Neural Computing and Applications (2020) 32:2829–2844
https://doi.org/10.1007/s00521-019-04111-1

ORIGINAL ARTICLE

Improved word-level handwritten Indic script identification by integrating small convolutional neural networks

Soumya Ukil[1] · Swarnendu Ghosh[1] · Sk Md Obaidullah[2] · K. C. Santosh[3] · Kaushik Roy[4] · Nibaran Das[1]

Received: 16 October 2017 / Accepted: 22 February 2019 / Published online: 6 March 2019

Abstract

Handwritten document recognition has been an active domain of research in the field of computer vision for several years since 1914 with the development of handheld scanner for reading printed texts called "optophone". In India, which has several different scripts in one document page, identifying them is a must to automate process: document understanding. We propose a novel technique in integrating convolutional neural networks (CNNs) for script identification. We combined small individually trainable small CNNs, and used several different levels of variation in the architectures of the individual CNNs. Such a collection of individually trainable modules vary with respect to the input image size, CNN's depth and wavelet transformation. In our test, we used publicly available dataset of size 11K words (1K per script) from 11 different Indic Scripts: Bangla, Devanagari, Gujarati, Gurumukhi, Kannada, Malayalam, Oriya, Roman, Tamil, Telugu and Urdu. Several ensemble strategies were implemented such as *max-voting* and *probabilistic voting* are used in addition to other conventional approaches like feature concatenation. We achieved a maximum accuracy of 95.04%, and it outperforms the accuracy of the state-of-the-art techniques like AlexNet by 2.9% and more importantly, benchmark techniques as (for script identification) on the dataset by more than 4%.

Keywords Convolutional neural network · Deep learning · Haar wavelet transform · Document analysis · Indic script recognition · More

1 Introduction

Optical Character Resolution (OCR) techniques convert printed and/or handwritten document images into machine encoded text. Along this process, script recognition task is a must in case we face multi-script document images that come from multi-lingual country like India, where several different scripts are available in one document such as postal documents and bank checks (see Fig. 1). Due to shared ancestry, many Indian scripts have lots of common attributes and shapes among them. These common shapes across various scripts pose a serious challenge in this domain of research. A traditional handwritten script recognition task takes an image containing a handwritten chunk of text as an input and the scripts that they belong to at the output. The purpose of the problem is not to recognize content, rather it deals with script identification. Our method involves word-level Indic script identification. Implementing a word-level approach, rather than a page-

✉ Nibaran Das
nibaran@gmail.com

Soumya Ukil
soumyaukil60@gmail.com

Swarnendu Ghosh
swarbir@gmail.com

Sk Md Obaidullah
sk.obaidullah@gmail.com

K. C. Santosh
santosh.kc@ieee.org

Kaushik Roy
kaushik.mrg@gmail.com

[1] Jadavpur University, Kolkata, WB 700032, India

[2] Aliah University, Kolkata, WB 700156, India

[3] The University of South Dakota, Vermillion, SD 57069, USA

[4] West Bengal State University, Kolkata 700126, WB, India

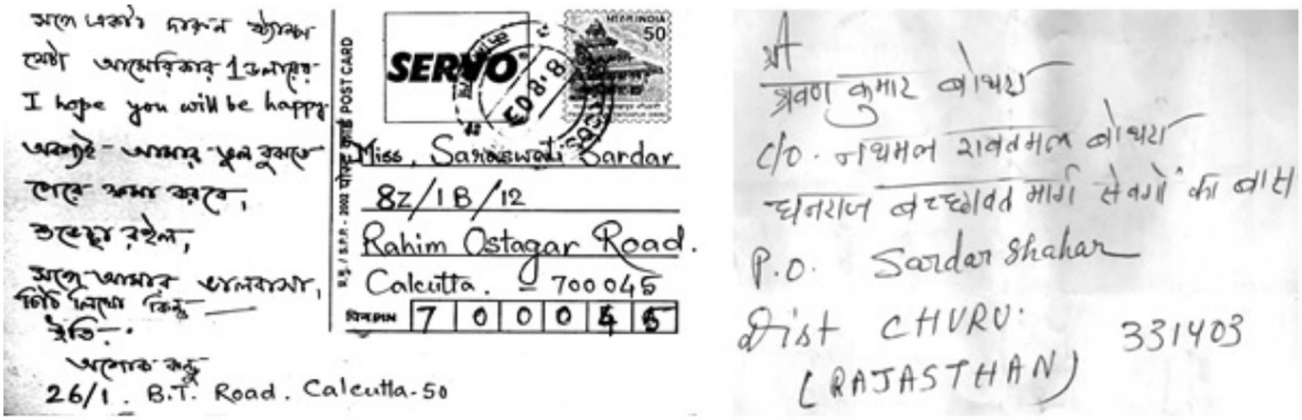

Fig. 1 Samples of multi-script document images

level approach, allows us to tackle multi-lingual documents since, more often a single page contains several scripts from line to line and/or from word to word. Our proposal involves combination of individually trainable small convolutional neural networks (CNNs). We provide a thorough study over design choices and performance of different combination strategies. In what follows, we discuss pertinent references from the literature (Sect. 1.1), and specific details of our contribution (see Sect. 1.2). Section 2 describes the various modules of our architecture. Section 3 explains the various experimentations performed. These experimentations have been thoroughly analyzed in Sect. 4. Each design choices have been scrutinized to understand their impact on the architecture. Finally, three combination approaches have been compared and their performance was fared against the benchmarks and some state-of-the-art architectures. In Sect. 5, we conclude our findings.

1.1 Related works

OCR techniques dealing with handwritten script identification has a rich literature [13–15, 22, 26–28, 40, 44]. More often, document recognition systems were script specific [9, 27, 36]. In 2002, an approach for word-level script identification was proposed for Tamil and Roman scripts, where a set of Gabor filters was used to distinguish two printed scripts [11]. Unlike printed scripts, handwritten script identification is found to be difficult in several aspects, such as variants in the properties of different writing instruments, variety in the style of handwriting among various writers, as well as variety in the style of a single writer during different points of time, situation and emotional conditions. All these make each handwritten sample unique in their own way, thus providing ample challenge to researchers. The availability of a Indic numeral dataset that is composed of Devanagari, Bangla and Oriya numerals accelerated the research in this domain [4]. Many feature-based approaches were available [5] since then. Clonal selection algorithms [12], texture-based features [7, 8], appearance-based (by using "Matra") [3], fractal-based features, component-based features and topological features [34, 35] are some of the notable features implemented in this respect. Other methods included application of probabilistic models like hidden Markov model (HMM) [37] and support vector machine (SVM) [45]. Another direction of research involved the use of Wavelet transforms [16]. Especially, discrete wavelet transforms proved to be a potent method [15]. Wavelet transform very much like Fourier transform (FT) is used to transform a signal from the time domain to the frequency domain. FT [6, 42] does not work well with non-stationary signals as it does not successfully provide information about what frequencies are present at what time. To overcome this shortcoming, Wavelet transform (WT) [10] and short time Fourier transform (STFT) [30] were conceived. Neural networks (NNs), in general, were also considered in this domain [38]. A plethora of research has been conducted at various levels of data [25] like page-level [24], block-level [23], line-level [32] etc. These micro-level approaches allowed to deal with documents containing texts with more than one script. Our methods deal with even a smaller level of recognition that is the word level. Not much research exists in such a micro-level [29, 33, 39, 41]. While feature-based showed its worth, the onset of deep learning techniques [18–20] set an entirely new benchmark for computation vision problems. Usage of hand-crafted features was a major setback for the previously mentioned techniques. Deep learning techniques excel in finding the optimum representation of visual data. Hence, deep learning methodologies [1, 2, 21] started surfacing to solve these challenges. However, deep learning techniques tend to be very computationally expensive. Due to the large sizes of the NNs and millions of floating

point operations, it normally relies on GPU-based parallel computations that use thousands of CUDA computational cores. Bigger models tend to require bigger GPUs and hence more costly hardware, on the other hand smaller model compromises on performance. One of the comprehensive data known by the name PHDIndic_11 dataset [24] has been made publicly available, and it contains 11 different Indic scripts. This provides us with an extremely challenging problem to deal with.

1.2 Our contribution

Feature-based approaches may not be generic enough since feature selection requires domain experts (to some extent). For a multi-script document images, difficulty has been more pronounced and it may require several experts (with a-priori information about scripts). Straight-forward use of CNNs have proved to be quite robust for a various handwritten document classification problem. With increasing trend of problem complexity, CNN architectures tend to be hardware intensive. Keeping such issues in mind, our contribution is mainly focused in extracting more information from a set of individually trainable linear CNN architectures that are relatively small in size. The primary advantage of such small CNNs is that it can be trained in comparatively less hardware intensive environment. Subtle variations were taken into consideration (in CNNs) on three specific grounds: (a) CNN's depth; (b) wavelet transformation (input image); and (c) scaling (scale input images to different sizes). We observed and provided how each of these variations affects the performance. We analyzed our method from two different phases. In the first phase, we hierarchically combined individual CNNs to see/observe how these variations contribute to the overall performance (i.e., accuracy). In our second phase, for script identification, different combination strategies were tested namely, max-voting, probabilistic voting that is based on the outputs of the individual CNNs and feature concatenation.

2 Materials and methods

As mentioned in Sect. 1.2, we have used a number of small CNNs in our work. There are two main branches of experiments. First, with normal inputs and secondly with inputs transformed using Haar Wavelet transformation. Firstly, wavelet transformation will be discussed showing how they were used to transform the inputs. After that, we explain CNN architectures used in the process. Finally, a discussion on the various ensemble techniques will conclude our entire set of models. In Fig. 2, a complete flowchart of how all these modules combine to create the different architectures is shown.

2.1 Image representation: wavelet transform (WT)

WTs are used most widely as a dimensionality reduction tool for high-dimensional data and they achieve this by converting a signal from the time domain representation to the frequency domain representation. In the frequency domain representation, we can neglect certain low-frequency components and reconstruct a signal which is very much alike the original signal using high-frequency components alone. However, during the process, noise is smoothened out [31] and rich contours which are crucial for script identification are left behind. Discrete Wavelet transform (DWT) has an inherent advantage of conceiving both frequency and location information(location in time space). To use DWT, we first resize input image to 128×128 and then we think of this image as a 2D time signal, to which we cascade multiple levels of 2D DWT, from the family of HWTs [43]. The coefficients C_1 obtained by doing a single-level two-dimensional Haar wavelet decomposition on an image I_0 can be expressed as:

$$\begin{aligned} C_1 &= \Psi_{\mathrm{HWT}}(I_0) = WIW^{\mathrm{T}} \\ &= \begin{bmatrix} L \\ H \end{bmatrix} I_0 \begin{bmatrix} L \\ H \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} LI_0L^{\mathrm{T}} & LI_0H^{\mathrm{T}} \\ HI_0L^{\mathrm{T}} & HI_0H^{\mathrm{T}} \end{bmatrix} \\ &= \begin{bmatrix} A_{I_1} & D^{\mathrm{H}}_{I_1} \\ D^{\mathrm{V}}_{I_1} & D^{\mathrm{D}}_{I_1} \end{bmatrix}. \end{aligned} \tag{1}$$

The top half L of the wavelet transformation kernel W is composed of low-pass or averaging filters like $(\sqrt{2}/2, \sqrt{2}/2)$. The bottom half H is composed of high-pass or difference kernels like $(-\sqrt{2}/2, \sqrt{2}/2)$. Every decomposition divides the image into four parts as shown above, where A_{I_k} is the approximation coefficient and $D^{\mathrm{H}}_{I_k}$, $D^{\mathrm{V}}_{I_k}$, and $D^{\mathrm{D}}_{I_k}$ are the horizontal, vertical and diagonal detail coefficients of the k-th level transform. We can get the higher level coefficients C_k by a decomposing the approximation coefficient of the previous level as shown below,

$$C_k = \Psi_{\mathrm{HWT}}(A_{I_{k-1}}). \tag{2}$$

The image can be reconstructed using all the coefficients obtained by the multilevel decomposition using inverse Haar wavelet transform.

$$I = \Psi^{-1}_{\mathrm{HWT}}(C) = W^{\mathrm{T}}CW. \tag{3}$$

At every level, the approximation coefficient becomes a quarter of the size of the image with each edge being

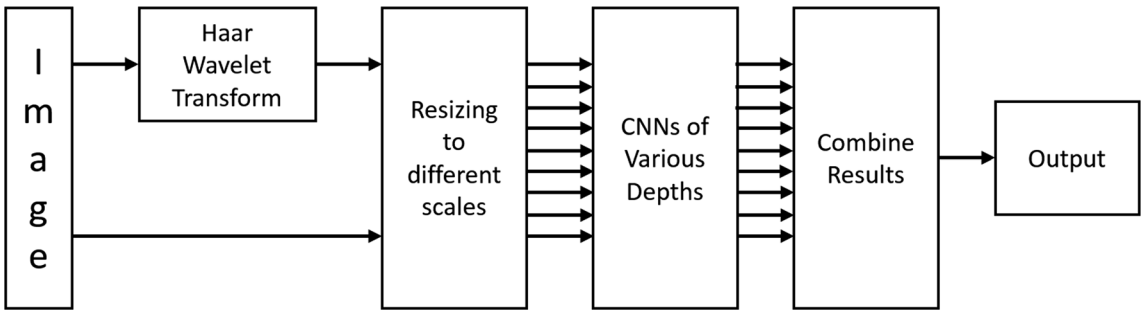

Fig. 2 Workflow: a broad overview of different modules in the approach

halved. With an input of 128×128 we can consider cascading a maximum of 7 such levels of decomposition. For our purpose, the final approximation coefficient A_{I_7} is ignored during the reconstruction by setting it to zero. This allows us to remove a lot background noise and force the network to focus only on important aspects. The decomposition and reconstruction process can be visualized as in Fig. 3. Some samples of reconstructed images are shown in Fig. 6. The 128×128 input image were then further rescaled to 32×32 and 64×64 thus resulting in 3 different scales of input for both raw and WT inputs. These pathways of experimentations have been named using a specific convention which will be discussed in Sect. 2.2.

2.2 Overview: convolutional neural network architectures

CNNs have been more realistic since the LeNet [20] proved its worth for the MNIST digit dataset. The popularity of CNNs was further boosted with the application of the AlexNet [18] over the 1000 class ImageNet object recognition challenge. CNN broadly consists of three different types of layers, namely convolutional layer (CL), pooling layer (PL), and fully connected layer (FCL). The CL has a set of filters, where parameters can be modified as training proceeds in such a way that they are able to provide different feature maps from the input. The PL is mainly used to reduce redundancy of the input and lesser memory requirement in higher layers. In our case, we performed max pooling at the pooling layers. The FCL is a multi-layer perceptron (MLP) that takes a feature vector as an input, which is produced after a sequence of convolution and pooling, and returns the output as soft-max probabilities for each of the scripts (i.e., 11 scripts in our dataset). Together, CL and PL are referred to as a convolutional block hereafter. We have used two different CNN architectures with two and three convolutional blocks, respectively, and our CNN architectures are shown in Fig. 4.

2.3 Nomenclature

As mentioned in Sect. 1.2, we have three specific parameters to work on different types of CNNs, and in general, we call it by $\text{CNN}_{\text{W,S,D}}$, where the factors that govern the changes/variations can be explained below.

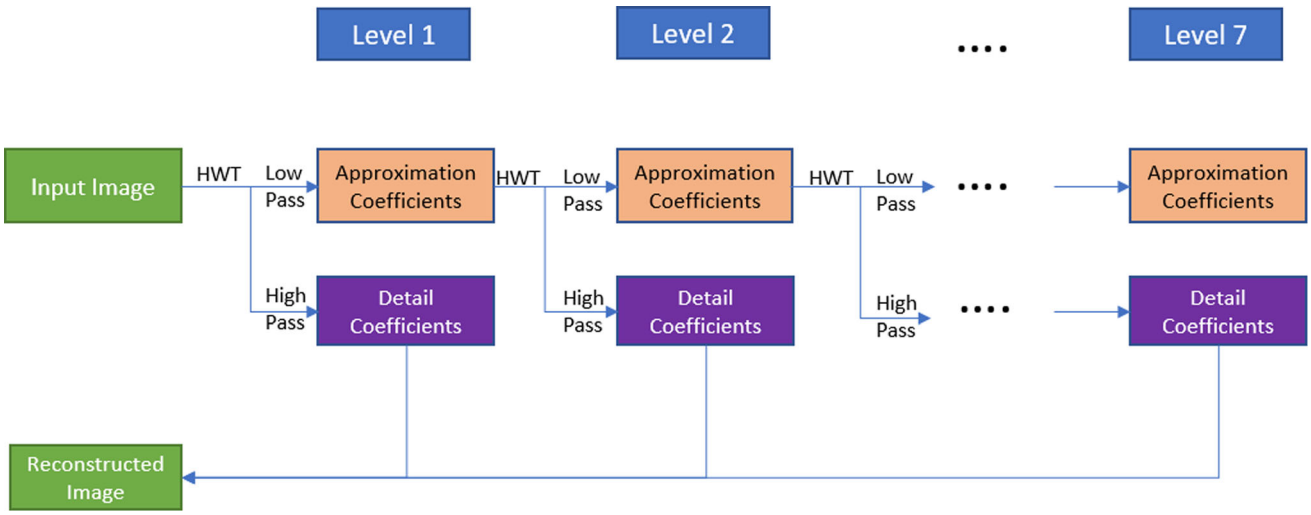

Fig. 3 Decomposition and reconstruction of input image using a seven level Haar wavelet transform

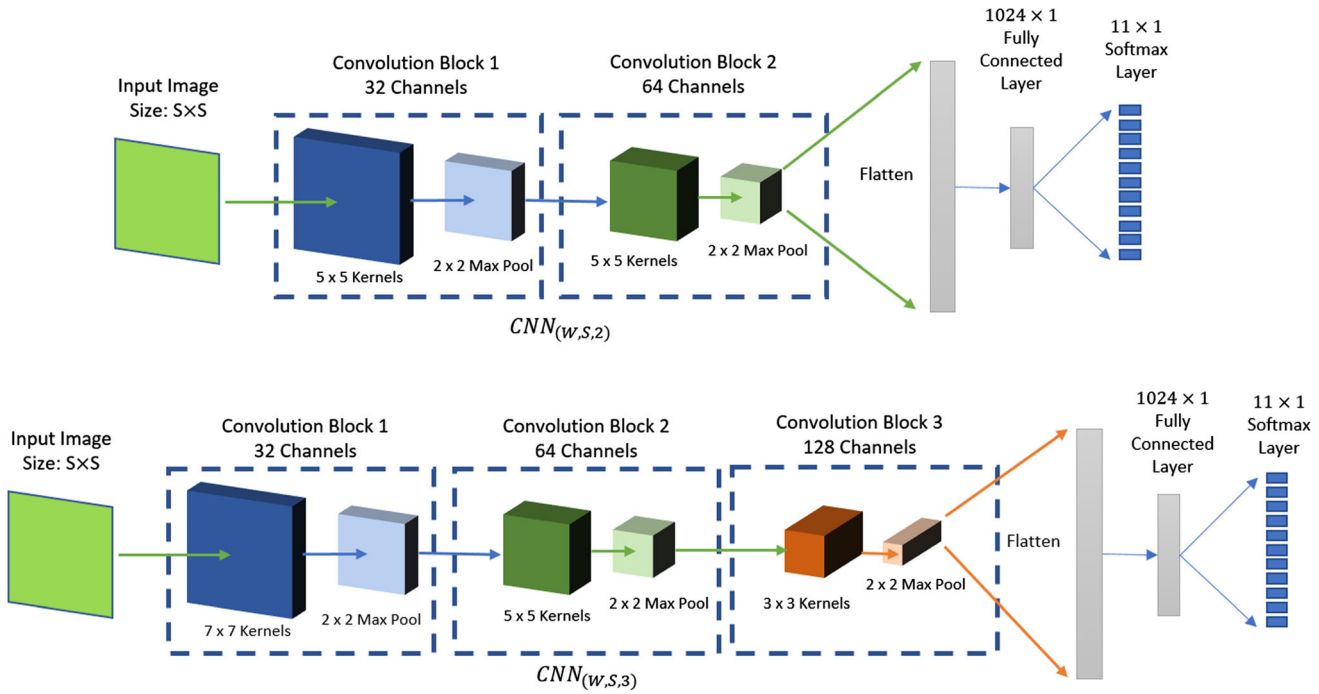

Fig. 4 Architecture of the two- and three-layer CNNs

1. Transformation (W): W refers to whether or not the HWT has been used for having a reconstructed version of the raw image from the seven level detailed components as mentioned in Sect. 2.1. W can take one of the two values: HWT or RAW, where the former one refers to HWT presence and the latter one refers to HWT absence.
2. Scale (S): S refers to the size of the input image. It can take values of 32, 64, 128 representing resolutions 32×32, 64×64 and 128×128, respectively. Input image size will control the intermediate size of the activation maps, and thus the number of weights in the FCL.
3. Depth (D): D refers to the number of convolutional block in our CNN architecture. In our study, one of values: 2 or 3, representing two- or three-levels of convolution and pooling can be used.

An illustration of the various architecture is shown in Fig. 5.

2.4 Justification for variations

While CNNs are more preferred as end-to-end solutions, we have used an ensemble of 12 networks with 3 main variations, namely transformation, scale and depth. One of our main focus is to keep the individual networks small. That creates lots of constraints in the learning capability. Each variation addresses such a constraint and their combination boost the features combination which are relevant to the current problem domain.

2.4.1 Reason for wavelet transformation

Discrete Wavelet Transform (DWT) has been used in our approach as a dimensionality reduction tool. DWT at single level of decomposition performs two operations to reduce the size of the input image, i.e. an averaging function and difference function at each pixel with respect to each of its neighboring pixels. The averaging function generates what is known as an approximation coefficient and the difference creates the detail coefficient. For the purpose of script identification, the contours play a crucial role and hence the detail coefficients are better suited for the purpose of classification, and the approximation coefficients represents a compressed version of the original image. We continue this process to multiple levels of decomposition to capture detail coefficients at higher levels of granularity. The wavelet transformation, in practice, suppresses background noise [31] and separates the text from the background. The increase in homogeneity definitely affects the quality of features learnt. A bilateral filter might have a similar effect, however, since we are using networks that are shallow in nature and operates under constrained resources the network may not have a luxury to learn such a filter. In that case, a forced transformation boosts the performance. Especially in case of problems like script recognition from handwritten text background noise can

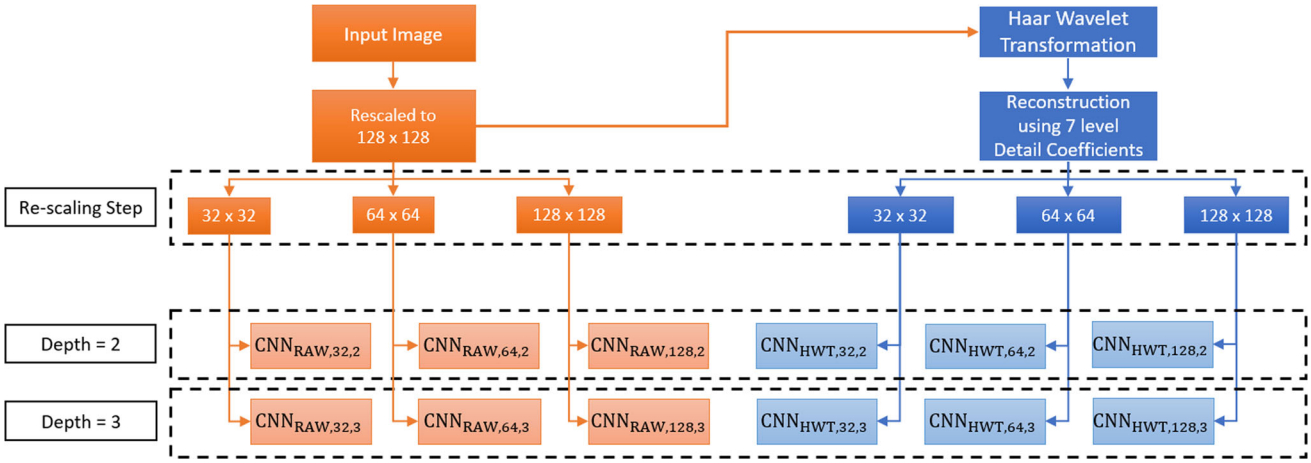

Fig. 5 The different variations in the networks

affect the learning process significantly. Hence, a combination of spatial features with high-frequency components can yield better performance. Some examples of reconstructed images from the detail coefficients of a 7-level Haar wavelet transform are shown in Fig. 6. Here, we can see how the background low-frequency components are removed and the highest information is near the contours of the text.

2.4.2 Reason for different scales of inputs

As CNNs are mostly used in case of extracting the important features from a large set of inputs, using a wide variety of input sizes to the CNNs will lead to extracting varied sets of features which can be used in the classification process. As the filters in the convolution layers of a CNN have a limited view of the input image at a particular layer, varying the size of the input image will thus change the sensory region of the kernels with respect to input and hence the training process is enhanced by using varied input sizes. The sensory area affected by the kernels of the different layers of the two proposed networks with respect to the image of a native resolution of 128×128 is shown in Table 1. To obtain such a wide variety of possible inputs, generally deeper networks are used with more convolutional layers. Due to constraints on computational resources to train extremely deep networks, we have separately trained multiple CNNs of small depth and incrementally ensembled them to achieve a similar result.

2.4.3 Reason behind using multiple networks of different depths

The depth of network mainly defines two things. Firstly, the complexity of features, and secondly the sensory region of the kernel with respect to the input image. Since by varying the scales of the input we have attained a variety of sizes in terms of sensory regions we do not need to build very deep networks. Secondly, script of handwritten words can be recognized by a variety of features of different levels of complexity. While some part of a text may be a simple line or curve, while other parts have a more complicated structure. Hence, networks of two different depths

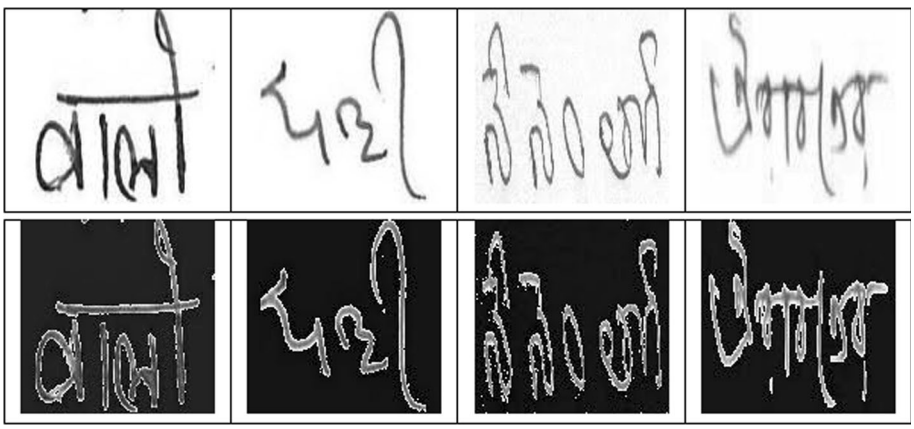

Fig. 6 Input images (top) and their reconstruction (below from detail coefficients of 7-level Haar wavelet transform

Table 1 Sensory area of the kernels with respect to unscaled input

Original scale	Input scale	CNN2		CNN3		
		Layer1	Layer2	Layer1	Layer2	Layer3
128	128	5×5	10×10	7×7	10×10	12×12
128	64	10×10	20×20	14×14	20×20	24×24
128	32	20×20	40×40	28×28	40×40	48×48

are used and combined later to capture both kinds of features.

2.5 Classification

Each of the CNNs (aforementioned) produces an output of a 11 dimensional soft-max probabilities. They can individually be trained. The final classification has been done using two main techniques. The first technique uses ensemble techniques on these 12 individual soft-max outputs, and the second technique uses the layer before the soft-max layer in the individual CNNs. Each CNN has a 1024 dimensional feature vector connecting to the soft-max layer. These 12 separate 1024 dimensional feature vectors were concatenated and plugged as an input to separate MLP that produced an output of the unified 11 soft-max. In what follows, we will discuss them in detail.

2.5.1 Ensemble module

The ensemble module expects as input, vectors of size 11, representing soft-max probabilities (trained CNN for each image) and produces as output the predicted class label. Using features, we then combine the CNNs into a single model using one of the two methods: max-voting scheme or probabilistic voting scheme.

1. Max-voting scheme (CNN*max*): As mentioned before, corresponding to each CNN, we get a vector of length eleven as output which represents the probability scores, and the predicted class label by the CNN is the class which has the maximum probability score from all the scores generated. Of all the predictions made by the ten different CNNs, for a particular image, the one prediction which was made the most number of times is the output predicted class label by our ensemble model. Alternately, if for each CNNW, S, D, we have the output probability scores as $\text{prob}_{\text{W,S,D}}$, then the predicted class by this CNN, $\text{class}_{\text{W,S,D}}$ will be,

$$\text{class}_{\text{W,S,D}} = \underset{i}{\text{argmax}}\left\{\text{prob}_{\text{W,S,D}}[i], \forall i \in [1, 11]\right\}.$$

Each CNN casts its vote as $\text{class}_{\text{W,S,D}}$ for a particular label, and the label with the maximum number of votes is the output of the model. The final class $\text{class}_{\text{final}}$ can then be expressed as:

$$\text{class}_{\text{final}} = \text{mode}(\text{class}_{\text{W,S,D}}).$$

2. Probabilistic voting scheme (CNN*prob*): It is similar to CNN*max*. However, instead of evaluating the class label predicted by each CNN, we perform an element-wise average for each probability score. Alternately, if for each CNNW, S, D, we have the output probability scores as $\text{prob}_{\text{W,S,D}}$, then the average soft-max scores over the ten possible CNNs prob*avg* is,

$$\text{prob}_{\text{avg}} = \frac{1}{N}\sum W, S, D\text{prob}_{\text{W,S,D}},$$

where W $\in \{\text{HWT}, \text{RAW}\}$, S $\in \{32, 64, 128\}$ and D $\in \{2, 3\}$. From the average score, we can then evaluate the predicted class label with the maximum probability score,

$$\text{class}_{\text{final}} = \underset{i}{\text{argmax}}\{\text{prob}avg[i], \forall i \in [1, 11]\}.$$

The aforementioned ensemble schemes are graphically illustrated in Fig. 7.

2.6 CNN as a feature extractor (CNN_{feat})

In second set of experiment, we used each trained CNN for feature extraction. As mentioned in the earlier section, preceding the soft-max layer of each CNN has a FCL with 1024 neurons. This layer can be treated as a feature vector of size 1024. For this method, we concatenated all features, which is then used as an input for MLP. Alternately, for each image, the output feature vector generated by $\text{CNN}_{\text{W,S,D}}$ be $feat_{\text{W,S,D}}$, then the concatenated feature vector can be expressed as:

$$\text{featVector} = \bigcup W, S, D\text{feat}_{\text{W,S,D}},$$

where W $\in \{\text{HWT}, \text{RAW}\}$, S $\in \{32, 64, 128\}$ and D $\in \{2, 3\}$. This feature vector has been fed as an input to MLP that is composed of 4 layers: input layer with 10,240 neurons, first hidden layer with 1024 neurons, second hidden layer with 512 neurons and an output soft-max layer with 11 neurons. Note that each CNN was trained individually beforehand, and the extracted features were used

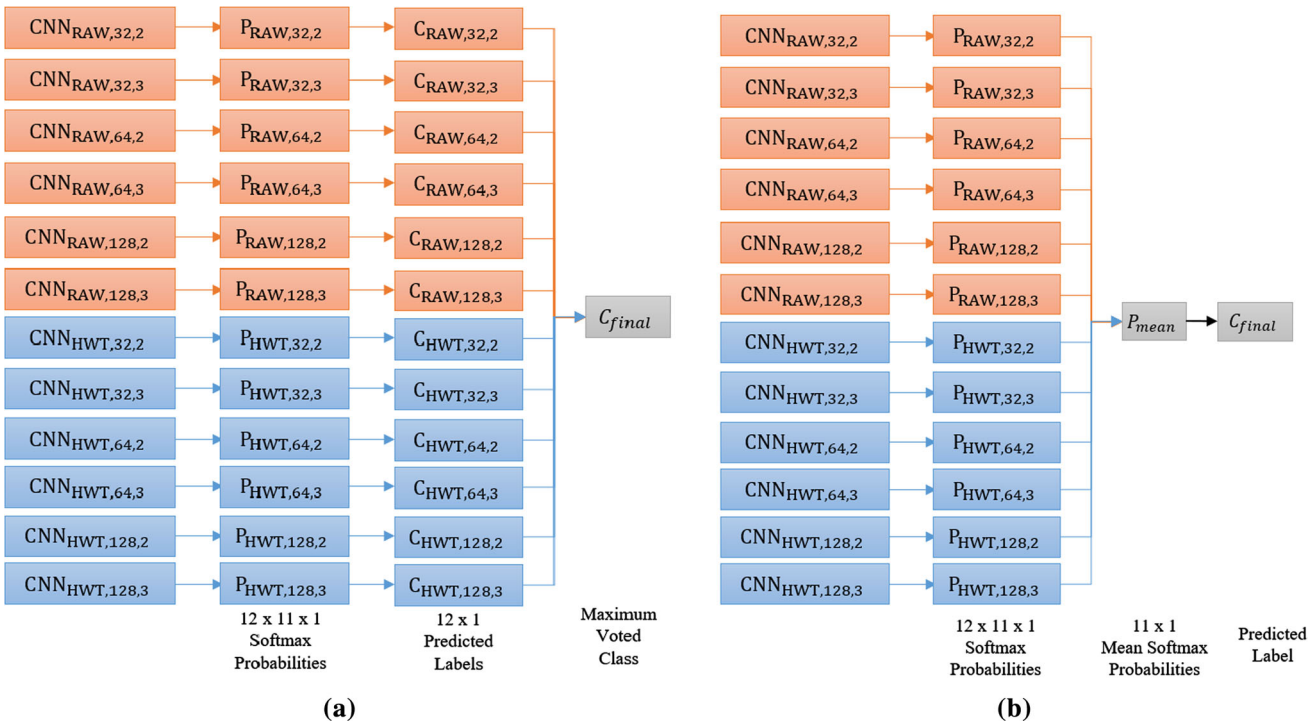

Fig. 7 **a** Max-voting scheme and **b** probabilistic voting scheme are illustrated

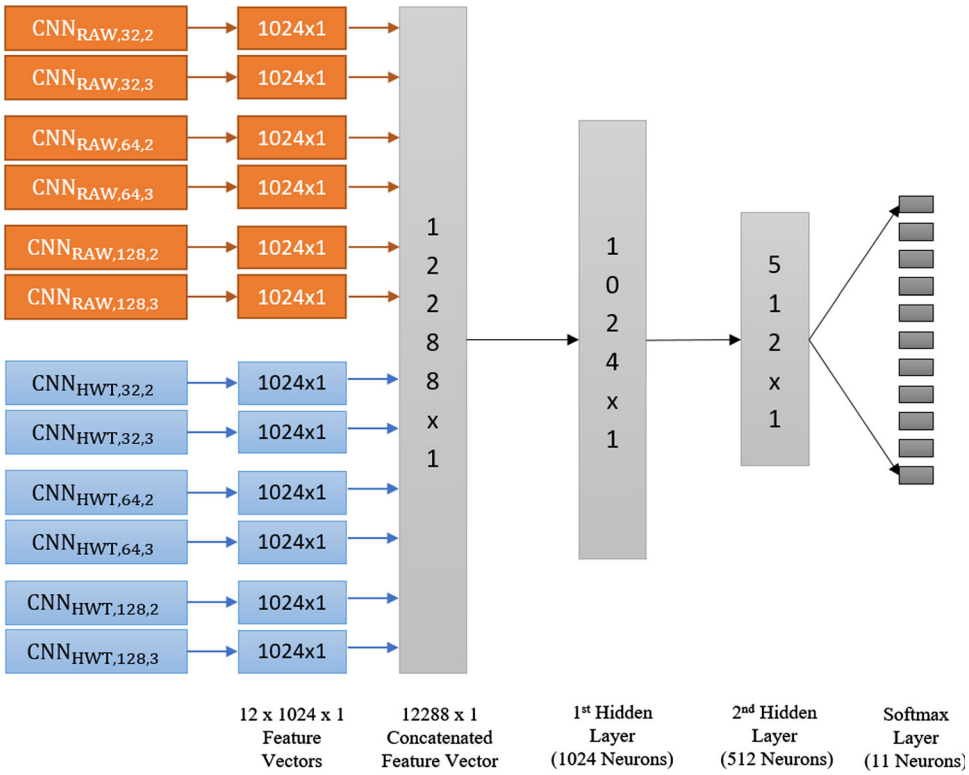

Fig. 8 CNNs can be used as feature extractor: the penultimate layer of the CNNs produces a feature vector of size 1024×1

to train this second MLP separately. For better understanding, a visual representation of our architecture is provided in Fig 8. After MLP has been trained, test features (new test images) can be used to evaluate soft-max scores prob, from which we obtain the predicted label ($\text{class}_{\text{final}}$) as:

$$\text{class}_{\text{final}} = \underset{i}{\text{argmax}}\{\text{prob}[i], \forall i \in [1, 11]\}.$$

3 Experiments

In the previous section, it has been mentioned that a variety of CNN architectures were used. Using the convention as mentioned Sect. 2.3, the parameters of W, S and D will take the values as shown in Table 2.

The first phase of our test justifies the purpose of the individual CNNs by showing how much each of them can contribute. We will perform hierarchical combination of the CNNs to see how each of the parameters individually can affect our performance. Once the parameter selection procedure is justified, we will compare how far ensemble methods boost the peformance.

3.1 Dataset

The dataset we have used to evaluate our approach for word-level script identification, consists of a total of 11,000 words, spread across 11 different Indic scripts: Bangla, Devanagari, Gujarati, Gurumukhi, Kannada, Malayalam, Oriya, Roman, Tamil, Telugu, and Urdu. Grayscale samples were obtained from the PHD_Indic_11 dataset [24]. To generate a separate training and testing sets, we divide our dataset in a 4:1 ratio, with 8800 images in the training set and 2200 images in the testing set. We have five different variations of the dataset that will be inputs for two different architectures of CNN (two- and three-layer). In addition, six variations in scale include 32×32, 64×64 and 128×128 for both raw and wavelet transformed image.

3.2 Training

Using our training set, each of the CNNs in our model ($\text{CNN}_{\text{W,S,D}}$) has been trained separately using images that are provided with or without wavelet transformation (W), at different scale (S) and depths (D). In each $\text{CNN}_{\text{W,S,D}}$, the second FCL has 1024 neurons and at training time we applied dropout to this layer with a probability of 0.5 and added a final output layer with 11 neurons. The output from the final layer is further passed through an eleven way soft-max function that generates class probabilities for each of the 11 different classes, given an input image. The class with the highest probability is the predicted output. For updating weights of the CNNs during back-propagation, we used the Adam optimizer [17] , with learning rate as 1×10^{-4} , β_1 as 0.9 and β_2 as 0.999. Training has been done using batches of 50 images, and training accuracy was calculated as the ratio of successfully classified images to the total number of images in a batch.

Table 2 Possible values of the parameters: W, S and D that define the CNN architecture

Parameters	Values (explanation)
Transformation (W)	HWT (presence of HWT)
	RAW (absence of HWT)
Scale (S)	32 (32×32, input image size)
	64 (64×64, input image size)
	128 (128×128, input image size)
Depth (D)	2 (2 convolutional blocks)
	3 (3 convolutional blocks)

3.3 Experimental setup

We performed two major experimental tests. The first test deals with a parameter selection, and the second test focuses on the combination schemes. To train individual CNNs, we used a machine with Intel Core2Quad Q6600, 4GB RAM, and a NVIDIA GT 730 Graphics card with 4GB of DDR3 VRAM.

1. *Parameter selection:*
   In this setup, we address to measure the performance for each individual CNNs to check their individual contributions. Furthermore, the individual CNNs were hierarchically ensembled to further establish the contribution of each individual parameter. While ensembling a group of individual CNNs the label for the ensembled model was equal to the label with the highest soft-max value across all the individual CNNs in consideration. The first level of ensemble was among CNNs with same input representation (W) and scale (S) but different depth (D). This is denoted later as $\text{CNN}_{(\text{W,S})}$. The second level of combination was for CNNs with same input representation but different input scale or sizes. Here, we have used 3 different scales, namely, 32, 64, and 128. At first two of the three scales have been combined resulting in the models referred to as $\text{CNN}_{(\text{W,S1\_S2})}$, where $S1, S2 \in \{32, 64, 128\}$ such that $S1 \neq S2$. Furthermore all the three scales for a specific transformation were combined to form the model $\text{CNN}_{(\text{W})}$.
2. *Combination schemes:* After CNN architectures are justified, two different ensemble schemes and a conventional feature concatenation scheme (with the use of MLP) were used. We have, (a) max-voting scheme (denoted by CNN_{max}), (b) a probabilistic voting scheme (denoted by CNN_{prob}) and (c) conventional feature concatenation scheme (denoted by

CNN_{feat}). Notethat three ensemble techniques were described in Sect. 2.5.1.

4 Results and analysis

This section involves calculation of accuracy, precision, recall and *f*-measure for all individual CNNs as well as their corresponding hierarchical combinations. The second experiment appends the previous statistics (results) with the accuracy, precision, recall and *f*-measure for all the three combination strategies. The results are provided in Table 3, where individual small CNN modules are denoted by $CNN_{W,S,D}$ along with their hierarchical combinations denoted by $CNN_{W,S}$(Combination across depths), $CNN_{W,S1\_S2}$(Combination of two different scales) and CNN_{W}(Combination across all different scales), and the three combination strategies namely CNN_{max}, CNN_{feat} and CNN_{prob}. Figure 9 shows a graphical representation for better understanding, where we observed that CNN_{prob} performed the best with an accuracy of 95.45%, precision of 95.36%, recall of 95.32% and *f*-measure of 95.33%.

4.1 Analysis

First phase of analysis deals with parameter selection (of the individual architectures), and the second phase discusses the results from three different combination strategies.

4.1.1 Effect of variation

If we look at the trend in Fig. 10, we notice that whenever we combined two or more lower level CNNs (to obtain a higher level), we gained in accuracy. This signifies that each of the individual architecture does have their unique contribution and therefore they are complimentary to each other. $CNN_{W,S}$ shows its gain over the average of all corresponding $CNN_{W,S,D}$, and CNN_{W} shows its gain over the average of all $CNN_{W,S}$. There is however an anomaly when considering partial combination of 2 different scales ($CNN_{W,S1\_S2}$) in case of wavelet transformed inputs. While combining scales 64 and 128 for wavelet transformed images, the performance dropped due to the poor performance of the largest scale in Wavelet transformed images. Finally, CNN_{prob} shows its gain over the average of all CNN_{W}.

Secondly, to see the effect varying depth factor, we take a look into Fig. 11a. While using raw images as input we observed that an increase depth has more effect for larger input image sizes. However, due to loss of information and induced sparsity after HWT, depth has lesser effect for larger input image sizes. The loss of information from HWT has also a negative impact in the accuracy as shown in Fig. 11b. We also observed that for medium sized inputs, specifically 64×64, HWT actually shows a negative loss or a gain in accuracy. However, HWT fails for higher sizes of input image. It can also be seen in Fig. 11c, where it shows an increase in accuracy with the increase in input image size. While raw images tend to perform better with higher resolution (or big size) of inputs, wavelet transformed images performed better with medium sized inputs. The fact can clearly be seen in the extreme fall of accuracy when 128×128 images were considered over 64×64 (in case of wavelet transformed images). In addition, in Fig. 11c, the impact is larger for a deeper architecture if we use raw images and vice-versa for wavelet transformed images. This confirms our observation in Fig. 11a. In short, our observations may be summarized as follows:

1. Architectures with higher depth can handle larger inputs better for raw images;
2. For wavelet transformed images, higher depth has a negative impact on accuracy for larger inputs;
3. Wavelet transformation is particularly useful inputs of medium size and has an adverse effect for larger input sizes;
4. Raw images tend to work well at a larger resolution.

4.1.2 Integrating small CNNs

As shown in Fig. 10, integrating small CNNs has a positive effect on the overall performance. It is interesting to analyze various combination strategies. We have demonstrated the performance of the three combination methodologies: CNN_{max}, CNN_{feat}, and CNN_{prob} in Table 3, and in Fig. 12 (for a closer look). The probabilistic voting technique provides better performance as compared to other schemes: max-voting and feature concatenation. The mean of probabilities works better than an MLP trained on the concatenated features because the twelve networks have been trained individually. The learned features have a lot of redundant information and hence the concatenated feature is not up to the mark. Had the networks been combined and back-propagated against a common loss function the learnt features would have been much better. It is definitely a valid way of training the networks and would also provide good results however our goal is to combine small networks that can be individually trained on cheap hardware with constraints on the available compute capability. The proposed work was carried out using a Nvidia GT 730 with 4GB of DDR3 Memory. The graphics card costs only

Table 3 Accuracy, precision, recall and *f*-measure for all variations of CNNs along with their hierarchical combinations and the three ensemble modules

Architecture	Accuracy	Precision	Recall	F-Measure
$CNN_{RAW,32,2}$	87.5455%	87.4384%	87.3323%	87.3387%
$CNN_{RAW,32,3}$	86.9091%	86.9876%	86.9610%	86.7374%
$CNN_{RAW,64,2}$	85.0455%	85.3179%	84.7427%	84.8816%
$CNN_{RAW,64,3}$	85.6818%	85.8752%	85.5871%	85.5505%
$CNN_{RAW,128,2}$	87.3182%	87.5172%	87.4528%	87.0640%
$CNN_{RAW,128,3}$	90.0000%	89.8828%	89.9499%	89.8364%
$CNN_{HWT,32,2}$	86.3182%	86.4908%	85.9028%	86.0410%
$CNN_{HWT,32,3}$	88.0455%	88.1745%	87.8513%	87.9065%
$CNN_{HWT,64,2}$	91.5000%	91.3082%	91.3473%	91.2758%
$CNN_{HWT,64,3}$	92.1818%	92.1121%	92.0389%	91.9549%
$CNN_{HWT,128,2}$	86.7273%	86.9571%	86.4817%	86.5368%
$CNN_{HWT,128,3}$	75.5455%	79.9510%	74.5490%	74.9420%
$CNN_{RAW,32}$	90.0909%	90.0134%	90.0367%	89.9530%
$CNN_{RAW,64}$	88.5000%	88.5541%	88.4008%	88.4129%
$CNN_{RAW,128}$	91.4091%	91.3026%	91.4039%	91.2335%
$CNN_{HWT,32}$	88.4545%	88.5375%	88.2114%	88.2843%
$CNN_{HWT,64}$	94.2273%	94.0972%	94.1016%	94.0543%
$CNN_{HWT,128}$	86.7273%	87.3288%	86.2867%	86.4852%
$CNN_{RAW,32\_64}$	90.4000%	90.2900%	90.3500%	90.2900%
$CNN_{RAW,64\_128}$	92.4500%	92.3300%	92.4500%	92.3200%
$CNN_{RAW,32\_128}$	91.1800%	91.0900%	91.1000%	91.0500%
$CNN_{HWT,32\_64}$	93.5000%	93.3800%	93.3200%	93.3100%
$CNN_{HWT,64\_128}$	89.0900%	89.4100%	88.6900%	88.8800%
$CNN_{HWT,32\_128}$	93.4500%	93.3600%	93.1800%	93.2300%
CNN_{RAW}	93.6818%	93.6086%	93.6015%	93.5799%
CNN_{HWT}	92.1818%	92.1451%	92.0924%	92.0585%
CNN_{MAX}	95.0000%	94.9456%	94.8896%	94.8893%
CNN_{PROB}	95.4545%	95.3646%	95.3199%	95.3265%
CNN_{FEAT}	94.6818%	94.7328%	94.6288%	94.6127%

*The chromatic scale depicts higher values in green and lower values in red.

around 90 dollars in the current US market. Thus, our proposed system is extremely useful for creating cheap systems. In addition, we have also performed correlation analysis using the Pearson's and Spearman's correlation coefficient. Correlation analysis is performed between the three ensembled models (CNN_{max}, CNN_{feat}, and CNN_{prob})

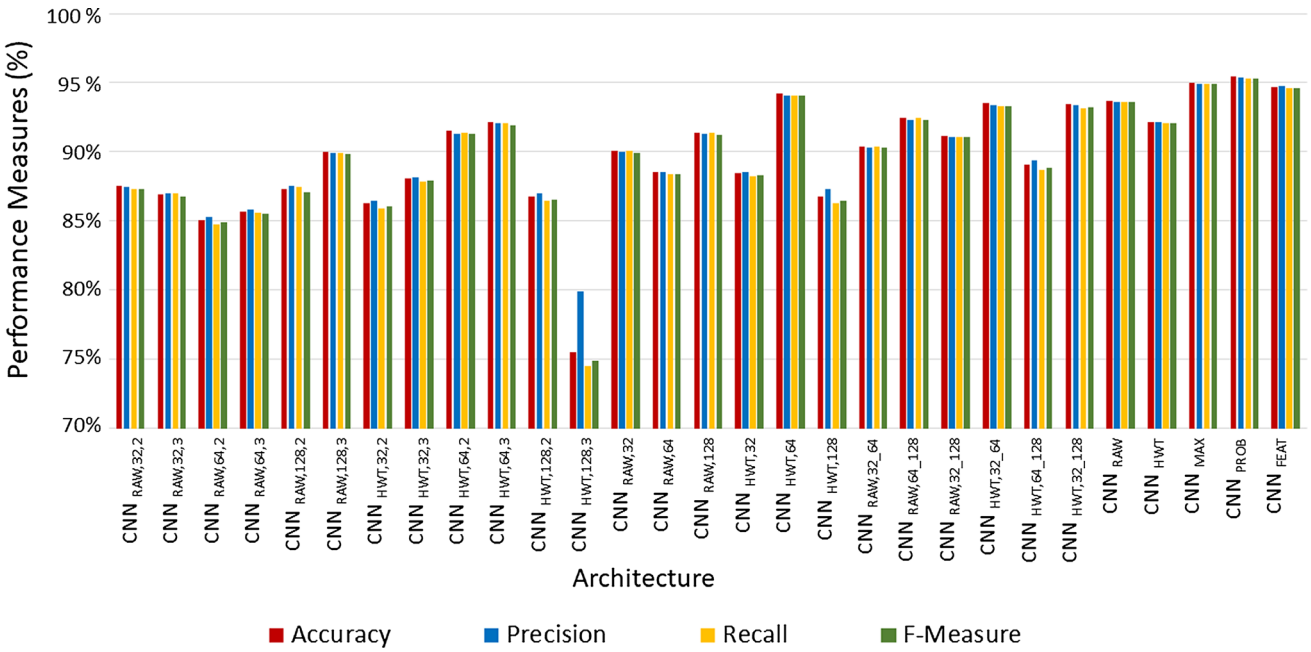

Fig. 9 Performance comparison: accuracy, precision, recall and *f*-measure for different variations of individual CNN modules and ensemble modules

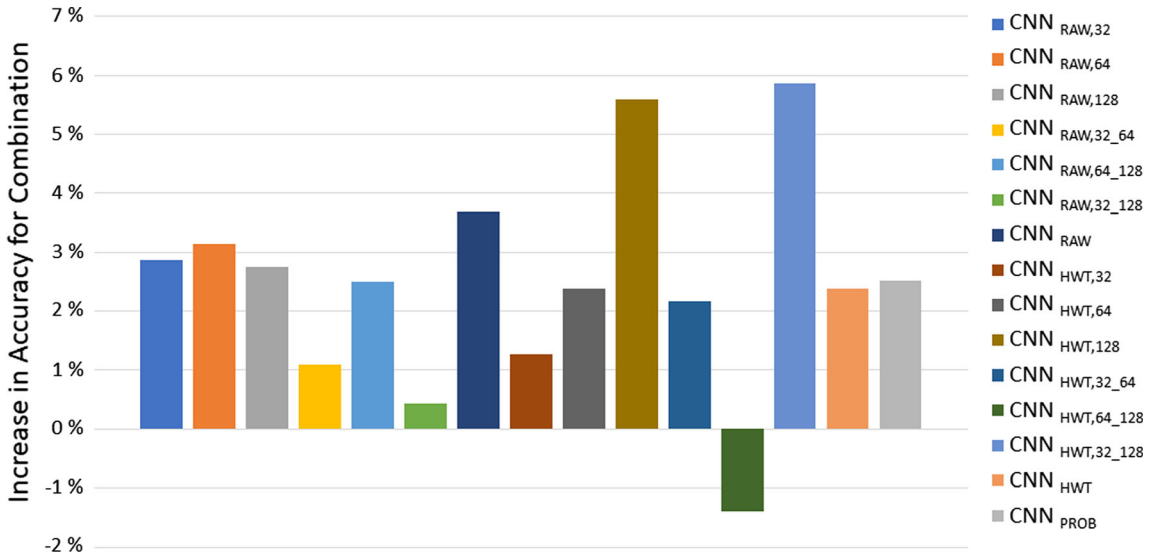

Fig. 10 Performance evaluation (in terms of accuracy) by combining architectures

and the individual CNNs ($CNN_{(W,S,D)}$) used in the ensembling process, to see how alike the different CNNs are in making predictions. The value of the coefficients lies between -1 to $+1$ and the higher the value of the correlation coefficient, the more likely the models in question are likely to predict the same output for a given input image. In Table 4, one can see combined models have much more correlated outputs as compared to the smaller architectures. What this signifies is that the concepts learned by combined networks have reached a much more saturated region as compared to the individual networks. The individual networks have learnt different concepts and hence were open to improvement upon ensembling. Finally, class specific precision, recall and *f*-measure are provided (for CNN_{prob} architecture) in Table 5, where Bangla script proved to be the most difficult script to recognize, while Telugu stood easiest i.e., better recognition performance.

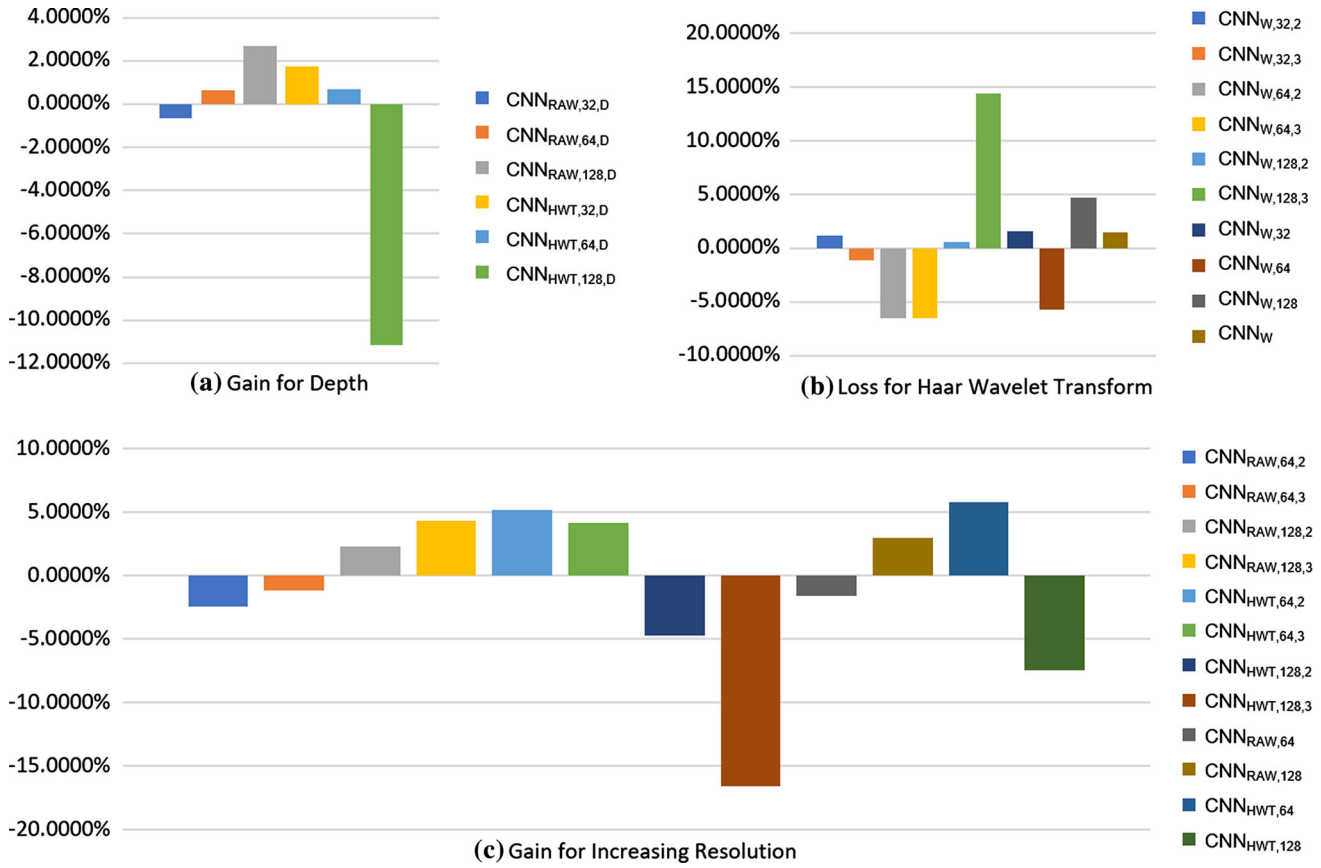

Fig. 11 Effect of variation of parameters, W,S,D 2: **a** gain of a 3 layered network over a 2 layered network; **b** Loss due to application of Haar wavelet transformation; and **c** gain (in accuracy) for a CNN with respect to the immediate smaller resolution

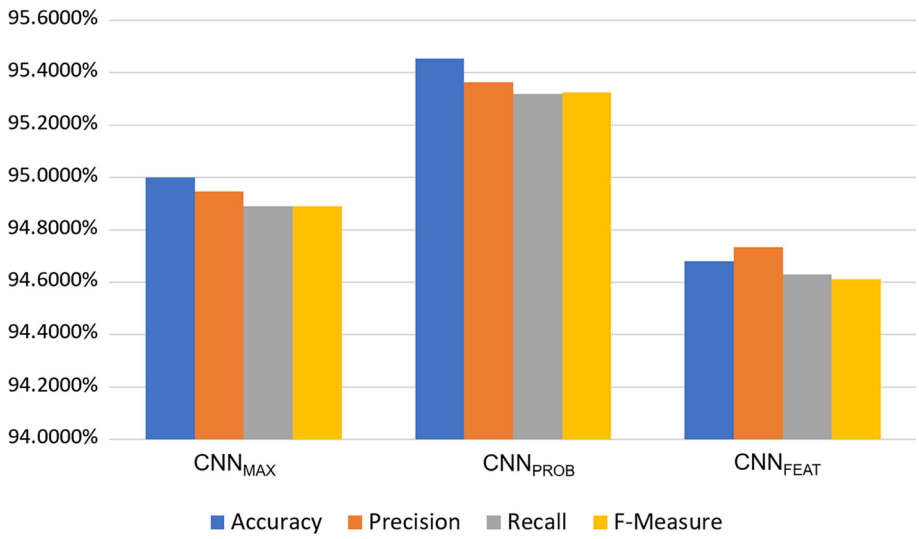

Fig. 12 Performance evaluation: accuracy, precision, recall and *f*-measure for all three combination strategies: CNN_{max}, CNN_{prob} and CNN_{feat}

4.2 Comparative study

Till now the performance of our system has been theoretically justified. Further, the proposed system is compared with the baseline algorithm reported in [24]. Note that the baseline algorithm relied on feature-based approaches. The algorithm has been applied on the currently used cropped word dataset for a fair comparison. In addition, we have shown our superiority over the LeNet and the AlexNet that have proven track record for performing on handwritten

Table 4 Pearson's and Spearman correlation coefficients[a] for all individual small CNNs and the three combined CNNs with respect to the three combined CNNs

	Pearson's Correlation Coefficient			Spearman's Correlation Coefficient		
Architecture	CNN_{MAX}	CNN_{PROB}	CNN_{FEAT}	CNN_{MAX}	CNN_{PROB}	CNN_{FEAT}
$CNN_{RAW,32,2}$	0.8840	0.8780	0.8630	0.8840	0.8780	0.8630
$CNN_{RAW,32,3}$	0.8760	0.8710	0.8440	0.8760	0.8710	0.8440
$CNN_{RAW,64,2}$	0.8545	0.8510	0.8330	0.8545	0.8510	0.8330
$CNN_{RAW,64,3}$	0.8555	0.8545	0.8445	0.8555	0.8545	0.8445
$CNN_{RAW,128,2}$	0.8765	0.8695	0.8620	0.8765	0.8695	0.8620
$CNN_{RAW,128,3}$	0.9040	0.8960	0.8880	0.9040	0.8960	0.8880
$CNN_{HWT,32,2}$	0.8645	0.8705	0.8645	0.8645	0.8705	0.8645
$CNN_{HWT,32,3}$	0.8800	0.8895	0.8725	0.8800	0.8895	0.8725
$CNN_{HWT,64,2}$	0.9335	0.9350	0.9170	0.9335	0.9350	0.9170
$CNN_{HWT,64,3}$	0.9170	0.9225	0.9075	0.9170	0.9225	0.9075
$CNN_{HWT,128,2}$	0.8740	0.8745	0.8620	0.8740	0.8745	0.8620
$CNN_{HWT,128,3}$	0.7450	0.7420	0.7315	0.7450	0.7420	0.7315
CNN_{MAX}	1.0000	0.9830	0.9440	1.0000	0.9830	0.9440
CNN_{PROB}	0.9830	1.0000	0.9500	0.9830	1.0000	0.9500
CNN_{FEAT}	0.9440	0.9500	1.0000	0.9440	0.9500	1.0000

[a]The chromatic scale depicts higher values in green and lower values in red

Table 5 Performance evaluation[a]: precision, recall and *f*-measure for each class using combination strategy: CNN_{prob}

Language	Precision	Recall	F-Measure
Bangla	0.8857143	0.8908046	0.8882521
Devanagari	0.9415205	0.90960452	0.9252874
Gujarati	0.9858491	0.98584906	0.9858491
Gurumukhi	0.947619	0.9255814	0.9364706
Kannada	0.9766355	0.9952381	0.9858491
Malayalam	0.9230769	0.93150685	0.9272727
Oriya	0.9253731	0.97894737	0.9514066
Roman	0.9842105	0.94923858	0.9664083
Tamil	0.9540816	0.93969849	0.9468354
Telugu	0.9820628	1	0.9909502
Urdu	0.9839572	0.9787234	0.9813333

[a]The chromatic scale depicts higher values in green and lower values in yellow

datasets. While performing analysis on the currently used dataset, using AlexNet and Lenet, we have resized the images in our datasets to conform to the expected input size of both the above mentioned models respectively. Our individually trainable modules are more comparable to the LeNet in terms of size. Though considerably larger, the AlexNet fails to beat our proposed ensemble approach. Table 6 shows how our approach attests the fact.

Table 6 Comparison with baseline method and state-of-the-art algorithms

Approach	Accuracy (%)
Obaidullah et al. 2017 [24]	91.00
LeNet 1998 [20]	82.00
AlexNet 2012 [18]	92.14
Our method (CNN_{max})	95.00
Our method (CNN_{prob})	95.45
Our method (CNN_{feat})	94.68

5 Conclusion

We have presented an improved word-level handwritten Indic script identification technique, where a set of small convolutional neural networks are combined. In our CNN's architecture, individually trainable modules that vary with respect to the input image size, CNN's depth and wavelet transformation have been employed as three different levels of variation. We have implemented several ensemble strategies such as *max-voting* and *probabilistic voting* and conventional approaches like feature concatenation. Thanks to small individually trainable CNNs, unlike conventional approaches, we have used a GTX 730, with 4GB of DDR3 VRAM for training, which is typically a household level GPU. Using publicly available dataset of size 11K words (1K per script) from 11 different Indic Scripts:

Bangla, Devanagari, Gujarati, Gurumukhi, Kannada, Malayalam, Oriya, Roman, Tamil, Telugu and Urdu, we have achieved a maximum script identification accuracy of 95.04%. Our performance outperforms the accuracy of the state-of-the-art techniques like AlexNet by 2.9% and more importantly, benchmark techniques (for script identification) on the dataset by more than 4%.

Acknowledgement This work is supported by the project order no. SB/S3/EECE/054/2016, dated 25/11/2016, sponsored by SERB (Government of India) and carried out at the Centre for Microprocessor Application for Training Education and Research, CSE Department, Jadavpur University, Kolkata, India.

References

1. Ahmed SB, Naz S, Razzak MI, Rashid SF, Afzal MZ, Breuel TM (2016) Evaluation of cursive and non-cursive scripts using recurrent neural networks. Neural Comput Appl 27(3):603–613
2. Anil R, Manjusha K, Kumar SS, Soman K (2015) Convolutional neural networks for the recognition of Malayalam characters. In: Proceedings of the 3rd international conference on frontiers of intelligent computing: theory and applications (FICTA) 2014. Springer, pp 493–500
3. Basu S, Das N, Sarkar R, Kundu M, Nasipuri M, Basu DK (2009) A hierarchical approach to recognition of handwritten bangla characters. Pattern Recognit 42(7):1467–1484
4. Bhattacharya U, Chaudhuri B (2005) Databases for research on recognition of handwritten characters of Indian scripts. In: Eighth international conference on document analysis and recognition, 2005. Proceedings. IEEE, pp 789–793
5. Bhattacharya U, Chaudhuri BB (2009) Handwritten numeral databases of Indian scripts and multistage recognition of mixed numerals. IEEE Trans Pattern Anal Mach Intell 31(3):444–457
6. Bracewell RN, Bracewell RN (1986) The Fourier transform and its applications, vol 31999. McGraw-Hill, New York
7. Brodić D, Amelio A, Milivojević ZN (2016) Language discrimination by texture analysis of the image corresponding to the text. Neural Comput Appl 29:1–22
8. Busch A, Boles WW, Sridharan S (2005) Texture for script identification. IEEE Trans Pattern Anal Mach Intell 27(11):1720–1732
9. Das N, Sarkar R, Basu S, Saha PK, Kundu M, Nasipuri M (2015) Handwritten bangla character recognition using a soft computing paradigm embedded in two pass approach. Pattern Recognit 48(6):2054–2071
10. Daubechies I (1990) The wavelet transform, time-frequency localization and signal analysis. IEEE Trans Inf Theory 36(5):961–1005
11. Dhanya D, Ramakrishnan A, Pati PB (2002) Script identification in printed bilingual documents. Sadhana 27(1):73–82
12. Garain U, Chakraborty M, Dasgupta D (2006) Recognition of handwritten indic script using clonal selection algorithm. In: Artificial immune systems, pp 256–266
13. Ghosh D, Dube T, Shivaprasad A (2010) Script recognition a review. IEEE Trans Pattern Anal Mach Intell 32(12):2142–2161
14. Govindaraju V, Setlur S (2009) Guide to OCR for indic scripts. Springer, Berlin
15. Hangarge M, Santosh K, Pardeshi R (2013) Directional discrete cosine transform for handwritten script identification. In: 2013 12th International conference on document analysis and recognition (ICDAR). IEEE, pp 344–348
16. John J, Pramod K, Balakrishnan K (2012) Unconstrained handwritten Malayalam character recognition using wavelet transform and support vector machine classifier. Procedia Eng 30:598–605
17. Kingma D, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
18. Krizhevsky A, Sutskever I, Hinton G.E (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
19. LeCun Y (1998) The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/
20. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
21. Mehrotra K, Jetley S, Deshmukh A, Belhe S (2013) Unconstrained handwritten devanagari character recognition using convolutional neural networks. In: Proceedings of the 4th international workshop on multilingual OCR. ACM, p 15
22. Neeba N, Jawahar C (2009) Empirical evaluation of character classification schemes. In: Seventh international conference on advances in pattern recognition, 2009. ICAPR'09. IEEE, pp 310–313
23. Obaidullah SM, Das N, Halder C, Roy K (2015) Indic script identification from handwritten document images an unconstrained block-level approach. In: 2015 IEEE 2nd international conference on recent trends in information systems (ReTIS). IEEE, pp 213–218
24. Obaidullah SM, Halder C, Santosh K, Das N, Roy K (2017) Phdindic_11: page-level handwritten document image dataset of 11 official indic scripts for script identification. Multimedia Tools Appl 77:1–36
25. Obaidullah SM, Santosh K, Halder C, Das N, Roy K (2017) Automatic indic script identification from handwritten documents: page, block, line and word-level approach. Int J Mach Learn Cybern 10:1–20
26. Pal U, Chaudhuri B (2004) Indian script character recognition: a survey. Pattern Recognit 37(9):1887–1899
27. Pal U, Jayadevan R, Sharma N (2012) Handwriting recognition in indian regional scripts: a survey of offline techniques. ACM Trans Asian Lang Inf Process (TALIP) 11(1):1
28. Pal U, Sinha S, Chaudhuri B (2003) Multi-script line identification from indian documents. In: Seventh international conference on document analysis and recognition, 2003. Proceedings. IEEE, pp 880–884
29. Pati PB, Ramakrishnan A (2008) Word level multi-script identification. Pattern Recogn Lett 29(9):1218–1229
30. Portnoff M (1980) Time-frequency representation of digital signals and systems based on short-time fourier analysis. IEEE Trans Acoust Speech Signal Process 28(1):55–69
31. Porwik P, Lisowska A (2004) The haar-wavelet transform in digital image processing: its status and achievements. Mach Graph Vis 13(1/2):79–98
32. Rajput G, Anita H (2013) Handwritten script recognition at line level-a multiple feature based approach. Int J Eng Innovative Technol 3(4):90–95
33. Rani R, Dhir R, Lehal GS (2013) Script identification of presegmented multi-font characters and digits. In: 2013 12th International conference on document analysis and recognition (ICDAR). IEEE, pp 1150–1154
34. Roy K, Das S.K, Obaidullah SM (2011) Script identification from handwritten document. In: 2011 Third national conference on computer vision, pattern recognition, image processing and graphics (NCVPRIPG). IEEE, pp 66–69
35. Roy S, Das N, Kundu M, Nasipuri M (2017) Handwritten isolated bangla compound character recognition: a new benchmark using a novel deep learning approach. Pattern Recogn Lett 90:15–21

36. Sarkhel R, Das N, Das A, Kundu M, 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 Recognit 71:78–93
37. Schenkel M, Guyon I, Henderson D (1995) On-line cursive script recognition using time-delay neural networks and hidden markov models. Mach Vis Appl 8(4):215–223
38. Sharma MK, Dhaka VP (2016) Pixel plot and trace based segmentation method for bilingual handwritten scripts using feed-forward neural network. Neural Comput Appl 27(7):1817–1829
39. Singh PK, Mondal A, Bhowmik S, Sarkar R, Nasipuri M (2015) Word-level script identification from handwritten multi-script documents. In: Proceedings of the 3rd international conference on frontiers of intelligent computing: theory and applications (FICTA) 2014. Springer, pp 551–558
40. Singh PK, Sarkar R, Nasipuri M (2015) Offline script identification from multilingual indic-script documents: a state-of-the-art. Comput Sci Rev 15:1–28
41. Singh PK, Sarkar R, Nasipuri M, Doermann D (2015) Word-level script identification for handwritten indic scripts. In: 2015 13th International conference on document analysis and recognition (ICDAR). IEEE, pp 1106–1110
42. Smith S (1997) Fourier transform properties. The scientist and engineers guide to digital signal processing. California Technical Publishing, San Diego, pp 185–208
43. Stanković RS, Falkowski BJ (2003) The haar wavelet transform: its status and achievements. Comput Electr Eng 29(1):25–44
44. Ubul K, Tursun G, Aysa A, Impedovo D, Pirlo G, Yibulayin T (2017) Script identification of multi-script documents: a survey. IEEE Access 5:6546–6559
45. Verma K, Sharma RK (2016) Comparison of HMM-and SVM-based stroke classifiers for Gurmukhi script. Neural Comput Appl 28:1–13