Sādhanā (2020) 45:243
https://doi.org/10.1007/s12046-020-01484-1

Devanagari Handwritten Character Recognition using fine-tuned Deep Convolutional Neural Network on trivial dataset

SHALAKA PRASAD DEORE[1,2,*] and ALBERT PRAVIN[1]

[1]Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India
[2]Department of Computer Engineering, M.E.S. College of Engineering, Pune, S.P Pune University, Pune, India
e-mail: anudeore9@gmail.com; pravin_ane@rediffmail.com

MS received 6 December 2019; revised 3 August 2020; accepted 17 August 2020; published online 23 September 2020

Abstract. In order to rapidly build an automatic and precise system for image recognition and categorization, deep learning is a vital technology. Handwritten character classification also gaining more attention due to its major contribution in automation and specially to develop applications for helping visually impaired people. Here, the proposed work highlighting on fine-tuning approach and analysis of state-of-the-art Deep Convolutional Neural Network (DCNN) designed for Devanagari Handwritten characters classification. A new Devanagari handwritten characters dataset is generated which is publicly available. Datasets consist of total 5800 isolated images of 58 unique character classes: 12 vowels, 36 consonants and 10 numerals. In addition to this database, a two-stage VGG16 deep learning model is implemented to recognize those characters using two advanced adaptive gradient methods. A two-stage approach of deep learning is developed to enhance overall success of the proposed Devanagari Handwritten Character Recognition System (DHCRS). The first model achieves 94.84% testing accuracy with training loss of 0.18 on new dataset. Moreover, the second fine-tuned model requires very fewer trainable parameters and notably less training time to achieve state-of-the-art performance on a very small dataset. It achieves 96.55% testing accuracy with training loss of 0.12. We also tested the proposed model on four different benchmark datasets of isolated characters as well as digits of Indic scripts. For all the datasets tested, we achieved the promising results.

Keywords. Deep learning; Deep Convolutional Neural Network; VGG16; bottleneck features; fine-tuned; computing time.

1. Introduction

Pattern identification is a perpetual area of specialization in the context of artificial intelligence, computer vision and machine learning [1]. Optical Character Recognition (OCR) is one of the leading dynamic applications of an image classification and gaining additional interest due to its numerous applications. The applications include automatic postal card sorting [2, 3], digital signature verification [4], automatic processing of bank cheques [5], processing of historical documents [6] and automatic handwritten text detection in classroom teaching [7]. So there is a prodigious need of OCR in the area of pattern identification. OCR method is used to recognise handwritten, printed or typed text by using its scanned images. Basically, OCR technique is used to convert these scanned images into an editable form which improves the interaction between humans and computers. Handwritten Character Recognition (HCR) mainly entails the OCR method. Many researchers worked on HCR and also achieved good recognition accuracy; however it is impossible to get 100% accuracy to any character recognition system in real-life example. Here, reliability is also very important than high recognition rate. In many real-life applications which are mentioned above required high reliability to reduce losses. The higher complexity of handwritten character versus in print character can be attributed to: i) Presence of clatter during collection process for while writing, ii) Individual's writing style causing significant discrepancy and variation in strokes of a character, iii) Influence of a situation resulting variation in Individual's handwriting on different occasions, iv) Form and shape resemblance and v) Combinations with matra and composite letters add further intricacies. Due to the above stated reasons, it is not always feasible option to implement a common classifier to classify handwritten characters written by various writers [8].

India has several dialects, languages and scripts. Officially over twenty two languages are recognized in India [9]. An ancient and still one of the most widely used script in India is Devanagari, an important writing base for Indic

*For correspondence

languages such as Marathi, Hindi, Gujrati, Nepali, and Sanskrit [10]. With 58 characters including 12 vowels, 36 consonant and 10 numbers this script poses a challenge to design an efficient system to identify unconstraint handwritten Devanagari writing. Many classification tasks in computer vision make use of multilayer perceptron networks. However, selection of virtuous features is prime reliant for performance of such a network [11–13]. To achieve good recognition rate extracted features play a very important role. But to extract manually features from images required more time and it is very tedious job and they can not process raw images also. On the other side, there is no need to describe any explicit features for deep convolutional neural networks instead they make use of raw pixel information to generate the best features for classification [14]. Nowadays popularity of DCNN is increasing due to its number of applications in the area of pattern identification, natural language processing, speech detection, computer vision, and commendation systems [15]. A DCNN consists of multiple layers between input and output layers which required more number of trainable parameters and more number of connections. Each layer has a specific task of categorizing and ordering of extracted features. With DCNN other various popular deep learning networks have been presented like recurrent neural networks (RNN) and deep belief networks (DBN). A convolutional neural network (CNN) is one of the popular DCNN architectures, most commonly used for classification purposes. Xiao Niu *et al* [16] presented CNN-SVM hybrid methodology for handwritten digits classification. In this hybrid approach, features are automatically extracted using CNN and these features are sent for recognition to the SVM classifier. Author used MNIST dataset and reported promising recognition rate and also performed experiments on reliability analysis. In another example of HCR given in paper [17] also used an automatic features extraction approach using the LeNet5 convolutional neural network and classification done using SVM and achieved high performance, whereas various deep convolutional neural networks like VGG Net, ResNet, FractalNet, DenseNet, etc. are evaluated and their performance is discussed on the application of Handwritten Bangla Character Recognition in [18]. Comparisons are done between all latest models of DCNN mentioned above and found DenseNet performing better for them among all. Younis [19], presented deep CNN based approach to categorize handwritten Arabic characters. After applying the presented model author achieved classification accuracies 94.8% and 97.6% on AIA9k and AHCD datasets respectively, whereas in [20] integrated small convolutional neural networks are presented to identify different 11 Indic scripts. Authors implemented small trainable individual CNNs for each script which varied in different levels of CNN architecture and combined them for script identification. Jangid, *et al* [21] presented a layer-wise-trained deep CNN model and achieved a good recognition rate compared with standard deep CNN. Authors designed six different architecture of deep CNN by taking a number of combinations of convolutional-pooling layers and number of neurons. In [22], authors introduced deep quad-tree based new model for prediction. It is the network of multiple level trees which perform recognition in faster way. Addition to this, multi-column multi-scale CNN architecture also presented. This architecture consists of three-level columns. The first-level of individual column comprises of a distinct CNN, however the succeeding two levels comprises of four and sixteen CNNs respectively. Proposed model evaluated on number of different datasets and shows promising results. Gupta *et al* [23] presented multi-objective Harmony search algorithm to identify local regions of the character in less time. With this two more objectives are achieved: minimum recognition time with less redundancy in local regions and higher classification rate using SVM classifier. In this paper, the best cost effective approach is presented to recognize isolated handwritten characters and digits. Deep CNN architecture with the dropout layer and dataset increment approach is presented in [24] to recognize large scale i.e., 92 thousand images of handwritten Devanagari characters. This large new benchmark dataset of 92000 of Devanagari script is created by the author and achieved promising accuracy. Aneja *et al* [25] presented transfer learning approach to recognize Devanagari alphabets in which various pre-trained networks such as VGG 11, VGG 16, VGG 19, AlexNet, DenseNet 121, DenseNet 201and Inception V3 are implemented. Highest accuracy of 99% is achieved using Inception V3 model due to different regularization techniques. AlexNet achieved better performance in the term of execution time means transfer learning is one of the good optimization method to increase the efficiency of the system. Guha *et al* [26] analysed different CNN models in terms of training parameters, memory space, and execution time. Mainly authors focused on designing part of the model to produce efficient model in the terms of less space and time. In this work, the presented model is executed on publicly accessible datasets and achieved promising results. Designed model is simple. It has less number of layers hence required less time for training.

As per deliberated above deep learning techniques, mostly convolutional neural network architecture is used for image classification. It has also been effectively implemented to recognize different languages of different scripts like Roman (MNIST) [16], Arabic [19] and Bangla [27–31].

Our core contributions are summarized below.

1. Created a new dataset of handwritten Devanagari characters is which is publicly available.
2. Presented a two-stage model to recognise Devanagari handwritten characters. The first model is trained using bottleneck features of a pre-trained network. The second model is fine-tuned on top of the pre-trained network to achieve better accuracy.

3. Various researches have been done in the area of deep learning using various deep networks to the best of our knowledge fine-tuning approach is not yet explored on small handwritten Devanagari dataset. Therefore, in this work a fine-tuned Devanagari Handwritten Character Recognition System (DHCRS) is presented to classify 58 isolated unconstrained characters using VGG16 architecture of deep CNN on small dataset. Here, features are extracted automatically from the handwritten character images by our proposed model and then it classifies an unknown character image to its relevant class.

The paper is structured as follows. Section 2 discussed about characteristics of the Devanagari script. Corpus creation is deliberated in section 3. The proposed work is discussed in section 4. In section 5, training model and the experimental results are discussed. Finally, the conclusions are given in section 6.

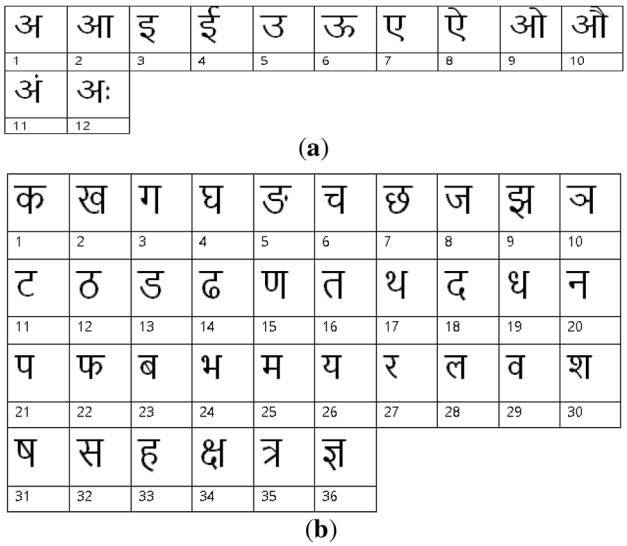

Figure 1. Printed samples (**a**) vowel characters and (**b**) consonant characters.

2. Devanagari script

In India, Devanagari script is one of the highly popular scripts and used as basis for the most spoken languages such as Marathi, Hindi, Nepali, and Sanskrit [10]. Devanagari script follows left to right direction for reading and writing. Horizontal line is used at the top of the letters which continue till end of each word. The horizontal line is called as "Shirorekha" and used to separates one word from other. With its origin in Sanskrit, the script has been adopted and modified for native languages in India. Handwritten Devanagari script is peculiar compared to English script as there is no cursive connected writing and one has to scribble letters or even curves, matras by lifting the hand. Upper case and lower case categorization is not included in Devanagari script.

2.1 *Devanagari characters and numerals dataset*

Devanagari script comprises of total 58 characters. Apart from these characters it also consists of composite characters which are combination of two or more basic characters. Printed vowel and consonants are depicted in figure 1 with numbering. Figure 2 represents handwritten vowel and consonant characters. Writing system of Devanagari script is mixture of characters, numerals and syllabary. It follows phonetic principle where many characters follow mixture of vowels and consonants as well as writing is also accordingly to the sounds of the characters. So Devanagari script is also called as phonetic script. It also consists of 14 modifiers which is nothing but the part of vowel and used with consonants. These modifiers can use at top, down and right of the consonant letter to form Barakhadi letters. For example Barakhadi letter "का" is written like "क+आ=का" . Further, consonants are categorised in many classes like stops, semivowels, spirants, etc. Accordingly to the character's structural feature 60% characters consists of vertical lines either in middle or at the end and 40% with not having vertical line. Some characters are rounded in shape. Handwritten numeral characters are shown in figure 3.

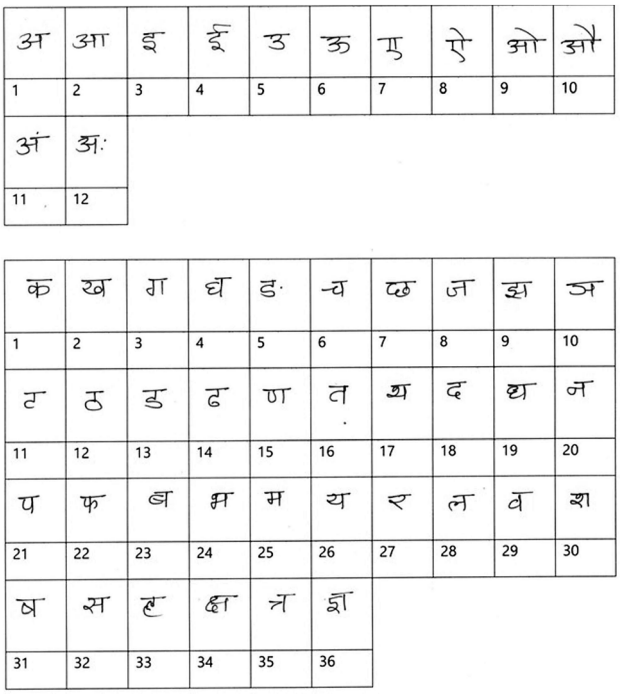

Figure 2. Handwritten samples of Devanagari vowels and consonants.

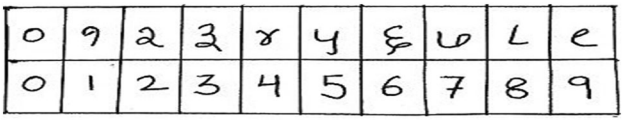

Figure 3. Devanagari numerals.

प	ष	The difference being middle line present
थ	य	The difference being loop present
अ	अं	The difference being single dot present on top
ढ	ढ	The difference being loop and down stroke present
ध	ध	The difference being horizontal line present at the top

Figure 4. Similar shape characters with their structural differences.

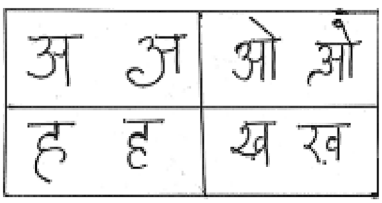

Figure 5. Different writing style.

2.2 *Different challenges occur in handwritten Devanagari character recognition*

The significant factor which makes Devanagari more complex is the similarity of characters as depicted in figure 4. This similarity in characters makes it difficult to distinguish characters especially when those characters are handwritten. In Devanagari script there are many characters found to be similar in shape. The only small difference is there in their structure like dot, loops, presence of horizontal line which is shown in figure 4. Due to their structural similarity, difficulty occurs while recognition. Different writing style of the same character with different pressure makes it further complex. Such examples are shown in figure 5.

3. Dataset corpus creation

Data pattern plays important role and has necessities in many applications of pattern recognition. A good quality data pattern always helps us to implement better recognition system. Formation of database is certainly a cumbersome and lengthy job. The objectives of creating new dataset are: i) To check the reliability of the proposed model on newly created dataset along with standard datasets; ii) To improve deep learning performance by providing new data samples on the basis of data collection method, by adding new variant data samples. This also helps to contribute towards the academic community for further research. Devanagari handwritten characters are collected under direction from different local writers. Piece of paper is given to them for writing isolated handwritten characters using ball pen. Constraints like quality of paper, ball pen and writing style are not kept while writing. Figures 2-3 show sample of handwritten Devanagari characters and numerals collected by one writer. Devanagari Handwritten Character corpus is generated by 100 writers of various age groups. Age group limit kept between 15 and 60 year old. Then all papers are scanned using scanner separately to create dataset of Devanagari characters. Each scanned image is labelled manually by its class name and sequence number. For example, first image of character "अ" of class 1 is labelled as C1_1.jpg. The character sample size was 1600×1600 pixels. Our Devanagari characters dataset consist of 58 character classes out of those 12 classes for vowels, consonant classes were 36 and 10 are numeral classes as depicted in figures 2-3. Class one consists of 100 samples of character "अ", class two consists of 100 samples of character "आ" and so on. Hence the total size of the dataset is 5800 images with 4800 characters and 1000 numerals. The newly created dataset has the following features:

- UCI Devanagari dataset consists of total 46 classes containing of 36 consonants and 10 numerals. Our new dataset consist of total 58 classes which contain 12 vowels along with 36 consonants and 10 numerals.
- No restriction on ball pen, writing pressure, quality of the paper and age of the writer.
- No restriction on the size, style and the location of written character on paper.

4. Proposed system

4.1 *Pre-processing and data augmentation*

The data is collected without keeping restriction on size and location as mentioned in earlier section may contain errors or outliers. So that before passing the data for training, needs to clean or pre-processed. Pre-processed data supports to boost the performance of the model. By using XnConvert batch processing tool consistent dataset is created by applying various filters like minimum filter. Minimum filter is used to increase the thickness of each character in the batch and curves are used to change the colour channels of an image to highlight or reduce some specific features. With consistent and noise free data

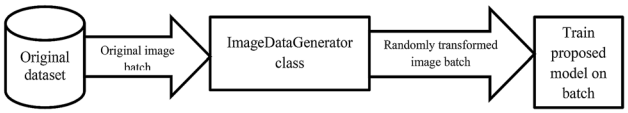

Figure 6. Real-time augmentation process.

DCNN also requires a transformed data to developed generalized model [32]. Data augmentation is the one which consists of many methods to generate new training samples from the original samples. When the network continuously takes modified new data for training, it learns more robust features of input data. So we performed real-time data augmentation to increase the generalizability of the proposed model.

ImageDataGenerator class of Keras is used for obtaining data augmentation when the model is training on the batches of images. Then applying different random transformations it create new random transformed batch of images which is used for training model as shown in figure 6. In our work, translation, rotation, scaling and zooming augmentation techniques are applied to the existing dataset [33]. Image normalization is also performed during training. One more reason to perform these mentioned augmentation operations is, ConvNet is not invariant to translation, rotation, scaling so to handle images with random transformations and to improve the efficiency of our model different data augmentation techniques are used. It also helped us to prevent model from overfitting.

4.2 *VGG16: Convolution Neural Network Architecture*

Convolution Neural Network (CNN/ConvNet) is a superior multi-layer neural network, aimed to automatically extract features right from the raw pixel images which required minor pre-processing [27]. One of the features of ConvNet is that it has the ability to learn features from the small dataset using a pre-trained model like ImageNet. ImageNet Large Scale Visual Recognition Competition (ILSVRC) held in year 2014. In this competition, Visual geometry group (VGG) was at second place [34]. VGG Net categorized into 3 types based on their architecture. They are categorized based on number of layers present while building network. These three types are: VGG11, VGG16, and VGG19. Devanagari Handwritten Character Recognition System (DHCRS) is trained using VGG16 and VGG19 architectures. On ImageNet VGG16 architecture achieved 92.7% top-5 test accuracy. ImageNet dataset consist of over fourteen million images of 1000 classes [35]. In the field of deep learning the VGG architecture is the first deep CNN to achieve most promising results. As well as it is very appealing network because of its simple and unifying architecture. VGG Net is mostly used to take out baseline features from the given input images. It is also not very deep to drop any features at the end of the network while training. It works finely on our small dataset that we have created. For balancing the computational cost in the ImageNet 2014 challenge, VGG architecture deploys smaller convolutional filters (3×3) and a lesser number of receptive field channels in exchange to enhance the depth in networks [36]. The VGG19 architecture is similar to VGG16 only difference that it is deeper as compared to VGG16 architecture. In VGG19 there are total 19 weight layers and three more convolution layers are added in Conv Block 3, 4 and 5 respectively. VGG16 architecture mainly comprises of 3 layers [35]: convolution layer, pooling layer, and fully connected layer as depicted in figure 7.

A) The convolution layer is the essential layer of a convolutional neural network. It makes use of small learnable filters to extract various features from the input image.
B) To decrease the training features and computations in network the pooling layer is employed. The pooling operation is done by the five max-pooling layers using 2×2 kernel. Max pooling extracts the maximum value of the block and produces a single output. Another way of doing pooling is taking average value or combination of both maximum and average value.
C) The fully connected layer mainly used for classification purpose. This layer can be changed as per our requirement.

The VGG16 architecture consists of 13 convolutional layers, 5 max-pooling layers and 3 dense layers. Conv1 has 64 filters of 3×3 kernel while Conv2 has 128 filters of 3×3 kernel, Conv3 has 256 filters of 3×3 kernel while Conv4 and Conv5 has 512 filters of 3×3 kernel as shown in figure 7. All max-pooling layers has 2×2 kernel. The ReLu activation function is added to each convolution layer and two dense layers so that it will not pass negative values to the subsequent layers. Softmax activation function is used by the last dense output layer for prediction.

Mainly for training network, weights are updated to reduce loss in network. Here learning rate plays an important role to determine weights adjustment in each layer of the network. Slow learning rate takes more time to minimize errors occur in the network where high learning rate distract network to get minimum error. Hence adaptive gradient optimizers are best choice for training network faster. In this paper for evaluating our DHCRS model two adaptive gradient optimizers, RMSprop and Adam are used.

The Root Mean Square Propagation (RMSprop) is one of the faster adaptive learning optimizers proposed by Geoff Hinton [37]. RMSprop automatically adjusts learning rate of all parameters separately. Next, the weight updations are done using following formula for each parameter (Eqs. (1)-(2)):

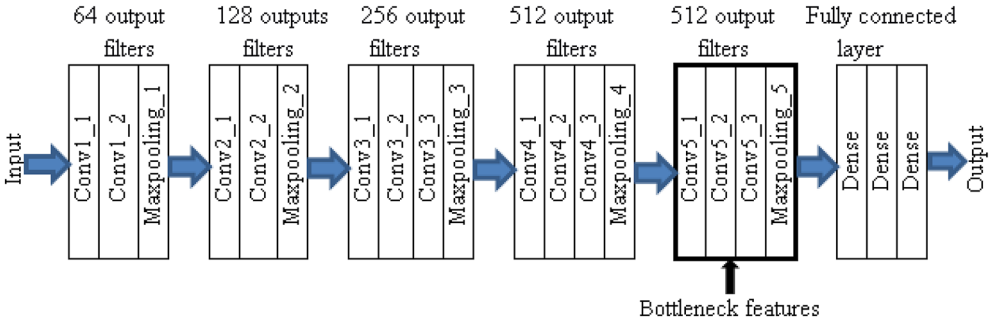

Figure 7. VGG16 architecture.

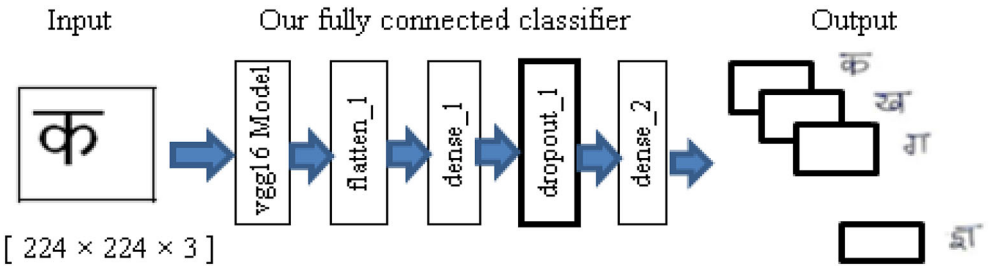

Figure 8. DHCRS Model.

$$X_t = \rho X_{t-1} + (1-\rho) \times G_t^2 \tag{1}$$

$$W_{t+1} = W_t + \left(-\eta / \sqrt{X_t + \in}\right) \times G_t \tag{2}$$

where X_t is exponential average of all gradients squares and G_t gradients at time t. It updates gradient components accordingly parameters directions. W_t is previous weight and W_{t+1} is updated weight using learning rate η.

The Adaptive Moment Estimation (Adam) [38] is combination of RMSprop and Momentum. It basically gives us first and second order moments of gradients. These moments are calculates using following formula (Eqs. (3)-(4)):

$$X_t = \beta_1 X_{t-1} - (1-\beta_1) \times G_t \tag{3}$$

$$Y_t = \beta_2 Y_{t-1} - (1-\beta_2) \times G_t^2 \tag{4}$$

where X_t and Y_t are 1st and 2nd order moments of gradients and β_1 and β_2 are hyper- parameters and its default values are set to 0.9 and 0.99, respectively.

4.3 *Proposed architectures of DHCRS*

Here effective two-stage architecture to recognize Devanagari handwritten characters using very small training samples is proposed. In the first stage, VGG16 pre-trained network is used. We removed its fully connected (FC) layer and built our own FC in place of it. And in the next stage fine-tuning is done on the upper layers of this pre-trained network.

4.3a *First stage: DHCRS architecture*: In first stage we only instantiate the convolution part indicated by bottleneck features in figure 7 and then added our fully connected layer on the top of the network. Figure 8 depicts the first stage of DHCRS model with one dense layer of 1024 units with ReLU activation function, one dropout layer having 0.5 dropout rate and the last output dense layer of 58 units with softmax activation function. ReLU is non-linear activation function stated as Eq. (5),

$$f(z) = \max(0, z) \tag{5}$$

The reason for using ReLU function instead of other popular and widely used nonlinear functions like sigmoid and hyperbolic tangent (Tanh) is because training with gradient-descent is moderately much faster [39] and ReLU does not face gradient vanishing problem also. ReLu activation function is less computationally costly than Tanh and sigmoid because it contains easy mathematical operations. One dropout layer is also added to avoid overfitting in our model. Dropout means eliminating the units both visible and invisible in the network [40]. Arbitrary units are removed temporarily along with all connections from the networks. During every training iteration, after elimination of arbitrary units from denser architecture a lighter network is left. Then this new network is trained and tested. Each

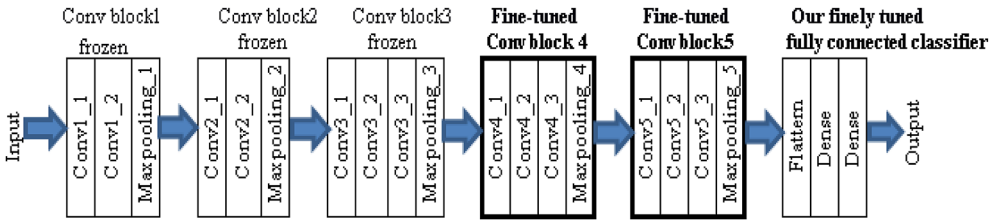

Figure 9. New Fine-tuned DHCRS architecture.

unit is reserved with the fixed probability say p and it is not dependent on other units. Here, p is assigned to 0.5 which is the optimal choice by many deep learning models to avoid overfitting problem. In paper [41], author has performed experiments on speech recognition by varying dropout rate from 0.2 to 0.5 and they achieved good result when the dropout rate is increased towards 50%. Dropout technique makes a network to learn very powerful features. It approximately doubles the number of iterations. However, the training time is less for each epoch. In dense_2 layer, the softmax function is applied to predict output classes. It uses softmax loss function to compute cross- entropy loss which is defined as Eq. (6),

$$Li = -\log\left(e^{fyi} \Big/ \sum_j e^{fj}\right) \tag{6}$$

The main goal of the network is minimizing the cross-entropy to reduce loss between the estimated output and target output. Finally, weights are saved for the fine-tuned network which is implemented in next stage.

4.3b *Second stage: New Fine-tuned DHCRS architecture*: To improve our result obtained by first stage architecture, we fine-tuned our model at the next stage. Convolutional block 5 and 4 which is the last block of the VGG16 model is fine-tuned. Initially, trained network is considered for fine-tuning approach, then doing minor weight updates re-trained it on our dataset. To fine-tune our model following three steps are performed: i) Instantiate the VGG16 convolutional base and load its weights, ii) Use our formerly trained first stage model, iii) Freeze top layers of the VGG16 model up to the convolutional block 3. Figure 9 shows our fine-tuned DHCRS model.

To fine-tune network on top of pre-trained network, we can not initialize fully connected network arbitrarily because after initializing randomly weights to the network large updation occur in gradient which can ruin the learned weights. So initially we have to begin with properly trained weights. Firstly, we make use of previously trained model and then only can start fine-tuning convolutional weights beside it. In order to prevent overfitting problem which occurs due to high entropic ability of entire network, instead of fine-tuning whole network we only fine-tuned convolution blocks 4 and 5 as shown in figure 9. By fine-tuning mentioned convolutional blocks more specific features are updated rather than more general features getting from first few blocks of the network so we kept first few blocks fixed (frozen). The new fine-tuned network trained using optimizer with slow learning rate because slow learning will keep magnitude of the updates less and so it will not ruin the previously learned features. Results are improved using a fine-tuning approach on DHCRS.

5. Experiments, results and discussions

5.1 *Model training*

The proposed model is evaluated on our own new hand-written Devanagari characters dataset which contains total 5800 Devanagari character images shown in figures 2 and 3. Out of 5800 samples, 4800 are vowels and consonants samples and 1000 are numerical samples collected from different 100 users. Dataset is split into training and testing dataset. We consider 80 random samples in training dataset any 20 random samples in testing dataset out of 100 samples of each class. The statistics of dataset used for experiment is summarized in table 1. Input shape of all character images are considered $[224 \times 224 \times 3]$ for performing experiments.

The DHCRS model is trained using two different approaches. The first stage model is broad and includes multiple different parameters. The training parameters of the first model are listed below and table 2 shows summary of parameters used in first model.

- Optimizer: RMSProp optimizer. It is an adaptive and faster learning optimizer and it work well on our sequential network than other optimizers like Adam. Results obtained using Adam optimizer is shown in table 3.
- Learning rate: 0.0001 (lr=1e-4)
- Number of epochs: 20
- Dropout : 0.5
- Mini-Batch size: training batch size=20 and validation batch size=10

The second stage model is fine-tuned last convolution block of our first model along with top-level classifier. It is a compact network as compared with the first network

Table 1. New Devanagari dataset statistics used for experiment.

Dataset	Training samples	Testing samples	Total samples	Number of classes
Devanagari characters (vowel + consonants)	3840	960	4800	48
Devanagari numerals	800	200	1000	10
Total	4640	1160	5800	58

Table 2. First model parameter specifications.

Layer(type)	Output shape	Parameters
vgg16(Model)	(None, 7, 7, 512)	14714688
flatten_3(Flatten)	(None, 25088)	0
dense_5(Dense)	(None, 1024)	25691136
dropout_3(Dropout)	(None, 1024)	0
dense_6(Dense)	(None, 58)	59450
Total parameters:	40,465,274	
Trainable parameters:	32,830,010	
Non-trainable parameters:	7,635,264	

because we are freezing other layers up to the last layer. It also increases overall efficiency of the first stage model. The training parameters of the second model are listed below.

- Optimizer: RMSprop optimizer with very slow learning rate because the magnitude updation remains very small and so that it will not ruin the previously learned features.
- Learning rate: le-6
- Number of epochs: 10
- Dropout : 0.5
- Mini-Batch size: training batch size=20 and validation batch size=10

In second model specification remain same only trainable parameters are zero. Also trainable parameter in maxpooling layer is zero because it passes max value to the next layer.

5.2 *Execution details*

The DHCRS is executed using Keras and TensorFlow framework using GPU runtime. One of the key attributes of using Keras is choice of hyper-parameters required for training network. They are based on preprogramed guidelines; hence the programmers can test with default parameters. All experiments are performed on Google Colaboratory which is free cloud service for developing deep learning applications. Google Colaboratory provides free single 12 GB NVIDIA Tesla K80 GPU. Real-time data augmentation is done using ImageDataGenerator class of Keras. Random horizontal flips, scaling and random rotation by 20 degree, data augmentation techniques are applied on batch of original input images to generate new batch of transformed images. Each optimizer has its default framework. In this work for testing the proposed model, the optimizer parameters are set by the authors. RMSprop optimizer is used for our two-stage model with low learning rate to avoid destruction in previously learned features.

5.3 *Results and discussion*

5.3a *Recognition accuracy archived on newly created Devanagari dataset*: Initially, the first model is executed using 10 epochs and achieved 93.28% testing and 91.67% training accuracy. As a number of epochs increased accuracy also increased approximately by 2%. For 20 epochs 94.83% testing and 95.7% training accuracy is achieved as epoch plays an important role in training network. Mini-batch training is faster than training on complete data set because it can take advantage of vectorised operations to process the entire mini-batch at once. It also helps in improving the flow of features properly. A mini-batch size of 20 instead of quite large 50 is chosen because if a larger mini-batch size is picked for our small dataset it slows down model and also minimizes accuracy. This is important for proposed model because the value is required for the gradient descent without degrading.

To avoid overfitting problem which may occur due to a huge amount of the trainable parameters real-time data augmentation and dropout techniques are applied while training proposed model. As well as these data augmentation techniques make our model invariant to translation, rotation, and scale.

Table 3. Performance of proposed two-stage model with different optimizer.

Optimizer	Model	Training Loss	Accuracy (%)	Average time/epoch	Total time
RMSprop	First stage Model	0.18	94.83	3.63 m (20 epochs)	72.68 m 20 s
	Second stage Model	**0.12**	**96.55**	**2.58 m (only 10 epochs)**	**25.88 m 8 s**
Adam	First stage Mode	0.24	92.76	3.53 m (20 epochs)	70.6 m 14 s
	Second stage Model	0.20	94.05	2.51 m(15 epochs)	37.67 m 10 s

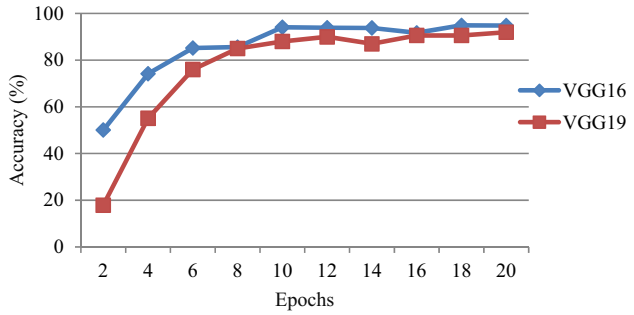

Figure 10. Comparison of testing accuracy results of VGG16 and VGG19.

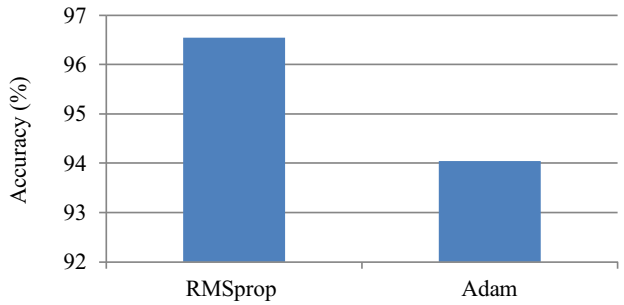

Figure 11. Testing accuracy obtained by different optimizers.

In deep learning fine-tuning approach gives good performance especially image classification [42]. The DHCRS recognition model also evaluated using VGG19 architecture. The testing accuracy results while training are depicted in figure 10. VGG16 model performing better than VGG19 clearly noted from figure 10. It also required more storage space compared to VGG16. Hence we decided to do fine-tuning on VGG16 architecture only using different adaptive optimizers. Second fine-tuned model is a very powerful model which improves result compared with the first model. It involves fine-tuning the pre-trained model function $f()$ with a smaller learning rate so that it will make fine adjustments in weights to further improve our result. With improving results overfitting issue is also handled by using aggressive data augmentation and regularization techniques.

Table 3 presents the obtained experimental results for the first and second models using different optimizers. The recognition results are promising. From table 3, it is clearly observed that our second fine-tuned model is more efficient. It achieved good performance in 10 epochs only and accuracy has also improved by approximately 2% with training loss of 0.12. Fine-tuning of pre-trained model really speeds up the training process with improved accuracy. The pre-trained model speeds up training process on new data especially when dataset size is small and it also results in more accurate and effective model. It provides initial point for other similar task. Hence transfer learning is

Table 4. Two-stage model performance on various benchmark datasets.

Proposed models	Datasets with size			
	Devanagari characters (72000)	Devanagari digits (20000)	Bangla characters (15000)	Bangla digits (6000)
First stage model	96.19%	98.90%	94.33%	94.55%
Second stage model	97.80%	99.40%	95.83%	97.45%

optimization method which enhances the efficiency of other task [25]. For direct supervised setting large amount of high-quality data is required. We had also implemented small network from scratch with total five layers of Conv2D- Batch Normalization- ReLu- MaxPooling2D. Initially the data was divided in batch size of 32 samples. Due to less number of samples used for validation, we encountered inconsistency in validation accuracy. We achieved 85.64% accuracy only. Then the experiment performed by increasing batch size with entire testing and training samples. With this the testing accuracy increased to 93.17%, but resulted in very high training time of 45.88 minutes. Our proposed fine-tuned network achieved 96.66% accuracy in 25.88 minutes. The accuracy is improved by approximately 3.5% in lesser time as compared to the model tested under supervised setting (no pre-training).

Distortion like scaling, rotation and transformation is done in both model using run-time data augmentation. Figure 11 depicts testing accuracy obtained by different adaptive gradient optimizers. For our DHCRS RMSprop optimizer is good choice which is giving us good accuracy with faster convergence rate. With this proposed fine-tuned model is also tested on our separate characters and numerals datasets. Testing accuracy achieved on character and numeral dataset is 97.20% and 95.50%, respectively.

5.3b *Recognition accuracy archived on standard Devanagari and other different datasets*: The proposed approaches are also evaluated on four standard publically available datasets belonging to Devanagari and Bangla Indic scripts. These datasets consist of characters and numerals data of above mentioned scripts. These scripts are dissimilar from one another. Apabhramsha is base for the Devanagari script [43], degraded from Prakrit [43] where the Bangla script inspired by Sanskrit as well as Magdhi Prakrit [44]. Hence the proposed model can be completely tested on these two different scripts. For experiments we considered total four benchmark datasets: UCI Devanagari character dataset [24] consist of both characters and digits, CMATERdb 3.1.1 of Bangla digit [45] and CMATERdb 3.1.2 of

Table 5. Relative analysis of proposed method with UCI Devanagari character dataset.

Dataset	Ref. No.	Methodology and Experimental Protocol	Features	Accuracy (%)
UCI Devanagari character	[49]	Deep feed-forward neural network with 2 hidden layers	Features extracted by CNN	95.57
		Convolutional neural network (1 Convolution layer, 1 Max-pooling and 2 fully connected layers) with RMSprop optimizer		97.33
		Convolutional neural network with Adam optimizer		96.02
	[50]	Convolutional neural network (3 Convolution layer, 3 Max-pooling and a fully connected layer) with Adam optimizer	Features extracted by CNN	93
		Tool: Tensorflow and Keras API, 4 GB NVIDIA GPU		
	Proposed Fine-tuned model	Second stage fine-tuned VGG16 Model with RMSprop optimizer	Features extracted by CNN	**97.80**
		Tool: Tensorflow and Keras API, Google Colaboratory		

Table 6. Relative analysis of proposed method with Devanagari numerals dataset.

Dataset	Ref. No.	Methodology and Experimental Protocol	Features	Accuracy (%)
UCI Devanagari Numerals	[24]	First model-Deep CNN	Features extracted by CNN	98.47
		Second model- LeNet CNN		98.26
	[47]	Backpropagation neural network	Projection Histogram	92.2
		Tool: MATLAB	Chaincode Histogram	92.7
		Deep Auto-encoder network-Restricted Boltzmann Machine(RBM)	Binary obtained by RBM	98.20
		Tool: MATLAB		
	Proposed Fine-tuned model	Second stage fine-tuned VGG16 Model with RMSprop optimizer	Features extracted by CNN	**99.40**
		Tool: Tensorflow and Keras API, Google Colaboratory		

Table 7. Relative analysis of proposed method with Bangla basic character dataset.

Dataset	Ref. No.	Methodology and Experimental Protocol	Features	Accuracy (%)
Bangla basic character	[27]	Convolutional neural network (2 Convolution layer, 2 Max-pooling and 2 fully connected layers)	Features extracted by CNN	85.96
	[28]	Multilayer perceptron-Backpropagation(MLP-BP) NN	Chaincode histogram	92.14
	[29]	Deep CNN((2 Convolution layer, 2 Max-pooling and 3 fully connected layers)	Features extracted by CNN	91.23
	[31]	Support Vector Machine(SVM) with linear kernel	Noise Adaptive Local Binary Pattern	93.40
	Proposed Fine-tuned model	Second stage fine-tuned VGG16 Model with RMSprop optimizer	Features extracted by CNN	**95.83**
		Tool: Tensorflow and Keras API, Google Colaboratory		

Table 8. Relative analysis of proposed method with Bangla digits dataset.

Dataset	Ref. No.	Methodology and Experimental Protocol	Features	Accuracy (%)
Bangla digits	[30]	K- Nearest Neighbor	Local binary pattern histogram	96.7
	[31]	Support Vector Machine(SVM) with linear kernel	Local Adaptive Image Descriptor	97.26
	[48]	Support Vector Machine(SVM) with RBF kernel Tool: WEKA	Gradient based directional and quad-tree based	97.45
	Proposed Fine-tuned model	Second stage fine-tuned VGG16 Model with RMSprop optimizer Tool: Tensorflow and Keras API, Google Colaboratory	Features extracted by CNN	**97.45**

Table 9. Mostly confusing characters.

Characters	Confused with
ट	ठ, द
घ	ध
प	५, य
ढ	६

Bangla basic characters [46]. Table 4 shows the good recognition results achieved on these four different standard datasets. Such promising results achieved on benchmark dataset of two Indic scripts prove the efficiency of our two-stage model.

5.3c *Comparison study*: Experiments done on different datasets implies promising results obtained by the proposed model. In this section comparison with other popular work presented on the different dataset as mentioned in table 4 using the proposed method. The comparative study reveals that the DHCRS model put forward by us is most sophisticated when working with small Devanagari character database. Our test outcomes are better than other as indicated in tables 5-8 and result of the proposed work marked in bold. Numbers highlighted in bold indicates the results obtained by our proposed model for comparison purpose.

Table 5 shows comparative analysis on Devanagari character dataset. This study reveals that the results obtained by the proposed Fine-tuned model on our dataset and on UCI dataset giving good accuracy. Our new dataset's statistics are given in table 1. UCI dataset is consisting of total 72000 character samples which are split into training (85%) and testing (15%) dataset.

Table 6 shows comparative analysis on Devanagari numeral dataset. Our new dataset's training statistics are given in table 1. UCI dataset is consisting of total 20000 numeral (17000 training and 3000 testing) samples. The proposed model achieved maximum accuracy on UCI numeral dataset as depicted in table 6.

Table 7 shows comparative analysis on Bangla basic character dataset. CMATERdb 3.1.2 isolated Bangla character dataset used for experiments. Dataset consist of total 15000 samples of 50 basic Bangla characters out of which 12000 used for training and 3000 used for testing.

Table 8 shows comparative analysis on Bangla digit dataset. Experiments are performed on CMATERdb 3.1.1 isolated Bangla digit dataset. Dataset consist of total 6000 samples of 10 Bangla digits out of which 4000 used for training and 2000 used for testing. Results are testimonial to the efficiency of the recommended model performing well of diverse datasets.

Hence the implemented handwritten Devanagari character recognition model is more powerful for a small dataset as well with a fine-tuning technique. In addition to this, more aggressive data augmentation techniques and dropout layer also added to prevent model from overfitting. The main advantage of the proposed model is good accuracy on trivial dataset. With this we also achieved good accuracy on different benchmark datasets. Secondly VGG architecture of CNN is most simple and found more suitable to fine-tune to improve result on our small size dataset. However the limitation of the proposed model is intensive memory of the model due to more number of training parameters. In the future we will address memory size reduction problem by working on model compression techniques without much reduction in accuracy.

After analysing, the misclassification errors occurred during testing phase it is found that the characters (table 9) are mostly confused with other class of characters due to their shape similarly and structure. The structure is also plays important role due to individual's cursive ways. Some characters are confused with numerals. Other problem for confusion occurs due to people's bad handwriting and scanning quality.

6. Conclusions

Identification of Indic languages is cumbersome and complex due to individual's style of writing. Devanagari is foundation of many Indic languages. Recently demand for deep networks is increasing in many research areas especially to solve image classification problems. In this work, a two-stage model has been developed to classify the isolated handwritten Devanagari characters of our newly created dataset. Newly created dataset which is publically available consists of a total of 5800 images of vowels and consonants. 80 images of each 58 classes are used for training the DHCRS model and 20 images of each 58 classes are used for testing. The VGG16 architecture of DCNN assisted us to detect the key attributes automatically and to also categorize them. The best classification accuracy of 96.55% is achieved using fine-tuned VGG16 architecture. It is the best result achieved on our trivial database. To make the DHCRS model more powerful we added more aggressive runtime data augmentation and some regularization techniques like Dropout and Batch Normalization. Results obtained by the recommended modes on two Indic scripts show the effectiveness of the proposed approach.

In future, deploying further compact deep convolution network to classify handwritten Devanagari character is planned. Also, we want to explore different deep convolution network architectures on compound Devanagari characters as well as Devanagari words. Structural features like concavity need to be explored to improve accuracy of similar shape characters.

Dataset accessibility Our newly created Handwritten Devanagari character dataset is publically available at https://www.kaggle.com/shalakadeore/handwritten-marathi-devanagari-characters.

References

[1] Bishop C 2006 *Pattern Recognition and Machine Learning*. Springer, Berlin

[2] Basu S, Das N, Sarkar R, Kundu M, Nasipuri M and Basu D K 2010 A novel framework for automatic sorting of postal documents with multi-script address blocks. *Pattern Recogn*. 43(10): 3507–3521

[3] Roy K, Vajda S, Pal U, Chaudhuri B B and Belaid A 2005 A system for Indian postal automation. In: *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, IEEE Computer Society, USA, pp. 1060–1064

[4] Pham T A, Le H H and Do N T 2015 Offline handwritten signature verification using local and global features. *Ann. Math. Artif. Intell*. 75: 231–247

[5] Palacios R and Gupta A 2008 A system for processing handwritten bank checks automatically. *Image Vis. Comput*. 26: 1297–1313

[6] Romero V, Serrano N, Toselli A H, Sanchez J A and Vidal E 2011 Handwritten Text Recognition for Historical Documents. In: *Proceedings of the workshop on Language Technologies for Digital Humanities and Cultural Heritage*, Hissar, Bulgaria, pp. 90–96

[7] Banerjee P, Bhattacharya U and Chaudhuri B B 2014 Automatic Detection of Handwritten Texts from Video Frames of Lectures. In: *Proceedings of 14th International Conference on Frontiers in Handwriting Recognition*, Heraklion, pp. 627–632

[8] Liu C L, Nakashima K, Sako H and Fujisawa H 2003 Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recogn*. 36(10): 2271–2285

[9] Singh P K, Sarkar R and Nasipuri M 2015 Offline Script Identification from multilingual Indic-script documents: a state-of-the-art. *Comput. Sci. Rev*. 15-16: 1–28

[10] Pal U and Chaudhuri B B 2004 Indian script character recognition: A survey. *Pattern Recogn*. 37(9): 1887–1899

[11] Deore S P, Pravin A 2017 Ensembling: Model of histogram of oriented gradient based handwritten Devanagari character recognition system. *Traitement du Signal*. 34(1-2): 7–20

[12] Ruck D W, Rogers S K and Kabrisky M 1990 Feature selection using a multilayer perceptron. *J. Neural Netw. Comput*. 2(2): 40–48

[13] Yang J, Shen K, Ong C and Li X 2009 Feature selection for mlp neural network: The use of random permutation of probabilistic outputs. *IEEE Trans. Neural Netw*. 20(12): 1911–1922

[14] Lee H, Grosse R, Ranganath R and Ng A Y 2009 Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pp. 609–616

[15] Liu W, Wang Z, Liu X, Zeng N, Liu Y and Alsaadi F E 2017 A survey of deep neural network architectures and their applications. *Neurocomputing* 234: 11–26

[16] Niu X and Suen C Y 2012 A novel hybrid CNN–SVM classifier for recognizing handwritten digits. *Pattern Recogn*. 45: 1318–1325

[17] Lauer F, Suen C Y and Bloch G 2007 A trainable feature extractor for handwritten digit recognition. *Pattern Recogn*. 40: 1816–1824

[18] Alom M Z, Sidike P, Taha T M and Asari V K 2018 Handwritten Bangla character recognition using the state-of-the-art deep convolutional neural networks. *Comput. Intell. Neurosci*. 2018: 1–13

[19] Younis K 2017 Arabic handwritten character recognition based on deep convolutional neural networks. *Jordan. J. Comput. Inf. Technol*. 3(3):186-200

[20] Ukil S, Ghosh S, Obaidullah S M, Santosh K C, Roy K and Das N 2019 Improved word-level handwritten Indic script identification by integrating small convolutional neural networks. *Neural Comput. Appl*. 32: 2829–2844

[21] Jangid M and Srivastava S 2018 Handwritten Devanagari character recognition using layer-wise training of deep convolutional neural networks and adaptive gradient methods. *J. Imaging*. 4: 1–14

[22] Sarkhel R, Das N, Das A, Kundu M and Nasipuri M 2017 A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular indic scripts. *Pattern Recogn.* 71: 78–93

[23] Gupta A, Sarkhel R, Das N and Kundu M 2019 Multiobjective optimization for recognition of isolated handwritten Indic scripts. *Pattern Recogn. Lett*. 128: 318-325

[24] Acharya S, Pant A and Gyawali P 2015 Deep Learning Based Large Scale Handwritten Devanagari Character Recognition. In: *Proceedings of the 9th International Conference on Software, Knowledge, Information Management and Applications*. Kathmandu, pp. 1–6

[25] Aneja N and Aneja S 2019 Transfer Learning using CNN for Handwritten Devanagari Character Recognition. In: *Proceeding of the 1st International Conference on Advances in Information Technology*. Chikmagalur, India, pp. 293–296.

[26] Guha R, Das N, Kundu M, Nasipuri M and Santosh K C 2019 DevNet: An efficient CNN architecture for handwritten Devanagari character recognition. *Int. J. Pattern Recogn. Artif. Intell.*

[27] Rahman Md M, Akhand M A H, Islam S, Shill P C and Rahman M M H 2015 Bangla handwritten character recognition using convolutional neural network. *Int. J. Image Graph. Signal Process*. 7: 42–49

[28] Bhattacharya U, Shridhar M and Parui S 2006 On recognition of handwritten Bangla characters. In: *Computer Vision, Graphics and Image Processing, Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 4338, pp. 817–828

[29] Purkaystha B, Datta T and Islam M 2017 Bengali handwritten character recognition using deep convolutional neural network. In: *Proceeding of 20th International Conference of Computer and Information Technology*. Dhaka, pp. 1–5

[30] Hassan T and Khan H 2015 Handwritten Bangla numeral recognition using Local Binary Pattern. In: *Proceeding of International Conference on Electrical Engineering and Information Communication Technology*. Dhaka, pp. 1–4

[31] Saha C, Faisal R and Mostafijur R 2018 Bangla Handwritten Character Recognition Using Local Binary Pattern and Its Variants. In: *Proceeding of International Conference on Innovations in Science, Engineering and Technology*. Chittagong, Bangladesh, pp. 236–241

[32] Kobetski M and Sullivan J 2013 Apprenticeship learning: transfer of knowledge via dataset augmentation. In: *Proceeding of the Scandinavian Conference on Image Analysis*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 7944, pp. 432–443

[33] Perez L and Wang J 2017 The effectiveness of data augmentation in image classification using deep learning. In: *arXiv preprint* arXiv:1712.04621

[34] Russakovsky O, Deng J and Su H 2015 Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis*. 115: 211–252

[35] Simonyan K and Zisserman A 2015 Very Deep Convolutional Networks for Large-Scale Image Recognition. In: arXiv:1409.1556v6

[36] Balci B, Saadati D and Shiferaw D 2017 Handwritten Text Recognition Using Deep Learning. In: *CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University. Course Project Report*

[37] Hinton G coursera course - lecture 6e 2012. http://www.cs.toronto.edu/tijmen/csc321/slides/lecture_slides_lec6.pdf

[38] Kingma D P and Lei Ba J 2015 Adam: a Method for Stochastic Optimization. In: *Proceeding of the International Conference on Learning Representations*. pp. 1–13

[39] Nair V and Hinton G E 2010 Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning*. Haifa, Israel, pp. 807–814

[40] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res*. 15(1): 1929–1958

[41] Passricha V and Aggarwal R K 2019 A comparative analysis of pooling strategies for convolutional neural network based Hindi ASR. *J. Ambient Intell. Humaniz. Comput*. 11: 675–691

[42] Too E C, Yujian L, Njuki S and Yingchun L 2019 A comparative study of fine-tuning deep learning models for plant disease identification. *Comput. Electron. Agric*. 161: 272–279

[43] Ostler N 2005 *Empires of the Word: A Language History of the World*. HarperCollins, New York

[44] Emeneau M 1956 India as a Lingustic Area. *Language (Baltim)*. 32: 3–16

[45] Sarkhel R, Das N, Basu S, Kundu M and Nasipuri M 2012 CMATERdb1: a database of unconstrained handwritten Bangla and Bangla – English mixed script document image. *Int. J. Doc. Anal. Recogn*. 15: 71–83

[46] Das N, Reddy J M, Sarkar R, Basu S, Kundu M, Nasipuri M and Basu D K 2012 A statistical–topological feature combination for recognition of handwritten numerals. *Appl. Soft Comput.* 12: 2486–2495

[47] Sarkar A, Singh K and Mukerjee A 2012 Handwritten Hindi Numerals Recognition System. *Webpage:* https://www.cse.iitk.ac.in/users/cs365/2012/submissions/aksarkar/cs365*, CS365 project report*

[48] Roy A, Das N, Sarkar R, Basu S and Kundu M 2014 An Axiomatic Fuzzy Set Theory Based Feature Selection Methodology for Handwritten Numeral Recognition. In: *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol I. Advances in Intelligent Systems and Computing*. 248, pp. 133–140

[49] Vijaya Kumar R and Babu U 2019 Handwritten Hindi character recognition using deep learning techniques. *Int. J. Comput. Sci. Eng*. 7(2): 1–7

[50] Saha P and Jaiswal A 2020 Handwriting recognition using active contour. In: *Proceeding of the Artificial Intelligence and Evolutionary Computations in Engineering Systems. Advances in Intelligent Systems and Computing*. Springer, Singapore, 1056, pp. 505–514