a consonant to another as a half character. This joined character is called a composite character and plays significant factors in recognition. Due to this composite, character recognition becomes more complex. There are several other factors that contribute to a recognition system's complexity and difficulty; these reasons are as follows: i) different handwriting styles (as shown in Figure 3); ii) similarly shaped characters (as shown in Figure 4); iii) noise present while collecting data; and iv) variations that are present in a person's handwriting under various conditions.

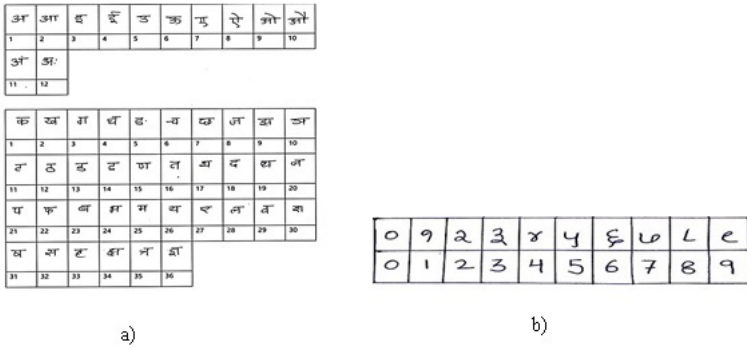


Figure 2. Handwritten sample: a) vowel & consonant characters; b) digits

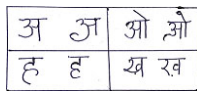


Figure 3. Different writing styles of same character

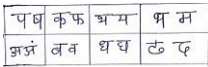


Figure 4. Similarly shaped characters

### 3. Dataset corpus creation

In the field of pattern identification, corpus creation is a rather vital task. Better-quality data helps us gain good accuracy and improves the recognition rate of a sys-

tem. For the corpus creation, fixed-sized papers were distributed to people of different age groups to write isolated characters using a ball-point pen; then, all of these handwritten characters were scanned separately. Constraints like the age of the person, the quality of the paper, the writing style, and the type of pen was not kept while taking the samples. Samples were collected from 50 people for 58 different classes (these consisted of 12 vowels, 36 consonants, and 10 numerals). The size of each image was  $1600 \times 1600$  pixels, and they were pre-processed and saved using the .jpg format. Each scanned image was named with a class name as well as an order number; for example, "च" was the 18<sup>th</sup> character, so it was labeled "C18\_1.jpg, C18\_2.jpg."

## 4. Proposed methodology

### 4.1. Pre-processing and data augmentation

This stage came after the data collection. The data that was created had variations and noise; hence, the images required pre-processing. For our created dataset, the characters were written on paper using a blue pen, and then all of the images were scanned separately. All of the images were converted into grayscale images. The XnConvert batch processing tool (which consists of many pre-processing operations) was applied to the dataset. The various filters that were used and their output is shown in Figure 5:

1. minimum filter increases thickness of each character;
2. resizing is done to enlarge or reduce size of image into some specific size (like  $64 \times 64$ );
3. zealous crop is content-aware crop that slices image through all boundaries until pixel at which content is found;
4. curve pre-processing operation changes color channels of image to enhance specific features.



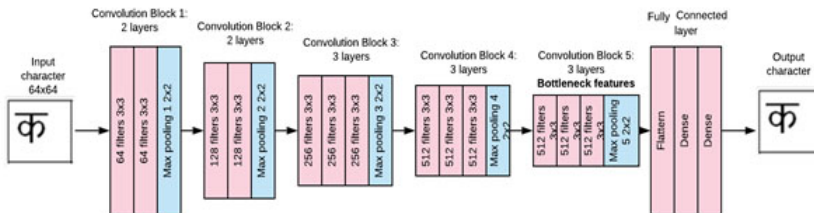
**Figure 5.** Pre-processing output

To avoid overfitting the model, data augmentation was performed. It was also used to enlarge the existing small dataset by performing various data-augmentation operations. This paper uses the ImageDataGenerator class of Keras for real-time data augmentation. It accepts an original batch of input images, applies random transformation, and generates a new batch of images for training. The model will get a new set of images for training each time, so it avoids the overfitting problem on our dataset.

## 4.2. Architecture of VGG16

The convolution neural network (CNN/ConvNet) is a multi-layer fully connected network that automatically extracts features from input images [23]. It extracts a sufficient number of features from the images, so it reduces the job of manual feature extraction. Basically, it introduces a large number of datasets for training to provide better accuracy. However, ConvNet also has the capability of learning features from a small dataset by using a pre-trained model like ImageNet. The ImageNet dataset consists of 14 million images that belong to 1000 different classes. Based the depth of the layers, the VGG architecture is classified in the VGG11, VGG16, and VGG19 architectures. The VGG16 model attained a top-5 test accuracy on the ImageNet dataset [31]. To balance the computational cost after increasing the depth of the network, VGG uses reduced convolutional filters of a size of  $3 \times 3$  and fewer field channels [26]. The architecture of the VGG19 network is the same as VGG16 except for its number of layers; it has a total of 19 weight layers. The VGG16 architecture is illustrated in Figure 6 and consists of the following three main layers:

- A) convolution layer: this is very important layer that is used for feature extraction from input images – it uses different sizes of filters for this purpose;
- B) pooling layer: this is mainly used to decrease dimensions of data (it is also called down sampling) – pooling carried out by using filter of size of  $2 \times 2$ ; most regularly used pooling methods are maximum (max) and average (avg) pooling;
- C) fully connected layer: the purpose of this layer is to identify images – flattened vector is sent to fully connected layer for identification; it uses principle of multi-layer perceptron.



**Figure 6.** Architecture of VGG16 pre-trained network

The learning rate (LR) is very important for training the model; it determines the weight modification in each network layer. A high LR diverts the network to achieve fewer errors, where a low LR requires more time to reduce errors; hence, adaptive gradient optimizers are the best choice for training a network faster. The proposed DHCR model is trained by using one of the faster adaptive learning optimizers – RMSprop (root mean square propagation). This automatically adjusts the learning rate of each parameter separately.



### 4.3. Architectures of proposed level-wised DHCR model

This paper has proposed an efficient level-wised architecture for identifying Devanagari isolated handwritten characters (as depicted in Figure 7). The main goal behind the implementation of a level-wised model is to create a powerful DHCR model with a small number of character images. Hence, a small convolutional network is designed from scratch on the first level as a baseline for evaluating its performance on small character samples. When there are fewer training samples, a good choice is to use pre-trained models that are trained on large-scale data with only minor changes. VGG16 is one of the most popular CNN architectures and is used frequently for image classification; therefore, the DHCR model is implemented using a VGG16 pre-trained network on the next level. The fully connected layer of the VGG16 pre-trained network is modified according to our requirement. To improve upon our prior performance, the fine-tuning is done on the top layers of the VGG16 pre-trained DHCR model on the last level. The features of the proposed level-wised architecture are as follows:

1. level-by-level enhancing performance of DHCR model;
2. promising results on small dataset;
3. to avoid overfitting of model, use real-time data augmentation and dropout layer;
4. fine-tuned model is computationally efficient.

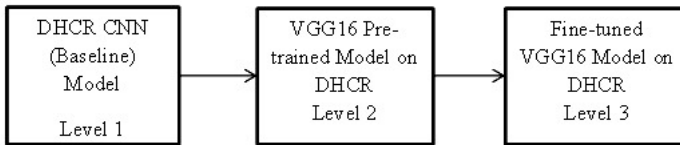


Figure 7. Proposed level-wised DHCR model

#### 4.3.1. First level: DHCR CNN architecture (baseline)

On the first level, the CNN model from scratch is implemented to recognize isolated Devanagari handwritten characters. Here, our main aim is to develop effective DHCR using CNN on a very small and newly created dataset. The architecture of the DHCR model is depicted in Figure 8. Our first-level model consists of a stack of five Convolution2D-BatchNormalization-ReLU-MaxPooling2D layers. The input images are resized to  $64 \times 64$  and divided into mini-batches. For fast training on a whole dataset, it is better to train the model on mini-batches; this also improves the feature flow during the training. Batch normalization is also used while training the model in order to stabilize the process of learning while also reducing the number of epochs. While training deep networks, small modifications affect the complete network; hence, mini-batch data normalization avoids this problem. Here, a  $3 \times 3$  convolution with fixed feature map dimensions 32, 64, 128, 256, and 512 were performed. The input

reduction between layers is obtained by using an increasing size of the stride (from one to two). In the convolution operation, a kernel of a size of 3 is used. In the batch normalization, there is no need to change the size of our volume because the batch-normalization operation is performed element-wise. The ReLU non-linear activation function is used (which is detailed in [Eq. 1]):

$$f(x) = \max(0, x). \quad (1)$$

Function  $f(x)$  returns 0 for a negative input; for any positive value of  $x$ , it returns that value back. The ReLU function consists of fewer mathematical tasks and is comparatively much faster than the other non-linear functions [17]. The output of a network is mapped using the Softmax function. The model is compiled using the RMSprop optimizer. The optimizer is used to optimize the results – to update the weights. This is an extension of the stochastic gradient algorithm; it is simple, straightforward, computationally efficient, and has fewer memory requirements.

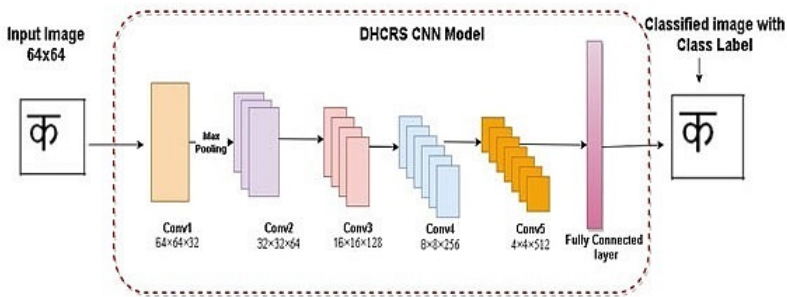


Figure 8. Architecture of DHCR model using CNN

#### 4.3.2. Second level: DHCR model using pre-trained bottleneck features

Pre-trained networks (transfer learning) often speed up the training process on new data (particularly when the dataset size is small), and they often produce a more accurate and effective model in general. A large amount of high-quality data is needed for direct supervised settings (which is an expensive process). A model that has already pre-trained on amply labeled training data will be able to handle a new similar task with trivial data in less time. During this stage, the convolution segment of a pre-trained network will be instantiated up to the fully connected layer (as shown in Figure 6) and then on top of the network, and our fully connected layer is added. Figure 9 represents the DHCR model that was derived from the VGG16 pre-trained model with two dense layers with one dropout layer (this dropout layer prevented our model from overfitting). The dropout technique is simply ignoring some random

neurons while training the network [33]. In this technique, the channel connections are temporarily removed from the network. After dropping the random units, the new lighter network is formed for training. Each neuron has a fixed probability of  $p$ ; setting  $p$  to 0.5 is one of the best options. A neural network learns more robust features by the dropout method. The Softmax function is used to map the network’s output to the expected output classes in the fully connected (dense 2) layer.

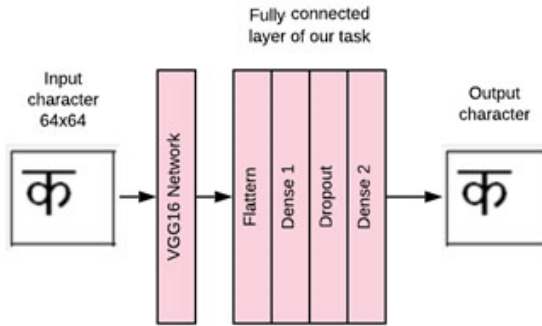


Figure 9. DHCR model using pre-trained bottleneck features

### 4.3.3. Third level: DHCR fine-tuned architecture

Fine-tuning the top level layers of the previous network is performed on this level to improve the overall efficiency of the DHCR model. Convolutional Blocks 5 and 4 of VGG16 is considered for fine-tuning (along with a top-level classifier). A minimum weight update is carried out while performing the fine-tuning. First, the VGG16 base is located; then, its weights are loaded. Next, our earlier defined fully connected model is added on top, and its weights are loaded. Finally, the VGG16 layers are frozen up to the third convolutional block of the network (as shown in Figure 10).

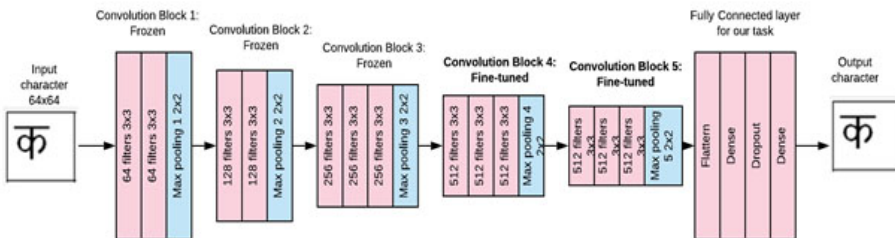


Figure 10. DHCR fine-tuned architecture

In the fine-tuning approach, the training is needed to be start with correctly trained weights, as randomly initializing the weights may ruin the learned weights of the convolutional base. Only Convolution Blocks 4 and 5 are fine-tuned (as seen in Figure 9) instead of the entire network to prevent the overfitting problem from arising due to the high entropic ability of the entire network. The more specific features of Convolutional Blocks 4 and 5 are updated instead of the more general features of the previous layer by freezing the initial blocks of the network. The new network is tainted when using a low LR, so destruction will be not be faced in the previously learned features. The steps of the fine-tuning of the DHCR model are mentioned in Algorithm 1.

Notations used in Algorithm 1:

- tr: training set,
- ts: testing set,
- bs: batch size,
- lr: learning rate.

// randomly divided total number of samples in Tr and Ts sets

Algorithm 1: Fine-tuned DHCR model

Input: (tr, ts, bs )

Output: Fine-tuned Devanagari handwritten character system model

```

1  Begin
2      Instantiate VGG16 model and load its weights
3      Add our second-level fully connected model on top of network
4      load its weights
5          top_model.add(Dense, ReLU)
6          top_model.add(Dropout, 0.5)
7          top_model.add(Output_classes, Softmax)
8          top_model.load_weights(first_level model)
9      Freeze trainable layers of model through third convolution block
10     top_model.compile(RMSProp Optimizer)
11     top_model.fit(tr, ts, bs, lr) // use slow learning rate
12 End

```

## 5. Experiments, results, and discussion

### 5.1. Experimental protocol

Our recommended framework was assessed on our Devanagari handwritten character dataset; in addition, it was also evaluated on some benchmark datasets. The recommended methods were assessed on the prevailing openly available Devanagari and Indic script datasets (including their character and digits). The four different datasets that were used for performing the experiments were the UCI Devanagari character and digit dataset [1], CMATERdb 3.1.1 of Bangla digits, and CMATERdb 3.1.2 of Bangla simple characters [30]. These datasets represent distinct scripts and

are a solid foundation for testing the recommended models. Refer to Table 1 for the dataset statistics that were used for the experiments.

**Table 1**  
Experimental data statistics

Dataset	No. of output classes	No. of trained samples	No. of test samples	Total samples
Our dataset (characters)	46	3840	960	4800
Our dataset (Digits)	10	800	200	1000
UCI Devanagari characters	36	61,200	10,800	72,000
UCI Devanagari digits	10	17,000	3000	20,000
CMATERdb 3.1.2	50	12,000	3000	15,000
CMATERdb 3.1.1	10	4000	2000	6000

The DHCR model was trained with diverse methods; each approach required a number of training parameters. A summary of the training hyperparameters for the different approaches is provided in Table 2. For the fine-tuned model, the trainable factor was zero with a constant condition. Also, the max-pooling layer's trainable parameter was zero since it passed the maximum value to the next layer. The different parameters (like machine, implementation tools, input image size, and image type) are depicted in Table 3.

**Table 2**  
Training hyperparameters of different proposed approaches

Model	Optimizer	Learning rate	No. of epochs	Mini-batch size
First-level model	RMSProp	1e-4	20	32
Second-level model	RMSProp	1e-4	20	For training = 20 and validation = 10
Third-level model	RMSProp	1e-6	10	For training = 20 and validation = 10

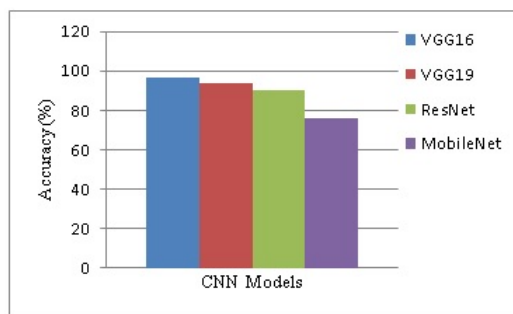
**Table 3**  
Other execution parameters

Parameter name	Parameter value
Machine	Google Colaboratory (12 GB NVIDIA Tesla K80 GPU)
Implementation tools	Keras and TensorFlow
Input image size	64 x 64
Input image type	JPG

## 5.2. Results and discussion

Our proposed level-wised DHCR has been implemented using different approaches; each approach has evolved, resulting in improvements in the recognition accuracy. The data was divided into batches of 32 samples during the first stage. Only an 85.64% accuracy was achieved by the model at this point; this uncertainty was due to the limited number of samples that were used for the validation. The experiment was then repeated with a larger batch size with all of the testing and training samples. The testing accuracy increased to 93.17% with a very high training time of 45 minutes and 53 seconds. Without any specific feature engineering, training a CNN from scratch on a small dataset will still provide reasonable results.

On the next level, the VGG16 pre-trained network was implemented. The use of a pre-trained network speeds up the training process on new data (particularly when a dataset's size is small), and it also results in a more accurate and effective model. Here, the pre-trained network increased the accuracy as compared to the first stage model; the results were achieved in 29.9 minutes with a 94.13% accuracy. Other pre-trained networks were also executed on our dataset to analyze the performance of various pre-trained networks; these testing accuracy results are shown in Figure 11.



**Figure 11.** Recognition accuracy rates of different models

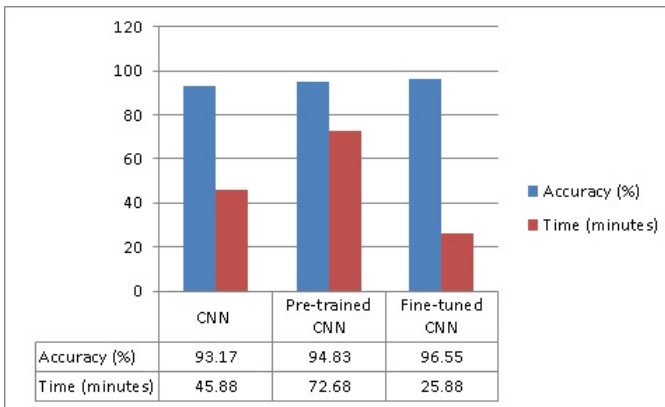
Figure 11 clearly shows that VGG16 performed better than the other models (especially with a small-sized dataset); hence, the VGG16 model was selected for

fine-tuning its parameters to improve the performance of the DHCR model. This approach contained a function  $f()$  that made a fine weight adjustment in the network by using a very low LR. Aggressive data augmentation and regularization techniques were also used in order to solve the overfitting problem.

**Table 4**  
Performance of second-level and fine-tuned models

Proposed model	Accuracy [%]	Average time/Epoch	Total time
Second-level model	94.13	1.50m (20 epochs)	29.9m 9s
Third-level model (Fine-tuned model)	96.47	1.62m (only 10 epochs)	16.2m 5s

Table 4 presents the obtained experimental results. It can be clearly observed that the proposed fine-tuned model is more efficient than the other stage models; the accuracy improved by approximately 2–3% in only 10 epochs. The performance analysis of the level-wised DHCR CNN model is shown in Figure 12. The fine-tuned DHCR CNN model improved the performance by 3% with a minimum time of 16.2 minutes.



**Figure 12.** Performance analysis of different level-wised DHCR models

By aggressively performing real-time data augmentation, distortions like transformation, scaling, and rotation can be handled. In real-time data augmentation, the batch of the original images are considered; then, a new transformed batch of images is generated for the training network by using the above augmentation operations. Each time a new variation of images is produced for learning features from images,

it helps to implement a more generalized DHCR model. With this the proposed fine-tuned method, we also tested on four popular standard handwritten datasets of different scripts that contained characters and digits. The proposed model was also separately tested on our character and digit dataset. The testing accuracy that was obtained on these four datasets is shown in Table 5. The proposed fine-tuned model shows encouraging outputs with different datasets (as depicted in Tables 5 and 6).

**Table 5**

Recognition rate of proposed fine-tuned model on different datasets

Datasets	Accuracy [%]
Our dataset (only characters)	97.05
Our dataset (only digits)	95.35
UCI Devanagari characters	97.80
UCI Devanagari digits	99.40
CMATERdb 3.1.2	95.83
CMATERdb 3.1.1	97.45

**Table 6**

Comparison of recognition accuracy with different research work

Dataset	Work reference	Accuracy [%]
UCI Devanagari character	[28]	93.00
	[35]	97.33
	Proposed fine-tuned model	97.80
UCI Devanagari numerals	[1]	98.47
	Proposed fine-tuned model	99.40
Bangla basic character	[23]	85.96
	[27]	93.40
	Proposed fine-tuned model	93.83
Bangla digit	[27]	97.26
	Proposed fine-tuned model	97.45

## 6. Conclusion

Deep neural networks have proven to be superior in pattern and script recognition for many languages with different script styles. Recognizing handwritten characters is a very complex job under unimpeded situations, so many researchers have worked on this problem in recent years. There have been several approaches for resolving this problem; however, there is a gap in automatic Devanagari character recognition. Realizing the need, we have attempted to implement a system for Devanagari handwritten character identification. In this paper, a newly created handwritten Devanagari character dataset was introduced (which is presented publicly); this is our contribution to the academic community for further research. DHCR has evolved by



using a different network architecture. On the first level, a CNN architecture was implemented from scratch in a supervised learning environment. On the next level, a pre-trained model was implemented using bottleneck features. Finally, a fine-tuned network was employed. The highest accuracy (96.47%) was achieved in 16.20 minutes when using the fine-tuned model on our small dataset. The proposed model was also executed on different Indic scripts. The obtained results prove the efficacy of the proposed fine-tuned model. In the future, modifications can be performed in the proposed algorithm to enhance the efficacy of the model in terms of space and time by developing a compact deep convolution network.

## Acknowledgements

Authors thank to Dr. S. H. Gawande, Professor, M.E.S. College of Engineering, Pune, India for his support and encouragement.

## References


- [1] Acharya S., Pant A.K., Gyawali P.K.: Deep Learning Based Large Scale Handwritten Devanagari Character Recognition. In: *Proceedings of the 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 121–126, 2015. doi: 10.1109/SKIMA.2015.7400041.
- [2] Alom M.Z., Sidike P., Hasan M., Taha T.M., Asari V.K.: Handwritten Bangla Character Recognition Using the State-of-the-Art Deep Convolutional Neural Networks, *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, 2018. doi: 10.1155/2018/6747098.
- [3] Bhunia A.K., Roy P.P., Mohta A., Pal U.: Cross-language framework for word recognition and spotting of indic scripts, *Pattern Recognition*, vol. 79, pp. 12–31, 2018. doi: 10.1016/j.patcog.2018.01.034.
- [4] Deore S.P., A. P.: Ensembling: Model of histogram of oriented gradient based handwritten Devanagari character recognition system, *Traitement du signal*, vol. 34(1–2), pp. 7–20, 2017. doi: 10.3166/ts.34.7-20.
- [5] Deore S.P., A. P.: Histogram of oriented gradients based off-line handwritten Devanagari characters recognition using SVM, K-NN and NN classifiers, *Revue d'Intelligence Artificielle*, vol. 33(6), pp. 441–446, 2019. doi: 10.18280/ria.330606.
- [6] Dutta K., Krishnan P., Mathew M., Jawahar C.: Towards Accurate Handwritten Word Recognition for Hindi and Bangla. In: *Computer Vision, Pattern Recognition, Image Processing, and Graphics. NCVPRIPG 2017, Communications in Computer and Information Science*, vol. 841, pp. 470–480, Springer, Singapore, 2017. doi: 10.1007/978-981-13-0020-2.41.
- [7] Guha R., Das N., Kundu M., Nasipuri M., Santosh K.C.: DevNet: An Efficient CNN Architecture for Handwritten Devanagari Character Recognition, *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34(12), 2019. doi: 10.1142/S0218001420520096.

- [8] He K., Zhang X., Ren S., Sun J.: Deep residual learning for image recognition, *CVPR arXiv:151203385[csCV]*, 2015.
- [9] Islam N., Islam Z., Noor N.: A Survey on Optical Character Recognition System, *Journal of Information & Communication Technology*, vol. 10(2), pp. 1–4, 2016.
- [10] Jangid M., Srivastava S.: Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods, *Journal of Imaging*, vol. 4(2), pp. 1–14, 2018. doi: 10.3390/jimaging4020041.
- [11] Jayadevan R., Kolhe S.R., Patil P.M., Pal U.: Offline Recognition of Devanagari Script: A Survey, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41(6), pp. 782–796, 2011. doi: 10.1109/TSMCC.2010.2095841.
- [12] Jino P.J., Balakrishnan K., Bhattacharya U.: Offline Handwritten Malayalam Word Recognition Using a Deep Architecture. In: *International Conference on Soft Computing for Problem Solving, Advances in Intelligent Systems and Computing*, vol. 816, pp. 913–925, Springer, Singapore, 2019. doi: 10.1007/978-981-13-1592-3\_73.
- [13] Kohli M., Kumar S.: Segmentation of handwritten words into characters, *Multi-media Tools and Applications*, vol. 80(14), pp. 22121–22133, 2021. doi: 10.1007/s11042-021-10638-0.
- [14] Lee H., Grosse R., Ranganath R., Ng A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML'09*, pp. 609–616, ACM, New York, 2009.
- [15] Liu W., Wang Z., Liu X., Zeng N., Liu Y., Alsaadi F.E.: A survey of deep neural network architectures and their applications, *Neurocomputing*, vol. 234, pp. 11–26, 2017. doi: 10.1016/j.neucom.2016.12.038.
- [16] Ly N.T., Nguyen C.T., Nguyen K.C., Nakagawa M.: Deep Convolutional Recurrent Network for Segmentation-Free Offline Handwritten Japanese Text Recognition. In: *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pp. 5–9, 2017. doi: 10.1109/ICDAR.2017.357.
- [17] Nair V., Hinton G.E.: Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning ICML'10*, pp. 807–814, Haifa, Israel, 2010.
- [18] Narang S., Jindal M.K., Kumar M.: Devanagari ancient documents recognition using statistical feature extraction techniques, *Sādhanā*, vol. 44(141), pp. 1–8, 2019. doi: 10.1007/s12046-019-1126-9.
- [19] Niu X.X., Suen C.Y.: A novel hybrid CNN–SVM classifier for recognizing handwritten digits, *Pattern Recognition*, vol. 45(4), pp. 1318–1325, 2012. doi: 10.1016/j.patcog.2011.09.021.

- [20] Obaidullah S.M., Halder C., Santosh K.C., Das N., Roy K.: PHDIndic\_11: page-level handwritten document image dataset of 11 official Indic scripts for script identification, *Multimedia Tools and Applications*, vol. 77, pp. 1643–1678, 2018. doi: 10.1007/s11042-017-4373-y.
- [21] Pal U., Chaudhuri B.B.: Indian script character recognition: A survey, *Pattern Recognition*, vol. 37(9), pp. 1887–1899, 2004.
- [22] Plamondon R., Srihari S.N.: Online and off-line handwriting recognition: a comprehensive survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22(1), pp. 63–84, 2000. doi: 10.1109/34.824821.
- [23] Rahman M.M., Akhand M.A.H., Islam S., Shill P.C., Rahman M.M.H.: Bangla Handwritten Character Recognition using Convolutional Neural Network, *International Journal of Image, Graphics and Signal Processing*, vol. 7(8), pp. 52–59, 2015. doi: 10.5815/ijigsp.2015.08.05.
- [24] Ren X., Zhou Y., Huang Z., Sun J., Yang X., Chen K.: A Novel Text Structure Feature Extractor for Chinese Scene Text Detection and Recognition, *IEEE Access*, vol. 5, pp. 3193–3204, 2017. doi: 10.1109/ACCESS.2017.2676158.
- [25] Roy P.P., Bhunia A.K., Das A., Dey P., Pal U.: HMM-based Indic handwritten word recognition using zone segmentation, *Pattern Recognition*, vol. 60, pp. 1057–1075, 2016. doi: 10.1016/j.patcog.2016.04.012.
- [26] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., et al.: ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015. doi: 10.1007/s11263-015-0816-y.
- [27] Saha C., Faisal R., Rahman M.M.: Bangla Handwritten Character Recognition Using Local Binary Pattern and Its Variants. In: *International Conference on Innovations in Science, Engineering and Technology (ICISSET)*, pp. 236–241, Chittagong, Bangladesh, 2018. doi: 10.1109/ICISSET.2018.8745645.
- [28] Saha P., Jaiswal A.: Handwriting Recognition Using Active Contour. In: *Artificial Intelligence and Evolutionary Computations in Engineering Systems, Advances in Intelligent Systems and Computing*, vol. 1056, pp. 505–514, Springer, Singapore, 2000. doi: 10.1007/978-981-15-0199-9\_43.
- [29] Santosh K.C., Wendling L.: Character recognition based on non-linear multi-projection profiles measure, *Frontiers of Computer Science*, vol. 9, pp. 678–690, 2015. doi: 10.1007/s11704-015-3400-2.
- [30] Sarkhel R., Das N., Basu S., Kundu M., Nasipuri M., Basu D.K.: CMATERdb1: a database of unconstrained handwritten Bangla and Bangla–English mixed script document image, *International Journal on Document Analysis and Recognition*, vol. 15, pp. 71–83, 2012. doi: 10.1007/s10032-011-0148-6.
- [31] Simonyan K., Zisserman A.: Very Deep Convolutional Networks for Large-Scale Image Recognition, *CVPR*, *arXiv:14091556v6[csCV]*, 2015.

- [32] Sonkusare M., Gupta R., Moghe A.: A Review on Character Segmentation Approach for Devanagari Script. In: *Intelligent Systems. Proceedings of SCIS 2021*, vol. 22(1), pp. 181–189, Algorithms for Intelligent Systems, Springer, Singapore, 2021. doi: 10.1007/978-981-16-2248-9\_19.
- [33] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R.: Dropout: A simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, vol. 15(56), pp. 1929–1958, 2014.
- [34] Su B., Lu S.: Accurate recognition of words in scenes without character segmentation using recurrent neural network, *Pattern Recognition*, vol. 63, pp. 397–405, 2017. doi: 10.1016/j.patcog.2016.10.016.
- [35] Vijaya Kumar Reddy R., Ravi Babu U.: Handwritten Hindi Character Recognition using Deep Learning Techniques, *International Journal of Computer Sciences and Engineering*, vol. 7(2), pp. 1–7, 2019. doi: 10.26438/ijcse/v7i2.17.
- [36] Wu Y.C., Yin F., Liu C.L.: Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models, *Pattern Recognition*, vol. 65, pp. 251–264, 2017. doi: 10.1016/j.patcog.2016.12.026.
- [37] Younis K.: Arabic Handwritten Character Recognition Based On Deep Convolutional Neural Networks, *Jordanian Journal of Computers and Information Technology*, vol. 3(3), pp. 186–200, 2018.

## Affiliations

Shalaka Prasad Deore 

S.P. Pune University, Department of Computer Engineering, M.E.S. College of Engineering, Pune, Maharashtra, India; shalakasonawane25@gmail.com, ORCID ID: <https://orcid.org/0000-0002-6767-6289>

**Received:** 25.09.2021

**Revised:** 28.03.2022

**Accepted:** 08.07.2022