IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 July 2022, accepted 3 August 2022, date of publication 29 August 2022, date of current version 9 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3202893

TOPICAL REVIEW

Two Decades of Bengali Handwritten Digit Recognition: A Survey

A. B. M. ASHIKUR RAHMAN, MD. BAKHTIAR HASAN, SABBIR AHMED, TASNIM AHMED, MD. HAMJAJUL ASHMAFEE, MOHAMMAD RIDWAN KABIR, AND MD. HASANUL KABIR, (Member, IEEE)

Department of Computer Science and Engineering, Islamic University of Technology, Dhaka 1704, Bangladesh

Corresponding author: A. B. M. Ashikur Rahman (ashikiut@iut-dhaka.edu)

ABSTRACT Handwritten Digit Recognition (HDR) is one of the most challenging tasks in the domain of Optical Character Recognition (OCR). Irrespective of language, there are some inherent challenges of HDR, which mostly arise due to the variations in writing styles across individuals, writing medium and environment, inability to maintain the same strokes while writing any digit repeatedly, etc. In addition to that, the structural complexities of the digits of a particular language may lead to ambiguous scenarios of HDR. Over the years, researchers have developed numerous offline and online HDR pipelines, where different image processing techniques are combined with traditional Machine Learning (ML)-based and/or Deep Learning (DL)-based architectures. Although evidence of extensive review studies on HDR exists in the literature for languages, such as English, Arabic, Indian, Farsi, Chinese, etc., few surveys on Bengali HDR (BHDR) can be found, which lack a comprehensive analysis of the challenges, the underlying recognition process, and possible future directions. In this paper, the characteristics and inherent ambiguities of Bengali handwritten digits along with a comprehensive insight of two decades of state-of-the-art datasets and approaches towards offline BHDR have been analyzed. Furthermore, several real-life application-specific studies, which involve BHDR, have also been discussed in detail. This paper will also serve as a compendium for researchers interested in the science behind offline BHDR, instigating the exploration of newer avenues of relevant research that may further lead to better offline recognition of Bengali handwritten digits in different application areas.

INDEX TERMS Digit classification, handwritten numeral recognition, applications of handwritten digit recognition, Bengali handwritten digit dataset, handwriting digit classification review.

I. INTRODUCTION

Communication between two or more human beings is defined as an exchange of information on a particular topic of interest, irrespective of their language, dialect, and cultural differences. Human communication has three basic modes: *written*, *verbal*, or *sign-language*.

Written means of communication include different symbols, such as: *letters*, *numerals*, and *special characters*, combinations of which are used for visualizing words and sentences in human speech, conveying meaningful messages similar to communicating *verbally*, or through *braille* and/or *sign language* [1]. Consequently, handwriting has been a ubiquitous means of communication and permanent information storage. However, these symbols are different across languages, and with time, different scripts and symbols belonging to the same language have emerged. For example, the Indian script itself has 9 major regional scripts: Bengali, Gujarati, Gurumukhi, Kannada, Malayalam, Nastaliq, Oriya, Tamil, and Telugu [2].

Written communication is mostly facilitated through *letters*, *digits*, *symbols*, and *special characters* in printed form using printing devices or in handwritten form using pen and paper, across different scripts and dialects of any language. However, with the evolution of technology, writing mediums and tools (digital tablets, touch screen PCs, etc.) have also

The associate editor coordinating the review of this manuscript and approving it for publication was Li Zhang.

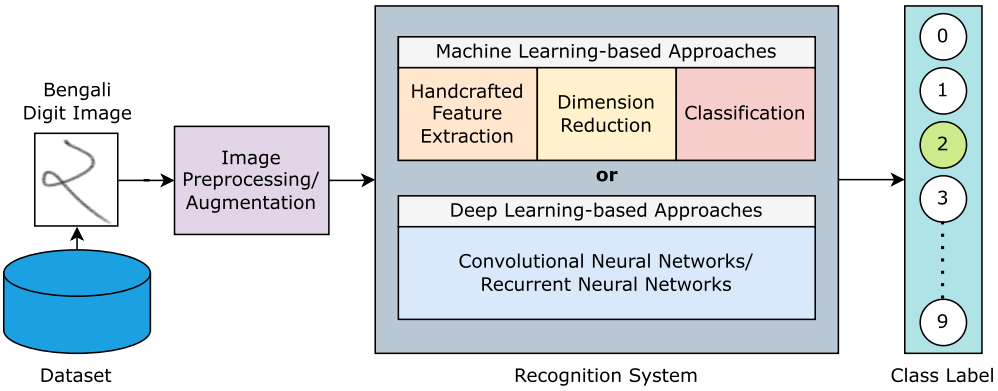


FIGURE 1. Overview of the Bengali Handwritten Digit Recognition Pipeline.

been developed for digitally capturing human handwriting, apart from the pen and paper method. Although the unique shapes of the constituent symbols of different scripts can be maintained exactly when typed in using a computer, it is not the case when they are handwritten. This is because every individual has a unique writing style, which is often dependent on their state of mind, writing environment, writing medium, etc. [3]. Due to these variations and ubiquitous applications, Handwritten Digit Recognition (HDR) has emerged as one of the most important research topics in the field of Optical Character Recognition (OCR) [4]. The purpose of HDR systems is to encode handwritten digits (0-9 in English) into a computer interpretable format for creating a digital footprint of the information [5], [6]. Even with the advances in computing technologies capable of achieving things that were previously considered impossible, the digit recognition process poses its own challenges. As evident from [3], [4], different factors such as the non-uniformity of the digits (size, shape, thickness, orientation, etc.), noise, and most importantly, the variation of writing styles from person to person introduce complexities in the process of HDR.

Researchers have applied HDR systems for recognizing digits of different languages using a combination of image processing tools with Machine Learning (ML) and/or Deep Learning (DL) techniques. Among the traditional ML techniques, Support Vector Machine (SVM), Genetic Algorithms, Decision Tree (DT), Random Forest (RF), k-Nearest Neighbor (kNN), Hidden Markov Models (HMM), etc. are noteworthy. On the other hand, DL techniques include Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Long Short–Term Memory (LSTM) network, Generative Adversarial Network (GAN), etc. [7], [8], [9], [10], [11], [12], [13].

Researchers from different origins have conducted extensive reviews on how different HDR systems have been used in the recognition of either offline (using scanned images) or online (using real-time input) handwritten digits of different scripts such as: English [10], [12], [14], [15], Chinese [16], Arabic [16], Indian Scripts [2], [16], [17], [18], Farsi [13], and Roman [16] summarizing methodologies, tools, feature selection, and results. Compared to other languages, developing a robust Bengali Handwritten Digit Recognition (BHDR) pipeline, given the inherent morphological complexity of Bengali digits, remains a challenging task. However, the challenges of the intended task vary across researchers based on the adopted research methodologies, and most importantly, the datasets. To the best of our knowledge, there is a significant lack of review-based works on BHDR. Therefore, with an aim to clearly depict the progress of research in this domain along with providing useful directions for researchers in the future, we investigated the status quo of the existing literature on BHDR published in the last 20 years. Based on our findings, our contributions are as follows:

1) Highlighted the characteristics of existing datasets on Bengali handwritten digits, exploited in different ML and DL-based frameworks.
2) Provided an in-depth analysis of the existing data preprocessing and resampling techniques, feature extraction, and classification methods using ML and DL approach suitable for BHDR.
3) Summarized the methodologies to find out the strengths and weaknesses of the existing BHDR-related works.
4) Explored the broader scopes of different BHDR systems to provide a guideline for applying them in real-life scenarios.
5) Created a reference for future researchers, allowing them to identify the challenges, and limitations, and provide research directions in the development of robust BHDR pipelines.

To structure our discussion, we have divided the BHDR pipeline into multiple parts as shown in Figure 1. Based on the division, the remaining sections are structured as follows: Section II discusses the complex morphological characters of Bengali digits. Section III provides an overview of the datasets available for Bengali handwritten digit recognition. Then we go through the status quo of digit recognition research discussing preprocessing (Section IV) and augmentation (Section V), earlier Machine Learning-based approaches (Section VI), and recent shift to Deep Learning (Section VII). After that, we assess the studies related to Bengali handwritten digit recognition beyond the conventional

TABLE 1. Different representations of the base digits of Bengali numerals.

Bangla Numeral	Corresponding Arabic Numeral	Standard Bangla Word	Romanized Bangla Word	Handwritten Example
০	0	শূন্য	shunnô	
১	1	এক	æk	
২	2	দুই	dui	
৩	3	তিন	tin	
৪	4	চার	char	
৫	5	পাঁচ	pãch	
৬	6	ছয়	chhôy	
৭	7	সাত	shat	
৮	8	আট	aṭ	
৯	9	নয়	nôy	

classification task in Section VIII. After in-depth analysis, we identified the research gaps and provided some future directions in section IX. Finally, Section X has drawn a conclusion to the work.

II. CHARACTERISTICS OF BENGALI DIGITS

Bengali, spoken by around 272 million people, is the second and the sixth most popular language in India and the world, respectively [19]. Even though it is an ancient Indo-Aryans Language [20], the representations of different Bengali numerals, as we know of and use them today, are derived from the Hindu-Arabic Numeral System [21].

Compared to the Arabic numerals, representations of the numerals *four* and *eight* in Bengali and Arabic scripts, respectively, are similar in nature [21]. The same can be stated for the representation of the numerals *zero*, *two* and *seven* in both scripts [22]. The representations of the digits "0" to "9" in both scripts have been summarized in Table 1 for better comprehension of such similarities.

The style of writing depends on various factors, such as state of mind, writing environment, writing medium, etc. It is intuitive that a person will not always be able to write a particular digit in the same manner. In other words, it is not possible to maintain uniformity in the size, shape, and orientation of a handwritten digit each time, the same or a different person writes it [20]. In connection to this, unlike the Arabic numerals, several Bengali numerals share almost similar representations, which, if not written properly, may be misclassified as different numerals. Therefore, it is worth mentioning that multiple ambiguities may appear in a single handwritten Bengali numeral. A few potential factors behind such ambiguities resulting in misclassifications are discussed below:

A. SIMILAR BASELINE SKELETON, WITH MINOR DIFFERENTIATING STROKES

Some of the digits of Bengali numerals share a similar baseline structure, where the difference is only a small stroke. Accidental strokes can make such pairs look alike, which is hard to differentiate, even for humans. For example, the digits *one* and *two* in Bengali have the same skeleton, with the digit *two* having an extra stroke at the bottom, compared to *one*. If these two numerals are not written properly, the BHDR model could suffer from misclassification. Few other pairs of such sort are: *five* and *six*, *one* and *nine*, *zero* and *three*, etc. An illustration showing the ambiguity can be seen in Figure 2.

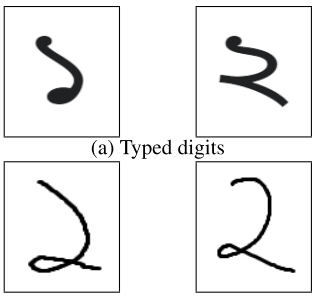


FIGURE 2. Bengali digits 1 and 2 having similar baseline skeleton with minor differentiating strokes.

B. SIMILAR BASELINE SKELETON, WITH DIFFERENT ORIENTATIONS

Real-life handwriting may contain slanted samples oriented in random directions. Even if a digit is properly written, a change in the orientation can make it very confusing for a machine to recognize. For example, the numeral *eight* in Bengali, may be considered to share a similar baseline skeleton with the numeral *six*. If the numeral *six* is written in a way that appears to be rotated, it might lead to misclassification. An illustration showing the ambiguity can be seen in Figure 3.

C. INCOMPLETE STROKES

The numeral *three* in Bengali is characterized by a circular shape at the top and a curve emanating from that shape to

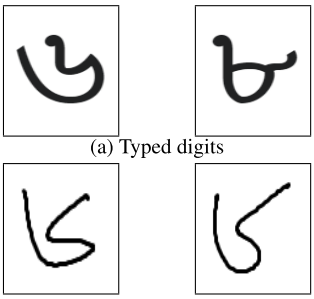

(a) Typed digits
(b) Handwritten digits

FIGURE 3. Bengali digits 6 and 8 having similar baseline skeleton with minor differentiating orientation.

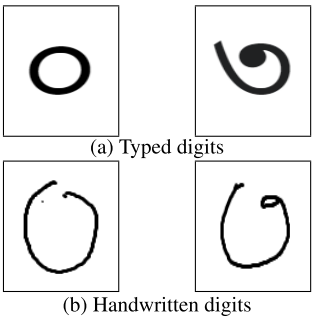

(a) Typed digits
(b) Handwritten digits

FIGURE 4. Bengali digits 0 and 3 having similarity due to incomplete stroke.

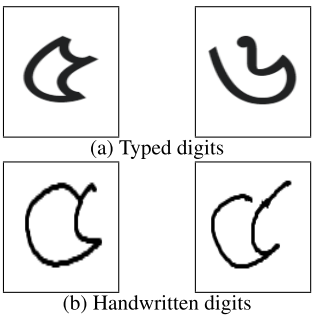

(a) Typed digits
(b) Handwritten digits

FIGURE 5. Bengali digits 5 and 6 having similarity due to extended stroke.

form an incomplete ellipse. Again, the numeral *zero* is simply a circular shape. If *zero* is written in an incomplete manner, i.e. the circle is not closed and if *three* is written without the initial circular shape, it could be challenging for a classifier to differentiate between these two. The numeral pairs, such as: *five* and *six*, *zero* and *seven*, etc. may also suffer from the issue of incomplete stroke. An illustration showing the ambiguity can be seen in Figure 4.

D. REDUNDANT OR EXTENDED STROKES

If *three* is written in a manner that the curved line emanating from the circular shape forms a closed loop, the BHDR model might confuse it with *zero*. The numerals *five* and *six* may also suffer from the issue of redundant or extended stroke. An illustration showing the ambiguity can be seen in Figure 5.

III. DATASETS

Well-organized datasets are always important for evaluating and benchmarking different methods and they should be consistent with certain quality and quantity standards. Besides, a relevant and well-balanced dataset is a must for smoother and faster training and better recognition [23]. A number of standard datasets are available for Bengali handwritten digits. Some of the datasets are created by taking samples using a specified form from a writer base. Banglalekha-Isolated (BLI) [24], NumtaDB [25], Ekush [26], Bengali Handwritten Numerals Dataset (BHaND) [27], etc. are examples of this kind. In these datasets, along with the class labels, information regarding the gender, age of the writer, etc. are also available. These datasets are structured and samples are of uniform size and class distribution is balanced.

Another type of dataset is generated from real-life handwritten documents such as forms, postcards, records, etc. As the sources are heterogeneous, writer information is often not available. However, this type of dataset provides more challenges in digit recognition. Some examples of this kind are ISI Handwritten Bangla Numeral (ISI-HBN) [28], CMATERdb [29], [30], etc.

The distribution of samples per class for each dataset is shown in Figure 6.

A. CMATERdb

One of the earliest datasets for Bengali handwritten numerals is CMATERdb, which was created at the Center for Microprocessor Applications for Training Education and Research (CMATER) research lab, Jadavpur University, India. The original dataset contains many samples of handwritten documents. Along with different handwritten characters, it contains numerals for 3 different languages: Bengali, Devanagari, and Tamil. Among the different versions of the dataset, CMATERdb 3.1.1 contains Bengali handwritten numerals consisting of a total of 6000 images: 600 RGB color images of size 32×32 per class.

B. ISI HANDWRITTEN BANGLA NUMERAL (ISI-HBN)

ISI Handwritten Bangla Numeral dataset was created at the Computer Vision and Pattern Recognition Unit laboratory of the Indian Statistical Institute (ISI), Kolkata. It has both online and offline samples for isolated handwritten digits. The offline samples are labeled and stored in TIFF image format. It has more than 23,000 samples in total, among those, around 19,000 are for training and 4,000 samples are for testing.

C. NumtaDB

NumtaDB is one of the most popular datasets in Bengali handwritten digit recognition, developed by Bengali-AI in 2018 [31]. This dataset is a combination of six different databases which are labeled 'a' to 'f'. It contains more than 85000 images from around 2700 writers both males and females with a ratio of roughly 60%/40%. All the sources have separate training and testing partitions in such a way that samples from the same contributor do not exist on both. To make the testing process even more difficult, different challenging image artifacts such as noises, occlusions, rotations, etc. were added.

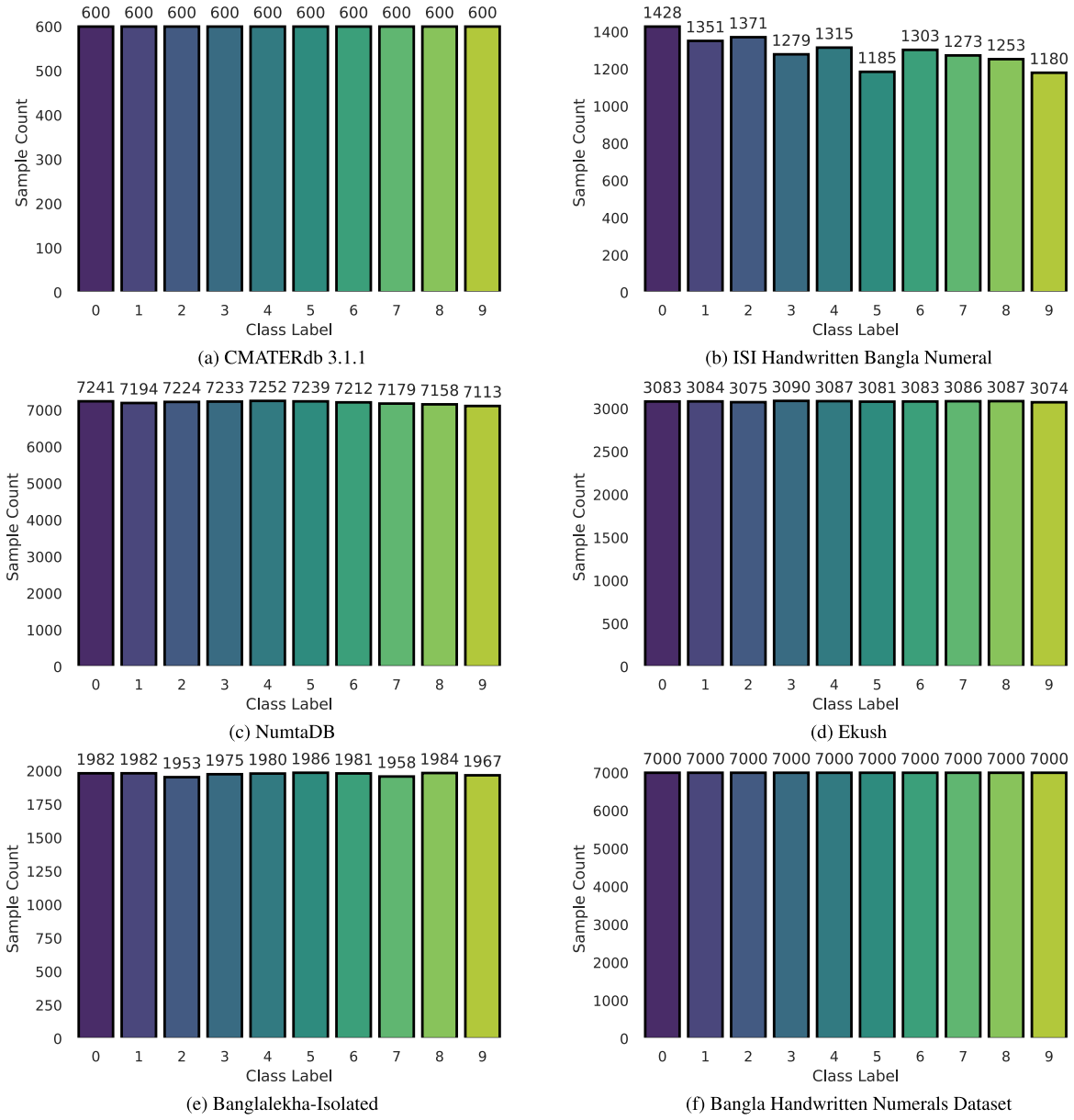


FIGURE 6. Class distribution of different Bengali handwritten digits datasets.

D. EKUSH

Ekush is one of the largest datasets known so far for Bengali handwritten characters, with 367,018 samples in 122 classes. Among those, more than 30 thousand samples are for digits with an equal male/female writer ratio. This dataset also comes with information about the age, gender, location, and educational status of the writers. This makes it useful for various applications like forensic investigation of detecting these modalities.

E. BANGLALEKHA-ISOLATED (BLI)

Although Banglalekha-Isolated is a dataset mainly for isolated Bengali characters, it also contains nearly twenty thousand samples of handwritten digits from 2,000 writers of diverse ages, gender, locations, and educational background. Moreover, the dataset also includes the age and gender labels for each sample, which makes it suitable for the investigation of age and gender influence on handwriting. It also contains an aesthetic quality index for each of the handwriting, marked by several experts. This unique feature provides new directions for investigation.

F. BANGLA HANDWRITTEN NUMERALS DATASET (BHaND)

The Bangla Handwritten Numerals Dataset (BHaND) has 70,000 samples, the same as MNIST [32], which has a 5:1:1 split for train, validation, and test. One basic difference with MNIST is the size of the images is 32×32 instead of 28×28. Authors claimed to have done this in order to make the size a multiple of two which may come in

TABLE 2. Summary of some of the databases used in Bengali Handwritten digit recognition.

Dataset	Source	Number of images	Writers	Male-Female ratio	Image format	Color format
CMATERdb [29], [30]	Real-life	6000	-	-	JPEG	RGB
ISI Handwritten Bangla Numeral [28]	Real-life	23392	-	-	TIFF	RGB
NumtaDB [25]	Curated	85000+	2742	Roughly 60-40	JPEG	RGB
Ekush [26]	Curated	30688	3086	50-50	JPEG	RGB
Banglalekha-Isolated [24]	Curated	19748	Roughly 2000	59.4-40.6	JPEG	RGB
BHaND [27]	Curated	70000	1750	56-44	JPEG	Gray-scale

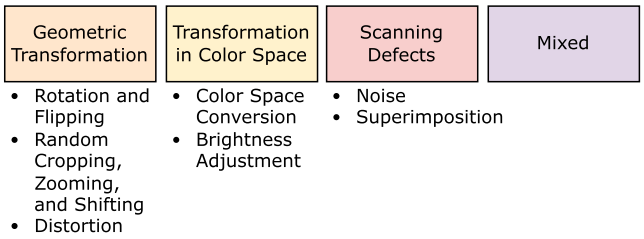


FIGURE 7. Taxonomy of the preprocessing methods used in Bengali handwritten digit recognition.

handy if down-sampling is needed while working with deep CNN-based architecture [27].

The metadata of the discussed datasets are listed in Table 2.

IV. PREPROCESSING

Data Preprocessing, also known as data cleaning, is one of the most common components in a BHDR pipeline with the lowest level of abstraction since both the input and output of this process are image intensity values. The purpose of image preprocessing is to remove distortions and/or enhance image features necessary before further processing without introducing undesirable artifacts. Various types of preprocessing methods seen in BHDR include image transformation, noise correction, geometric and morphological operations, image filtering, segmentation, etc. A taxonomy of the preprocessing techniques is shown in Figure 7.

A. GEOMETRIC TRANSFORMATION

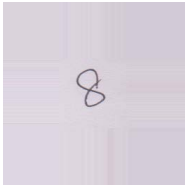
Geometric transformation deals with altering the coordinates of the pixels without changing the intensity values. A set of common geometric transformation techniques used in BHDR is illustrated in Table 3.

1) RESIZING

Since vision-based models tend to focus on extracting image textures, reducing the size of the input image does not hurt the performance of the model [33]. Rather, these sorts of resizing techniques are often recommended in mini-batch learning to reduce the time required to train the learning model [34]. Again, some deep neural network-based architectures might have strict requirements for input image dimensions in order to utilize the pretrained weights. For these reasons, in BHDR systems, the digit images are often resized to reduce the input dimensions.

Most of the existing literature on BHDR either use 28×28 [35], [36], [37], [38], [39], [40], [41], [42], [43] or

TABLE 3. Common geometric transformation techniques used in the existing literature. Original image taken from NumtaDB dataset [25].

Technique	Original Image	Preprocessed Image
Image Resizing		
Padding		
Cropping		
Slant Correction		

32×32 [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59] input dimensions. Other dimensions include 16×16 [60], 48×48 [61], 64×64 [62], and 128×128 [28], [63]. The popularity of smaller input dimensions indicates that digit images can be reduced in size without affecting the performance of the learning model. Various interpolation techniques are utilized in this regard to preserve the details [64]. Most of the existing literature uses Bilinear interpolation. Bicubic interpolation is also used in some of the works [65], [66].

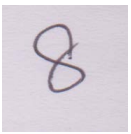
2) PADDING

Padding is used to meet the input dimension requirement for the learning model. Digit images from the same dataset can have different sizes. On the other hand, resizing smaller images to larger ones may introduce unwanted artifacts. To avoid these issues, padding can be performed to increase the size of the smaller images. For rotated images, padding can further ensure the preservation of the original aspect ratio of the digit [61]. To fill up the additional pixels padded to the digit images, constant values, neighboring values, mirrored values, etc. are used.

TABLE 4. Common color space transformation techniques used in the existing literature. Original image taken from NumtaDB dataset [25].

Technique	Original Image	Preprocessed Image
Color Inversion		
Grayscale Conversion		

TABLE 5. Common morphological operations used in the existing literature. Original image taken from NumtaDB dataset [25].

Technique	Original Image	Preprocessed Image
Thinning		
Thickening		

3) CROPPING

Cropping the input image can help the model to focus only on the digit portion, ignoring background noise. Again, based on the requirement of input dimensions of different models, the cropped images can be resized afterward. References [40], [59] achieved this by cropping the digit images based on the largest contour of the digits. On the other hand, some of the existing literature used manual cropping [37], [38], [67], [68], [69]. However, this is not feasible for large datasets and even from an application perspective, it should be avoided so that the performance of the model is not dependent on the manual cropping of digits.

4) SLANT CORRECTION

Slanted handwritten digits affect the accuracy of the learning model. For example, a slightly tilted six (৬) might be considered as eight (৮) as they have similar baseline skeletons with different orientations. KSC algorithm [70] is used to fix individual slanted digits [47], [48].

B. TRANSFORMATION IN COLOR SPACE

Color space transformation modifies the intensity values of the pixels in a digit image. A set of such techniques used in BHDR is illustrated in Table 4.

1) COLOR INVERSION

Since a majority of the dataset curators collect data from users/annotators on white paper using a black pen, scanned digit images contain white background and dark text in most cases. In 8-bit images, for example, the dark pixels have an intensity value of 0 and the white pixels have an intensity value of 255. That means, all the information regarding digits is stored as 0 values. It may be computationally expensive since models have to work with larger values corresponding to the background.

To avoid this, images are converted to binary form via thresholding techniques (such as Otsu's method [40], [54], [62], adaptive thresholding [47], [71]) and then inverted, so that the digits have an intensity value of 255 and the background has an intensity value of 0. This also helps get rid of random noises that might be present in the background of the digits. Although earlier machine learning-based approaches benefitted from grayscale images [72], having binary intensity does not affect the accuracy of convolutional neural networks [73]. This can be attributed to the invariance of the filters used in the convolutional layers that can easily identify edges regardless of the color space of the image [73]. Furthermore, it can reduce the computational complexity since most of the convolution is performed over pixels with an intensity value of zero [69].

2) GRAYSCALE CONVERSION

Since the color information of the handwritten digit image has no impact on the performance of the deep learning-based model, the input is often converted to grayscale. This reduces the number of channels in the input image, reducing the training time by decreasing the computational cost [41], [56], [74]. It also ensures the unanimity of the samples collected from multiple sources [40], [56]. To convert to grayscale images, usually the intensity values for each pixel are considered in the existing literature.

C. MORPHOLOGICAL OPERATIONS

Morphological operations were mostly popular with machine learning-based techniques, as these approaches may utilize pixel count to generate features. A set of common morphological operations is illustrated in Table 5.

1) THINNING

Based on the stroke, the thickness of the handwritten digit can vary. Digits with thicker strokes contain a larger number of pixels, requiring greater computational cost in feature extraction. To alleviate the issue, the thinning operation is performed to make the strokes have the same thickness [60], [66], [75]. Machine learning-based models that focus on the number of pixels to extract handcrafted features mostly use this technique.

2) THICKENING

Thickening is the opposite of the thinning operation. Due to the artifacts introduced during the scanning of the handwritten digits, some unintended gaps might be introduced between strokes. Considering the similarities between Bengali digits, this can hamper the performance of the model.

For example, if any gap is introduced in the middle portion of digit four (৪), it can be perceived as two separate zeros (০). Reference [39] used thickening operation to remove those gaps.

D. MISCELLANEOUS

Other common preprocessing techniques are illustrated in Table 6.

1) NOISE REMOVAL

Noise can distort images to the point that they become unrecognizable. This can result in the learning models fitting to the noise, reducing the overall performance. Adopting a proper noise removal technique can solve this problem. For example, to remove salt and pepper noise from the handwritten digits, median blur is used [40], [52], [54], [56]. To remove Gaussian noise, the Gaussian filter seems to be the popular choice [36], [43], [47], [48], [56], [59], [60], [75], [76]. Finally, in order to remove small artifacts, caused due to resizing the digit images, a combination of morphological opening and closing operations can be performed [59].

2) DEBLURRING

Having sharp curves and edges in the digit portion of the images can help improve the performance of the models that rely on shapes and texture information as features [54], [76]. On the contrary, blurry images can hamper recognition performance. To reduce the blurring effect, a copy of the original image is blurred using Gaussian blur and subtracted from the original image to create an unsharp mask. That mask is again added to the original image to reduce the blurring effect [54]. Another way is to use a sharpening kernel and perform 2D convolution on the image. For example, [76] and [77] used the Laplacian filter to extract edge information, which is then added to the original image to sharpen blurred samples.

3) REMOVAL OF COARSE DROPOUT

During the scanning of digits, artifacts like coarse dropout can be introduced in images. This can hamper the training process of the learning models by obscuring the digit to be recognized. To remove coarse dropout, contour approximation is used, which detects and converts the dropout pixels to white in order to match the background of the digit [40].

4) NORMALIZATION

Normalization is performed to reduce the effect of illumination differences in images. Considering the maximum and minimum intensity value of pixels in all images, the intensity values are scaled between 0 and 1. It also helps learning models converge faster. Min-max normalization is used to perform this operation [41], [78].

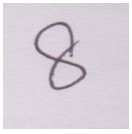
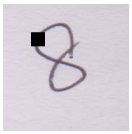
V. AUGMENTATION

Data augmentation techniques are applied to artificially increase the size of the dataset by creating a modified version of the samples. Considering the huge amount of data required to train deep learning models, various image augmentation techniques can be applied to handwritten digits to generate multiple copies of the same image with slight variation(s) [79]. These techniques also reduce the dependency of the model on preprocessing, since the model learns to recognize samples with imperfections during training. Not considering common augmentation techniques such as perspective transform, blurring, shearing, hue shifting, etc. can significantly affect the performance of the models [31].

Another benefit of augmentation is to fix the class imbalance issue. Class imbalance issue occurs when the distribution of samples among the known classes is skewed. This class imbalance can cause a few problems. First, the model fails to learn generalized features as it only sees a few instances of the classes with a lower number of samples [80]. Additionally, due to the small contribution of the small-sized classes in the overall accuracy, a model might achieve high accuracy even when it fails to learn about the small-sized classes [81].

Several augmentation techniques can be found in the existing literature that emulates real-life scenarios. A taxonomy of the augmentation techniques are shown in Figure 8.

TABLE 6. Other preprocessing techniques used in the existing literature. Original image taken from NumtaDB dataset [25].

Technique	Original Image	Preprocessed Image
Noise Removal		
Deblurring		
Coarse Dropout Removal		

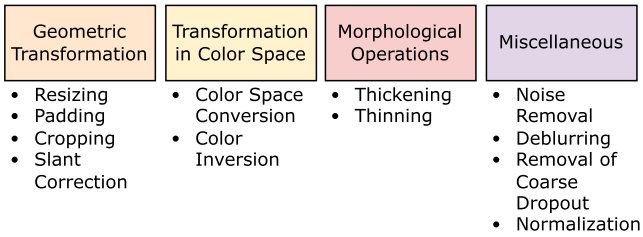


FIGURE 8. Taxonomy of the augmentation techniques used in Bengali handwritten digit recognition.

A. GEOMETRIC TRANSFORMATION

1) ROTATION AND FLIPPING

To help the models recognize slanted texts, rotation can be performed with respect to a certain pixel (Figure 9(b)). Hence, the existing literature on BHDR rotated the digit images by

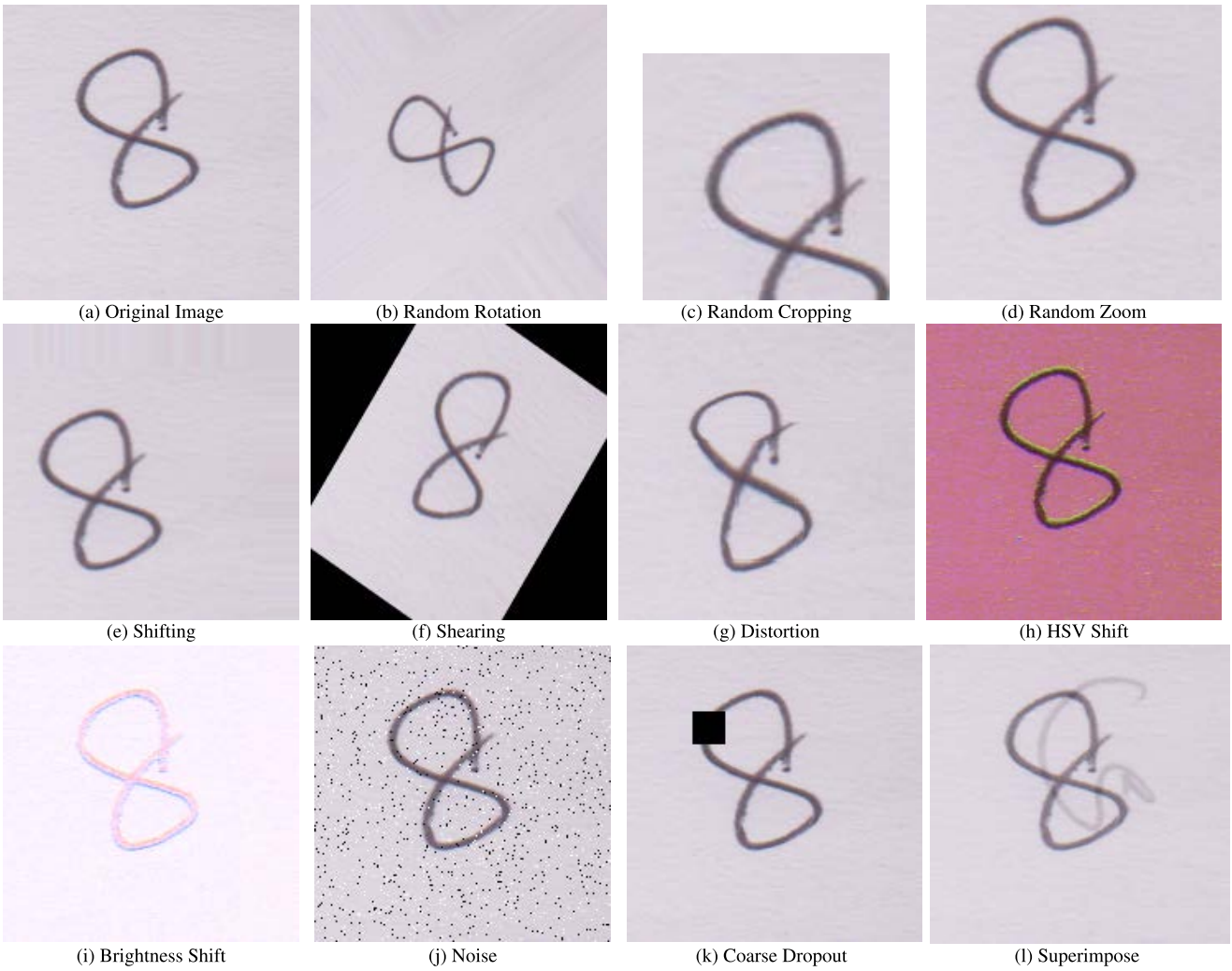


FIGURE 9. Common data augmentations techniques used in the existing literature. Original image taken from NumtaDB dataset [25].

a randomly chosen rotation angle within a range of 0 to 50 degrees on either direction [36], [39], [42], [61], [62], [63], [73], [77], [82]. One key consideration is that, after performing rotation, additional pixels are padded to preserve the original dimension of the image [73]. As an extension of rotation, flipping can be performed to flip the image horizontally or vertically along the centerline of the image.

2) RANDOM CROPPING, ZOOMING, AND SHIFTING

Cropping random portions of the image or zooming into it can enable the model to recognize digits by only seeing a portion of them (Figure 9(c)). Removing the discriminative portion of the digits (For example, the bottom part of '৯' and '৯') can help models learn to ignore non-important portions of the image. It also helps the model recognize occluded characters. Reference [61] applied random cropping to remove 0 to 20% of the digit images. A similar effect can be achieved using zooming and shifting. Zooming is performed via pixel replication while keeping the image dimension same (Figure 9(d)). In BHDR, 0 to 30% zooming augmentation can be seen [56], [62], [63], [73], [77], [78]. Shifting requires translating each pixel of a sample by a constant factor chosen randomly within a certain range (Figure 9(e)). The common values for the constant factor range from 0 to 0.30 [42], [56], [61], [62], [63], [73], [77], [78], [82]. In this process, the pixels that are shifted outside the image boundary are discarded and the pixels that become empty are filled with constant values or values of the nearest pixel.

3) DISTORTION

Shearing can be performed to replicate the effect of distorted digit images (Figure 9(f)). It moves each pixel in a specific direction. The magnitude of the movement is determined by the distance of the corresponding pixels from a certain baseline. The magnitude can also be controlled using a constant factor. Common values for constant factors are often in the range of 0 to 0.25 [61], [62], [77]. Shearing in this manner can help the model learn to recognize slanted texts.

Apart from that, application elastic transform [62] and grid distortions [39] are also used to distort the digit images (Figure 9(g)).

B. TRANSFORMATION IN COLOR SPACE

Different color channels found in different color spaces can convey various information regarding the sample. To exploit this information, color space transformation techniques are used to augment the training samples.

1) COLOR SPACE CONVERSION

Reference [61] used samples in both RGB and HSV color space to extract the latent information of each color channel (Figure 9(h)). On the other hand, reference [73] randomly shifted the value of the hue and saturation channel. Reference [71] inverted the images by converting the image to grayscale color space and then inverting the intensity values. Even though this technique is popular in deep learning-based models, it is expected that the models themselves should use a set of convolution layers to learn from different color channels, since these transformations can be represented using linear or nonlinear equations.

2) BRIGHTNESS ADJUSTMENT

Considering the heterogeneity of the lighting conditions during the scanning of digits, models can be trained using the same image with different brightness (Figure 9(i)). This can be achieved by shifting the value of the intensity channel in the HSI color space. Reference [63] adjusted the brightness of the samples within the range 0.75 to 1.25. Again, [62] used a washed-out version of the original samples to achieve a similar effect. This effect can also be achieved using histogram equalization which also enhances the image without any loss of details [83], [84].

C. SCANNING DEFECTS

To ensure that the model learns to recognize digits captured with lens blur, data can be augmented using blurred versions of the original samples. To achieve this effect, a Gaussian filter can be applied to the samples [61], [62], [73].

1) NOISE

Considering the adverse effect of noise in digit recognition, models can be trained to recognize noisy digits (Figure 9(j)). Consequently, during data augmentation, Salt and Pepper noise [62], [71], [73], Gaussian noise [61], random noise [71], etc. can be added. Additionally, to incorporate the effects of coarse dropout, random squared portions of the image can be selected and replaced with 0s (Figure 9(k)) [62], [71], [73].

2) SUPERIMPOSITION

While scanning a digit, if there is something written on the opposite side of the page, a horizontally flipped and blurred version of the text can appear superimposed on the original digit (Figure 9(l)). To ensure that the model learns to recognize the original digit in these scenarios, multiple training images can be superimposed that achieve the same effect. To do that, a weighted Gaussian blur of one image is added with another image [62], [73], [77]. This process is known as superimposition.

D. MIXED

To add further variety to the augmented samples, multiple augmentations can be combined to generate additional samples. For example, [73] randomly picked 1-3 augmentations and applied them simultaneously on their dataset to create 5 to 7 new images from each image of the training set.

VI. MACHINE LEARNING-BASED APPROACHES

Machine Learning-based approaches mostly focus on feature extraction and classification. In such works, handcrafted features are extracted from the digit images and then fed to a classifier to generate the prediction. Similar to other tasks involving the classification of images, the development of a conventional machine learning-based system for recognizing handwritten Bengali digits typically entails the completion of three phases. The first thing that we do is extract crucial information from the image that we are given, and the information that is extracted is referred to as features. In a later step, the feature space is shrunk down to a smaller dimensional size. This can be beneficial for lowering the required level of computing complexity or getting rid of features that are superfluous or biased and have the potential to confuse a model. In the last step, the characteristics that were chosen are sent into a classifier so that it may be trained. Previous research has demonstrated that it is possible to successfully use a variety of approaches to each of these processes. Figure 10 illustrates a schematic representation of the taxonomy of the machine learning-based approach used for this particular task.

A. FEATURE EXTRACTION

One of the most challenging tasks of handwritten digit classification in earlier machine learning-based approaches was to find a suitable feature extraction technique that can uniquely represent the individual digits. For Bengali handwritten digit recognition, these descriptors can be divided into four broad categories:

1) Geometric Features
2) Statistical Features
3) Pixel-based Features
4) Miscellaneous

1) GEOMETRIC FEATURE EXTRACTION METHODS

a: LOCAL BINARY PATTERN (LBP)

Local Binary Pattern is an efficient texture operator, with an easy and robust calculation method [85]. It iterates through each pixel of the image and generates a label by comparing each neighboring pixel with the pixel itself. Each label is either '0' or '1' based on whether it is bigger or smaller than the center pixel. All the labels of the neighboring pixels are concatenated to form a binary number. Later, it creates a histogram using the values of those binary numbers. An illustration of calculating LBP is shown in Figure 11.

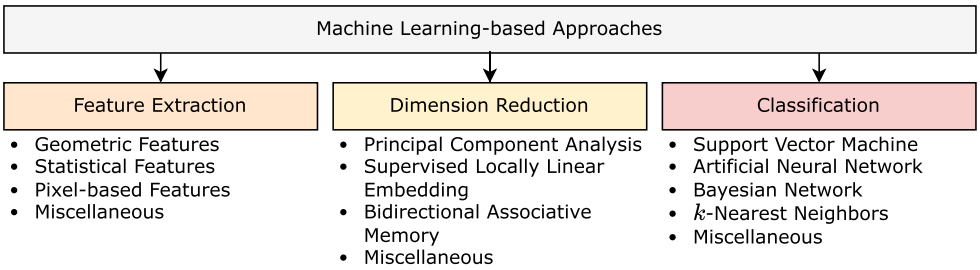


FIGURE 10. Taxonomy of the machine learning-based techniques used in Bengali handwritten digit recognition.

Reference [48] used the basic, uniform, and simplified variations of LBP. Each of them produced an accuracy of around 96.5% while simplified LBP slightly underperformed. On the other hand, [43] used LBP with 10 neighbors with a radius of 3 and reported an accuracy of 95.3% on the CMATERdb dataset. However, the performance of LBP was comparatively worse in datasets containing challenging samples such as NumtaDB and Ekush. This can be attributed to the fact that LBP is prone to variations in illumination and random noises [86], which are present in digit samples of those datasets.

b: HISTOGRAM OF ORIENTED GRADIENT (HOG)

The Histogram of Oriented Gradient (HOG) feature descriptor exploits the count of the occurrence of the gradient orientation in localized regions of an image [87]. It determines whether a pixel is on an edge or not, along with the gradient and the direction of that pixel. It calculates the gradient for every pixel along both axes of the image. There are different filters that can be used to calculate the gradient such as Sobel [88], Kirsch [89], Roberts [90], etc. Then the magnitude and the orientation are calculated using Equation 1 and 2, respectively.

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2} \tag{1}$$

$$\text{Orientation} = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2}$$

where G_x and G_y are the difference between the neighboring pixel values on the x-axis and the y-axis, respectively.

After that, a histogram is generated, storing the magnitude into several bins based on the value of the orientation. The size of the filter and the number of bins in the histogram depends on the implementation. This also controls the number of features generated. An illustration of calculating HOG is shown in Figure 12.

Reference [53] used 8×8 cell to create a feature vector along with the color histogram and reported 98.05% accuracy on the CMATERdb dataset with SVM as the classifier. In another work, the authors used 2×2 cell to create a feature vector of length 6084 [43]. On the CMATERdb dataset, they reported 98.08% accuracy. However, the performance on other datasets was relatively low: 93.32% on NumtaDB and 95.68% on Ekush. Reference [72] used Sobel and Roberts filters to create the histogram. They presented a comparative analysis of the performances of the ISI-HBN dataset from different viewpoints. The authors claimed that the Sobel operator performs slightly better than Roberts with a maximum accuracy of 99.40%. Reference [52] also used HOG features with normalization.

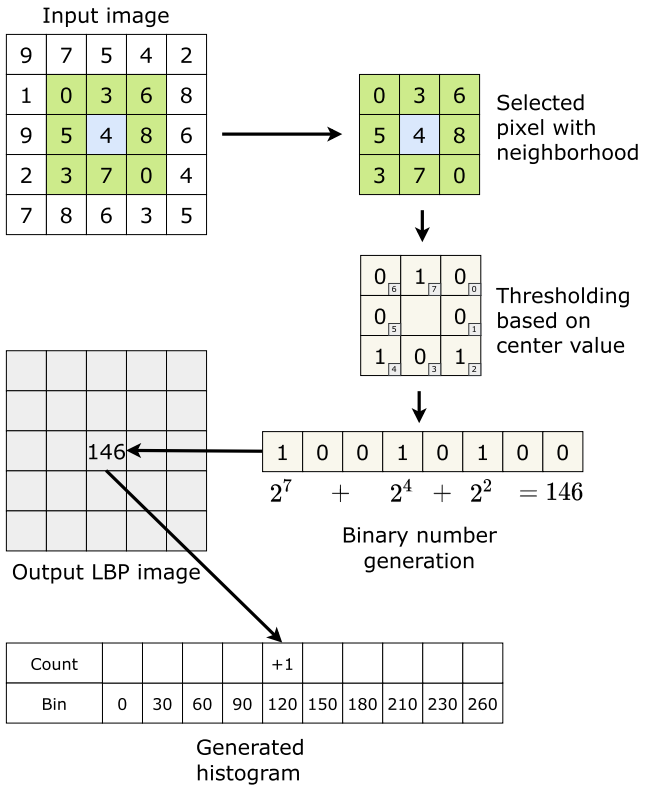


FIGURE 11. Local Binary Pattern (LBP) generation from a pixel. This process is repeated for all the pixels in the input image.

c: DIRECTIONAL PATTERN

A number of directional patterns, such as: Local Directional Pattern (LDP) [91], Gradient Directional Pattern (GDP) [92], etc. have been used as feature descriptor over the years.

LDP is similar to LBP, except that it creates the binary code using 8 different Kirsch filters instead of direct thresholding. As it considers edge responses instead of the pixel density of the neighboring pixels, it can provide more consistency in the presence of noise compared to LBP [93]. On the other hand, GDP applies operators like the Sobel masks to the image to find the gradient along both the X and Y-axis. The gradient orientation uses the same equation as the orientation

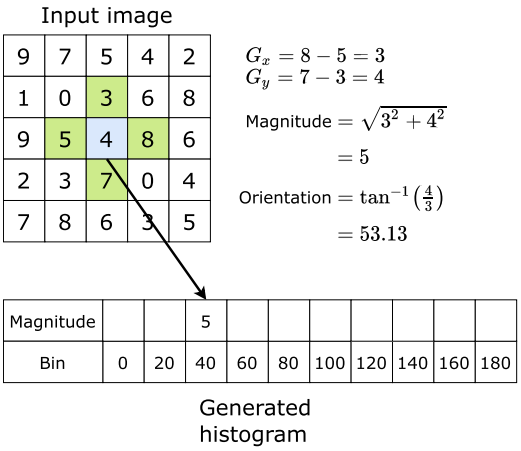


FIGURE 12. Histogram of Oriented Gradients (HOG)-based histogram generation from a pixel. This process is repeated for all the pixels in the image.

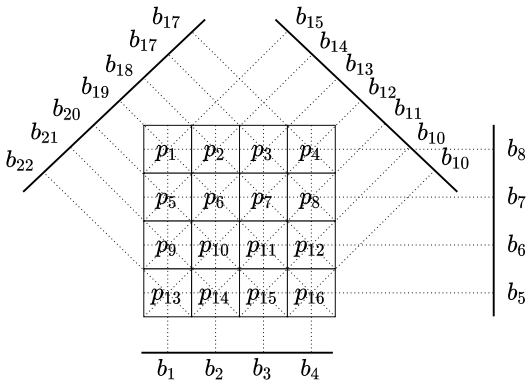


FIGURE 13. Projection histogram generated from a 4 × 4 image. Adapted from [98].

of HOG features Equation 2. After that, the orientation of a pixel is compared with the neighboring pixels, and based on a threshold, it is coded as either '0' or '1', creating a binary number. Finally, a histogram is obtained from the values of the binary number similar to LDP. The benefit of using GDP is that the produced codes are consistent for identical and near-identical regions of digit images, unlike the ones produced by LDP [92].

Reference [51] investigated the performance of both GDP and LDP. Authors claimed that an ensemble of both features yields maximum accuracy of 95.62% on the CMATERdb dataset.

d: WAVELET FILTER AND CHAIN CODE

Wavelet filters are a well-established tool for multi-resolution analysis of handwritten digit images [94]. Wavelet filters decompose an image into a hierarchy of several levels of resolution, making a coarse-to-fine recognition scheme possible. There exist many sets of wavelets used in image analysis. Usually, the input image is split into two components: a smooth component and a detail component by using low-pass and a high-pass filter, respectively. Both components are then down-sampled by a factor of 2. The low-pass component is then split further into low-pass and high-pass components as above for the second time and they are again down-sampled by a factor of 2. This process of splitting and down-sampling, also known as the 'pyramidal algorithm', is continued as far as required.

Reference [28] proposed a system using a wavelet filter and chain-code-based feature extraction technique. The authors used Daubechies-4 wavelets [95], which works better in both the time domain and frequency domain. After that, chain code histogram features are obtained corresponding to each of the detailed image components. The bounding box is divided into equal sizes of blocks and for each block, a local histogram of chain code is computed. The size of the feature vector is reduced to 64 by down-sampling using a Gaussian filter. Later, an MLP classifier is used to classify. A wavelet filter, consisting of linear operations, is computationally fast and suitable for real-life applications. The authors achieved 98.2% test accuracy on the ISI-HBN dataset.

e: GABOR FEATURE

The Gabor filter can approximate the characteristics of a certain cell in an image in the visual cortex of a mammal [96]. It is a variant of band-pass filters that allows passing a band of frequency in a certain orientation. It analyzes whether there is any specific frequency content in the image in specific directions within a localized region around the point or region of analysis. Reference [43] used a 2D Gabor filter with default parameters for digit recognition and compared the performance of Gabor features with HOG and LBP using a variety of classifiers on NumtaDB, CMATERdb, and Ekush datasets. They concluded that Gabor features perform poorly compared to other features in all the datasets used.

f: PROJECTION HISTOGRAM

To generate a projection histogram, the image is iterated row-wise and the number of pixels that are part of the digit is counted for each row. The same process is repeated for each column. Then, considering the rows and columns as bins, a histogram is created. This feature is also used in digit segmentation. However, [97] concluded that this feature performs poorly in digit recognition with an accuracy of around 82% on a self-curated dataset. This was further improved by considering a celled version of projection, where the image is divided into multiple cells and the length of the projection is considered. Celled projection can improve the accuracy up to 92.60% on the same dataset [97]. In another work, [98] exploited Mojette transformation that converts the 2D image into a set of discrete 1D projections. It generates a set of vectors, where each of the elements is calculated using the sum of the value of pixels that lie on the projection line.

g: SHADOW-BASED FEATURES

Shadow features are calculated using the length of the shadow that is supposed to project on a surface. As the shape of Bengali digits varies a lot, the shadow length also varies from different lighting positions. The images are divided into

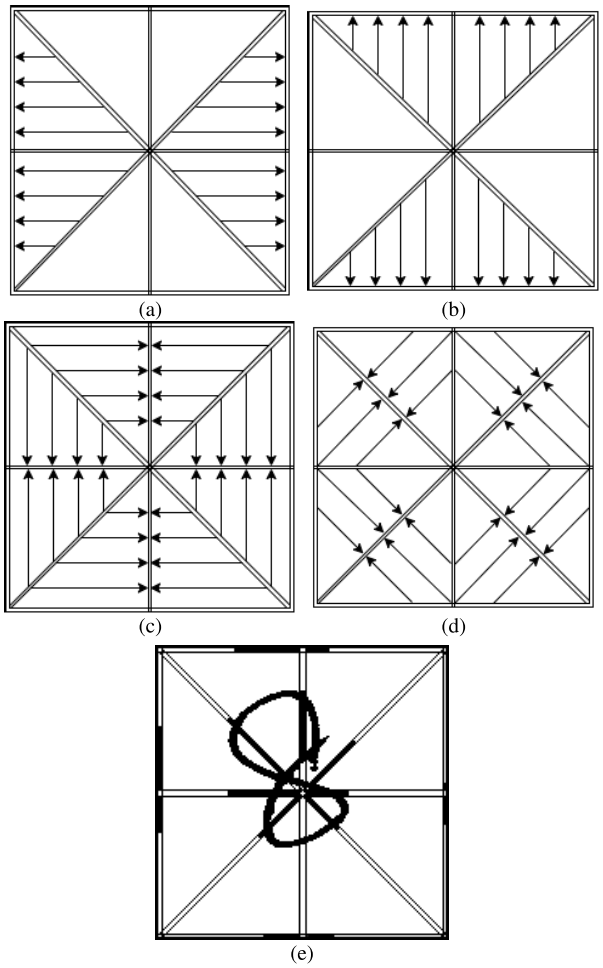


FIGURE 14. Extraction of Shadow-based Features. The arrows show the direction of light for generating the shadow. Digit image taken from NumtaDB dataset [25]. (a) Projection in a horizontally outward direction. (b) Projection in a vertically outward direction. (c) Projection in horizontally and vertically inward direction. (d) Projection in a diagonal inward direction. (e) Estimated projection for Bengali digit 'Four'.

different zones to extract multiple shadow features. References [30], [44], [45] proposed zone-based shadow features dividing the image into 8 triangular shaped zones extracting 24 shadow lengths (Figure 14).

In a more recent work, [99] proposed a single light source-based shadow feature named Point Light Source-based Shadow (PLSS). The shadow changes if the position of the light source is changed (Figure 15). They obtained 20 shadow lengths from four different source positions for four local regions and one global image. All the methods combined shadow-based features with other features, i.e. centroids, pixel density, and crossing to construct the feature vector.

h: CONTOUR ANGULAR TECHNIQUE

Reference [66] proposed a system named Contour Angular Technique (CAT) to capture the aspects of the curvature in the numeral image and angular co-occurrence. CAT is a fast implementation of quantized angle co-occurrence computation similar to the Hinge feature [100]. In this method, the original image is divided into 4 × 4 non-overlapping blocks. Then 8-directional code for identifying the contour of the neighboring pixels of a starting point is calculated for each of the blocks. Along with this, it also calculates an angular co-occurrence histogram with 64 elements that approximates the angular co-occurrence probability along the contours. By combining both outputs, a feature vector of size 192 is extracted which is then fed to SVM classifiers. Authors reported 96% accuracy on a self-made dataset.

2) STATISTICAL FEATURE EXTRACTION METHODS

a: MOMENTS

Moments calculate the center of gravity of the image considering the pixel distribution to capture global shape information [101]. They can be used to describe quantities at a distance from an axis or a point. They were popular due to their invariance to rotation, translation, and scaling. Reference [97] constructed a feature vector with fifteen translation-invariant central moments and achieved a maximum accuracy of 86.7% with a Feed Forward Back Propagation Neural Network. In another work, [102] extracted a feature vector containing 130 features that combines six types of moments: Complex moment [103], Zernike moment [104], Legendre moment [105], Geometric moment [106], Affine moment invariant [107], and Moment invariant [108]. They reported 99.5% accuracy on the CMATERdb dataset with MLP as the classifier.

b: CENTROIDS

Centroid is the middle point of an object. In image analysis, centroids are obtained as the weighted average of the pixel coordinates belonging to the digit in a region of interest. Global centroids are calculated considering all the pixels of the image, whereas local centroids are calculated for different zones. Figure 16 shows the centroids for each octant along with the global centroid.

Reference [109] used the global centroid to divide the image vertically into two regions. A scalene triangle was created using the three centroids (one global centroid and two centroids of two regions) as the corner points. Then 9 different features were obtained using triangular geometry.

c: PIXEL DENSITY

Pixel density is another statistical feature that is often used in different image classification tasks. It counts the number of black pixels within a region of interest; be it the whole image or a zone division within the image. In some methods, it is normalized by the size of the zone to obtain the density.

Reference [99] divided the image into eight octants and created a histogram called Histogram of Oriented Pixel Positions (HOPP), counting the number of black pixels within the region. The HOPP feature is combined with the PLSS feature to achieve 98.5% accuracy on CMATERdb 3.1.1 dataset. In other works, the whole image was divided into rectangular zones, and zone-wise local density was taken as features [47], [65], [97]. Reference [97] used 4 × 4 zoning and achieved 90.30% accuracy on self-curated dataset.

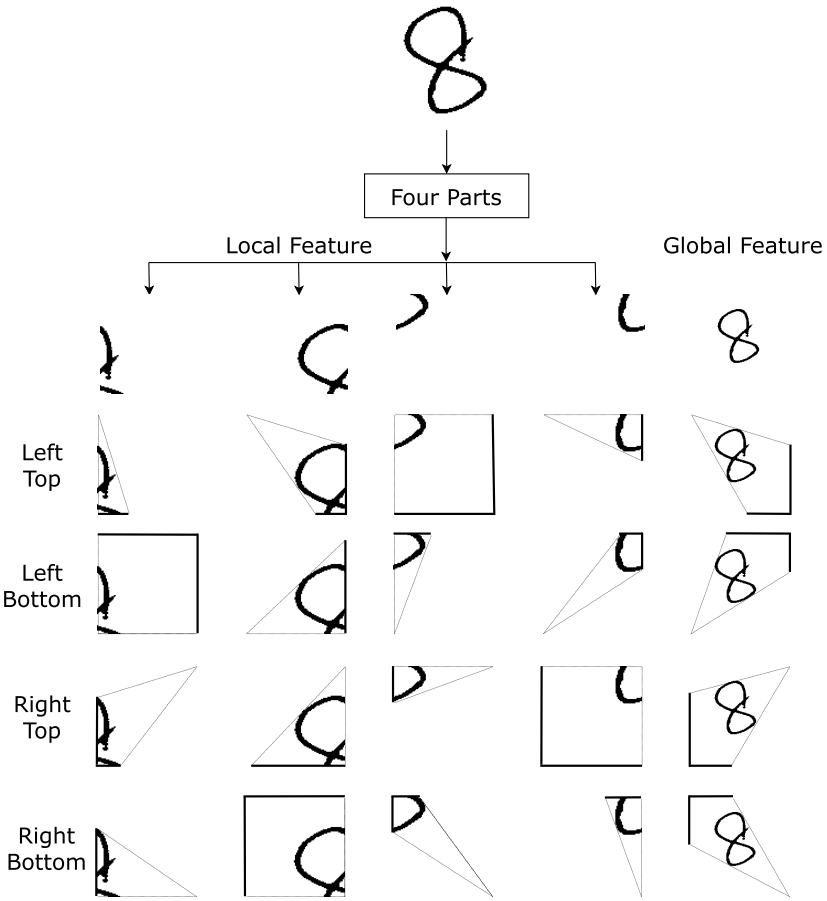


FIGURE 15. Light source-based shadow feature extraction. Digit image taken from NumtaDB dataset [25].

Reference [65] combined pixel density with other statistical features and reported 95.7% accuracy on their own dataset. Reference [47] reported 94.0% accuracy on ISI-HBN dataset using 8×8 zoning.

d: LONGEST RUN

The longest run of a numeral image is the maximum number of consecutive black pixels along with any directions. The longest run is usually calculated considering four directions: horizontal, vertical, diagonal, and anti-diagonal. Reference [30] used the length of the longest run in four directions. As illustrated in Figure 18, the longest run along the row is calculated considering the length of the longest bar that fits the maximum consecutive black pixel is computed for each row. Then the sum of the lengths for all the rows is calculated to get the horizontal longest run. The same method is repeated for the vertical and the two diagonals. Each of the values is then normalized based on the length of the direction. References [44], [45], [110] used a normalized longest run for 9 overlapping windows.

e: CROSSING

Crossing is obtained by counting the number of transitions between the background to the foreground or vice versa in any direction in the image. In simple words, it counts the number of changes between black and white pixels along each row or column. It can be obtained along the diagonals as well. Unlike other features, it is invariant to the width of the strokes and does not need thinning. Reference [97] computed crossings for every column and row to construct the feature vector of the image and reported 86.40% accuracy on a self-curated dataset of 12000 images. Reference [111] divided the image into 16 zones and a mask of 4 regions surrounding each other was applied to each of the neighboring zones. Crossing was counted for each of the regions along 4 different directions: vertical, horizontal, and the two diagonals. Authors claimed that a feature set of length 144 yielded 97.85% accuracy on a self-curated dataset of 10,000 images.

f: THE HOTSPOT TECHNIQUE

Reference [66] proposed a feature extraction method to represent the numeral image that considers the distance between evenly spaced hotspots in the image and the nearest black pixels in any direction. The distance is set to a maximum threshold value if the black pixel is not found in any direction. The authors used 25 hotspots and 4 directions to create a feature vector of length 100. As this is a global feature and lacks the ability to uniquely represent the numerals, this

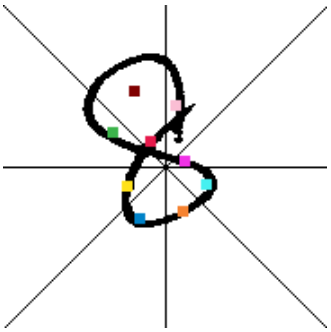
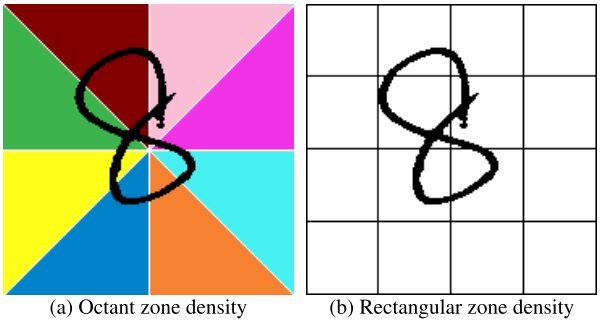


FIGURE 16. Centroid features extracted from each octant. The red dot denotes the global centroid and the rest are local. Original image taken from NumtaDB dataset [25]. (Best viewed in color).



(a) Octant zone density (b) Rectangular zone density

FIGURE 17. Extraction of zone-based density features. Original image taken from NumtaDB dataset [25].

feature accumulated maximum accuracy of 92.7% on a self-curated dataset.

g: LOOP FEATURES

The area of the completely enclosed portion of the digit pattern is considered a loop. In Bengali digits, the number and the size of the loop vary from digit to digit (Figure 19). This feature was utilized by [30]. In other works, the relative positions of the loops were considered in addition to the area and the number [65], [112].

h: WATER OVERFLOW FROM RESERVOIR

Water overflow from reservoir takes advantage of the rounded nature of Bengali digits (Figure 20). It considers that the closed portion of the digits was filled with water, and extracts features based on that. Its use was first seen in [112], who used it as an extension of loop features. If there was no loop to be found, this feature was used. In another work, [113] extracted overflow features such as the direction of the overflow, the height of the water during overflow; reservoir features such as position and shape of the reservoir, etc. These features were particularly useful in the segmentation of touching numerals.

3) PIXEL-BASED FEATURE EXTRACTION METHODS

Instead of calculating the features through statistical or geometric methods, the pixel-based method directly uses the intensity of the pixel values from either the original image or sub-images. The benefit of using this method is the avoidance of error due to the miscalculation of features, which can occur in complex images. As a result, the performance of

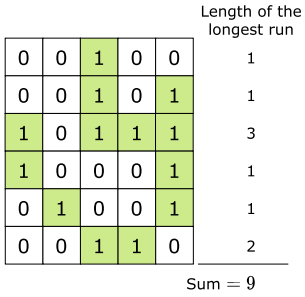


FIGURE 18. Longest run feature extraction in the horizontal direction of a binary image of the digit six (৬).

these methods can be better than that of geometric/statistical feature-based methods. These pixel values are often fed into artificial neural networks [49]. Reference [75] used a combination of Back-Propagation Neural Network (BPNN) and Bidirectional Associative Memory (BAM) to accomplish this task. Reference [46] proposed the use of Hierarchical Bayesian Network (HBN). It took 32×32 images that are divided into patches of 4×4 pixels. The input layer of the network consisted of 16 nodes, each corresponding to one pixel of the patch. The hidden layer had 4 children nodes for each of the nodes in the input layer. These nodes are fed into a single node in the output layer. Another work on pixel-based feature extraction methods depended on Supervised Locally Linear Embedding (SLLE) to reduce the dimension of the feature vector before feeding it to an SVM classifier [114].

4) MISCELLANEOUS

a: IMAGE DISTORTION MODEL DISTANCE

Image Distortion Model Distance (IDMD), proposed in [115], is a well-known feature used for image recognition or matching. Despite having high accuracy in recognition, the computation of this distance is quite expensive while handling a large dataset. Reference [116] proposed a method of selecting a limited number of images to keep the computation time within reach while maintaining high accuracy. They proposed an ensemble of multi-classifier approaches to achieve an accuracy of 98.55% on the ISI-HBN dataset.

b: FOURIER TRANSFORM

Fourier transform is a popular method in signal processing. It provides information about the frequencies present in a signal; image in our case. Low frequency carries shape and structure information, whereas high frequency provides finer details. In digit recognition, shape information is much more important than finer details. Reference [97] proposed the use of 64 low-frequencies as a feature vector since it reduces the feature dimension while also keeping the shape information intact. As subtle changes in the time domain do not yield a significant change in the frequency domain, the performance of this feature is not worth mentioning.

B. DIMENSION REDUCTION

Feature dimension reduction is applied to reduce the relatively redundant features resulting in an increase in the speed

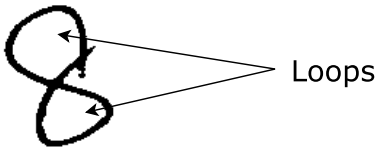


FIGURE 19. Bengali digit 4 containing 2 loops.

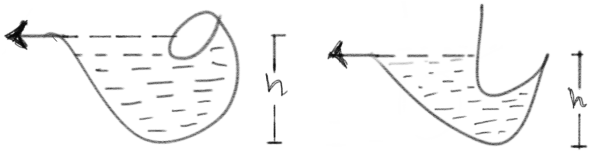


FIGURE 20. Water overflow features for digit 3 and 6.

of the recognition process. Principal Component Analysis (PCA) is used to extract uncorrelated components by linearly transforming a high-dimensional input vector into a low-dimensional one [120]. This technique was applied to reduce feature dimension and tested on eight different classifiers, resulting in the reduction of processing speed and improvement in performance [119]. In another work [60], the features, reduced using PCA, were fed to kNN ($k = 1$) and SVM classifiers where SVM provided better performance. PCA was also found to be useful in reducing different histogram-based features [72].

Among other feature reduction techniques, Supervised Locally Linear Embedding (SLLE) [121] was used to project the input features in a lower-dimensional feature space, which reduced the training time [114]. However, this resulted in a slight reduction in accuracy compared to the original feature space. In another work, Bidirectional Associative Memory (BAM) [122] was used to reduce input features for Artificial Neural Network (ANN), which may help reduce overfitting problems by decreasing the number of hidden layers and their neurons [75]. Reference [123] proposed a feature selection method named Memory-Based Histogram-Oriented Multi-objective Genetic Algorithm (M-HMOGA). This method applied Genetic Algorithm (GA) for feature selection that ensures adequate exploration of the entire search space by the use of histograms and storing the best set of candidate solutions throughout the generations. Authors claimed their method improved the performance of different classifiers and used only half of the feature set.

One interesting prospect that was seen in the existing literature is that the authors often selected the feature extraction technique in such a way that the number of features will be limited. For example, after experimenting with different cell sizes, [43] selected 4×4 HOG cell size that maintains a significantly high accuracy using less number of features. Similarly, other literature compared different feature extraction techniques and selected zone density of dimension 8×8 (as opposed to zone dimensions 32×32, 16×16 and 4×4) [47], uniform LBP (as opposed to basic and simplified LBP) [48], Histogram of Oriented Pixel Positions (as opposed to Point-Light Source-based Shadow and their combined approach) [99], celled projection (as opposed to crossings, Fourier transforms, moments, projection histograms, zoning) [97], and contour angle technique (as opposed to gray pixel-based method, hotspot technique, black and white down scaled method) [66] to keep the number of features minimum while also achieving a respectable accuracy.

C. CLASSIFICATION

Majority of the ML-based works achieved the best results using Support Vector Machine [30], [43], [51], [52], [53], [60], [66], [72], [110], [114], [114] and Artificial Neural Networks [28], [44], [45], [49], [65], [75]. Apart from these two, other works on Bengali handwritten digit recognition employed Probabilistic Neural Networks [97], Hierarchical Bayesian Network [46], Bayesian Discriminant [119], k-Nearest Neighbor [48], Sparse Representation Classifier [47], Random Forests [99], Decision Tree (DT) [112], [113], Modified Quadratic Discriminant Function (MQDF) [118], etc.

1) SUPPORT VECTOR MACHINE (SVM)

Before the popularization of deep learning techniques, SVM [124] was one of the most robust techniques for handwritten digit recognition. SVM divides a dataset into two classes based on hyperplanes, maximizing the distance between those classes. For multiclass problems, such as handwritten digit recognition, One Versus One (OVO) or One Versus All (OVA) SVM is used [125], [126]. Considering the digit classes may not be linearly separable, a nonlinear kernel can be used to map the data into a new, higher-dimensional space in which they can be linearly separated, keeping the margin as wide as possible. There are different variants of kernel functions that can be used to build the hyperplane [30]. Before applying SVM, it is important to normalize the feature vector to avoid the attributes having a very large numeric range so that all features get equal importance [66]. The most commonly used kernels in Bengali digit classification are linear kernel [30], [114], Gaussian kernel [30], Polynomial kernel [30], [51], [60], and RBF kernel [43], [66], [72], [110]. Reference [52] showed that linear kernels work better for a large number of features. On the other hand, if the feature set is smaller than the number of samples, then RBF and polynomial kernels were preferred. RBF kernels can outperform polynomial kernels, however, it requires higher computational resources [72].

In BHDR, SVM has performed the best using HOG features [43], [52], [53] compared to other features such as LBP and Gabor. This is because HOG features preserve local pixel interactions which cannot be captured using LBP due to its limited capabilities. Combining global features with a genetic algorithm, simulated annealing, and hill-climbing approaches to extract local features have also been seen to provide good accuracy [30]. Reference [72] showed that SVM classifiers can provide similar accuracy as other classifiers such as polynomial network classifier, class-specific feature polynomial classifier, and discriminative learning quadratic discriminant function. SVM has also been shown to provide comparable accuracy using directional edge features [60], gradient and

TABLE 7. Performances of the traditional machine learning-based approaches in self-curated datasets.

Reference	Year	Feature Extraction	Classifiers	Number of Images	Accuracy (%)
[112]	2000	Water resorvoir	Decision tree	10000	91.98
[113]	2006	Water overflow from resorvoir, topological and structural features of the numerals	Binary tree classification	12000	92.80
[65]	2006	Pixel and shape features: curves, holes, terminating and intersecting points	Multilayer Perceptron	10500	97.20
[117]	2006	Fuzzy frequency using histogram analysis	Rule-based	N/A	85.00
[60]	2007	3×3 Kirsch Mask for edge detection	Support Vector Machine	16000	95.05
[46]	2008	Not applicable	Hierarchical Bayesian Network	2000	87.50
[118]	2008	Histogram of direction chain code of the contour points and gradient based features	Modified quadratic discriminant function	2692	99.08
[28]	2009	Orthogonal wavelet and chain code histogram	Multilayer perceptron	23392	98.20
[97]	2011	Celled projections	Probabilistic Neural Network	12000	94.12
[119]	2012	Polynomial and Gaussian kernels	Kernel and Bayesian Discriminant-based Classifier	45000	96.91
[66]	2013	Contour Angular Technique	Support Vector Machine	10920	96.80
[75]	2014	Pixel values	Backpropagation Neural Network	1400	91.10
[49]	2016	Pixel values	Multilayer Perceptron	70000	96.05

longest run features [110], global and local directional pattern [51], curvature features [66], and even pixel-values of the image samples [114]. Reference [66] was the only study to consider the ensembling of four different SVM classifiers using the Unweighted Majority Voting (UMV) method to combine their results. To break the tie between the classes they used a random method. Their accuracy increased from 96.4% to 96.8% when UMV based ensemble technique was used.

2) ARTIFICIAL NEURAL NETWORK (ANN)

An Artificial Neural Network (ANN) works similar to the human brain, consisting of many interconnected neurons that are used to transfer information [127], [128]. The connections carry weights that are used to approximate a function estimating the output class from the input features. These weights are updated via adaptive learning [129]. A commonly used variant of ANN is the multilayer perceptron (MLP), which consists of input, hidden, and output layers. The number of nodes in the input layer and the output layer corresponds to the number of features and the number of classes, respectively. Hence, in digit recognition, the number of nodes in the output layer is 10. The number of nodes in the hidden layer can be identified using grid search to find the optimal configuration [130]. However, increasing the number of nodes generally results in an increase in computational complexity during training and inference.

One of the earliest works on ANN experimented with different choices of hyperparameters such as activation function, number of hidden layers, number of neurons, and batch sizes [49]. The authors found that a single hidden layer containing 700 neurons with a ReLU activation function can achieve the highest accuracy (96.05%) on a self-curated dataset of 70,000 images. Another work also experimented with the number of neurons in a single hidden layer [45]. They achieved the highest accuracy (96.65%) on a self-curated dataset containing 6,000 images using 65 neurons. This reduction in neurons is due to the lower number of samples in the training set. A similar pattern can be seen in [75], which required 52 neurons in the hidden layer in the proposed ANN architecture, trained using only 800 digit samples.

Reference [44] utilized Dempster-Shafer's (DS) theory [131] to propose an ensemble technique containing two MLPs trained using different sets of features. The ensembled architecture was able to increase the accuracy of digit recognition by at least 1.4% compared to the individual MLPs (93.7% and 92.62%). On the other hand, [65] proposed a multi-layer NN trained using both printed and handwritten Bengali numerals, which achieved 95.7% accuracy in recognizing handwritten digits and 99.2% accuracy in recognizing printed digits. Reference [28] employed three cascaded MLP classifiers provided with images of different resolution levels for digit recognition. If the confidence of the output label is below a certain threshold, another MLP is used to combine the outputs of the three MLPs to provide a final prediction.

3) BAYESIAN NETWORK

Bayesian Network (BN) simplifies the Bayes theorem given that both the probability distributions for the random variables and the interactions among them are subjectively described and the model can be regarded to represent "belief" about a complex domain [132]. Bayesian probability is the study of subjective probabilities or confidence in an outcome, as opposed to the statistical inference method. One drawback of BNs is that finding the optimal solution is NP-Hard, requiring the implementation of approximation algorithms to achieve a polynomial time solution [133]. Reference [97] implemented a Bayesian classification-based Probabilistic Neural Network (PNN) using a rapid feature extraction method, Celled Projection to compute the projections of each section as features from an image. This PNN architecture had two layers: the radial basis layer and the competitive layer. Some training is required for the estimation of the probability density function associated with each class. The performance of PNN depended on the spread factor which was chosen not more than 3 by experiment. They

TABLE 8. Performances of the traditional machine learning-based approaches in the commonly used datasets.

Reference	Year	Feature Extraction	Classifiers	Dataset	Number of Images	Accuracy (%)
[52]	2017	Histogram of Oriented Gradient, Histogram generation, and block normalization	Support Vector Machine	BLI	6000	97.08
[114]	2009	Pixel features based on supervised locally linear embeddings	Support Vector Machine	ISI-HBN	70000	97.06
[72]	2009	8-directional gradient histogram extracted using Kirsch and Roberts mask	Discriminative Learning Quadratic Discriminant Function	ISI-HBN	23392	99.40
[110]	2012	Quad tree-based directional gradient and longest run	Support Vector Machine	ISI-HBN	4200	93.34
[44]	2005	Octant-based (shadow and centroid) and longest run	Multilayer Perceptron	ISI-HBN	6000	95.10
[45]	2005	Shadow, centroid and longest run features	Multilayer Perceptron	CMATERdb 3.1.1	6000	96.67
[30]	2012	Genetic Algorithm for Local Region Sampling	Support Vector Machine	CMATERdb 3.1.1	6000	97.00
[47]	2014	Zone density	Sparse Representation Classifier	CMATERdb 3.1.1	6000	94.00
[48]	2015	Local Binary Pattern with Basic, Uniform and Simplified Zoning	k-Nearest Neighbor	CMATERdb 3.1.1	6000	96.70
[102]	2016	Moments	Multilayer Perceptron	CMATERdb 3.1.1	6000	99.50
[51]	2017	Local and Global Directional Pattern	Ensemble of k-Nearest Neighbor and Support Vector Machine	CMATERdb 3.1.1	6000	95.62
[53]	2018	Histogram of Oriented Gradients	Support Vector Machine	CMATERdb 3.1.1	6000	98.05
[99]	2021	Point Light Source-based shadow and Histogram of Oriented Pixel Positions	Random Forest	CMATERdb 3.1.1	6000	98.50
[43]	2021	Histogram of Oriented Gradient	Support Vector Machine	CMATERdb 3.1.1	6000	98.08
				NumtaDB	85596	93.32
				Ekush	30688	95.68
				BDRW	1393	89.68

have also explored Feed-forward Back Propagation Neural Networks (FBPN) and k-NN along with PNN which achieved higher accuracy with less training time.

Reference [46] proposed a hierarchical Bayesian network where the original digit images were used directly as input rather than extracting the feature vectors. Reference [119] utilized Polynomial and Gaussian kernels to extract features and a Bayesian discriminant to classify the digits selecting the optimal parameters to minimize the classification error in a given database. This technique made full use of characteristics of each class distribution such as the class mean and covariance even though it did not average the covariance matrix of all classes. In all the eigenvalues of each class, a small value threshold (λ_0) was used to substitute the smaller eigenvalues to overcome the problem of the non-existing inverse matrix for covariance matrix so that the classification error in a given database was minimized.

4) k-NEAREST NEIGHBOURS (kNN)

k-Nearest Neighbours (kNN) [134] is a similarity-based image classification approach that is based on the nearest training samples in the feature space. During the training phase, this algorithm stores and labels all of the training images' feature vectors. Then it assigns a label or class to a test image based on the majority vote of its k nearest neighbors. The performance of this classifier in digit recognition was found to be highly dependent on the choice of k and the distance metric used to calculate the distances between neighbors [97]. The distance parameter is frequently chosen as Euclidean distance, City Block distance, or Manhattan distance, and the number of neighbors is typically 3, 5, and 7 [51], [135]. One disadvantage of kNN is that complex features may require significantly more time to perform inference [13]. Reference [51] demonstrated that kNN with Euclidean distance is effective in recognizing digits from local and global directional patterns. The authors also experimented with different values of k and found $k = 3$ to be the most effective, achieving an accuracy of 95.62% on the CMATERdb dataset. On the same dataset, [48] incorporated different variations of local binary patterns extracted from digits with kNN. Here, considering only one neighbor provided the best accuracy (96.70%). Despite that, the performance of kNN was poor compared to SVM and ANN in more complex datasets such as NumtaDB and Ekush [43], [97]. As a result, even though kNN requires lesser computational resources compared to SVM and ANN, the latters are often preferred.

5) MISCELLANEOUS

The zone density features of an unknown Bengali digit sample can be represented as a linear combination of the features extracted from the training samples from the same class [47].

Taking advantage of this property, [47] utilized Sparse Representation Classifiers [136] to build a dictionary of features from the training samples to represent Bengali digits. The authors used L1-minimization to efficiently compute the most sparse linear representation of the feature vectors of the test samples using the dictionary. Based on the representing samples, the class label was decided. The authors achieved 94% accuracy on the CMATERdb dataset.

Reference [112] utilized Decision Tree (DT) [137] to generate a hierarchy of rules in order to classify Bengali digits using water overflow from reservoir-based features. The authors achieved 91.98% accuracy on a self-curated dataset with 10,000 images. An improvement can be seen in the performance of the classifier if the number of digit samples is increased [113]. Recently, [99] constructed a Random Forest Classifier [138] by ensembling a set of decision trees that outperformed other classifiers such as Sequential Minimal Optimization (SMO) [139], Artificial Neural Network (ANN), and Logistic Regression (LR) [140] on the CMATERdb dataset achieving 98.50% accuracy. The authors performed two post hoc tests, namely Nemenyi test [141] and Bonferroni–Dunn test [142], [143] to further consolidate the superiority of the classifier.

VII. DEEP LEARNING-BASED APPROACHES

Traditional approaches of BHDR depended highly on extreme feature engineering, which limited the performance of the model with several constraints. However, to achieve high performance on large and challenging datasets under noisy conditions, researchers have found the remarkable generalization capacity of different Deep Learning (DL)-based algorithms extremely useful [144]. The DL-based techniques are capable of creating a more abstract representation of the data as the network grows deeper, by which the model can learn highly meaningful features that the ML-based methods often fail to grasp. Again, the high inter-class similarities along with diversity in writing styles make it extremely difficult to design highly efficient hand-crafted feature extractors for ML methods; which the DL-based methods can automatically learn. For these reasons, a major shift in paradigm is observed in recent works and almost all of them have focused on the DL-based pipelines.

This section discusses the Deep learning-based approaches of BHDR under two major categories: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). These two basic categories and their subcategories are shown schematically in Figure 21. Along with the discussion of the status quo, the trends in choices of different hyperparameters have been analyzed.

A. CNN-BASED ARCHITECTURES

A Convolutional Neural Network (CNN) based architecture consists of different types of layers such as convolution, activation, pooling, batch normalization, dense, etc. (Figure 22). The convolution operation produces activation maps containing meaningful patterns by applying a set of filters throughout the image. The output of this layer is passed through activation functions to introduce non-linearity. Moreover, the activation maps can be downsampled using pooling layers to reduce computational costs. The fully-connected layers receive a flattened activation map from the convolution block and consider further meaningful combinations of the learned features. Finally, a softmax layer is used to produce the class label.

The ability of Convolutional Neural Networks (CNN) to extract features utilizing the spatial dimension of images has been proven to be extremely useful in image classification tasks. Thus in recent times, most of the works on BHDR have been revolving around it one way or another and researchers have utilized this in different ways, such as, providing custom configurations of CNN-based models, and applying transfer learning by fine-tuning pretrained architectures, ensembling multiple models, etc.

1) CUSTOM CNN-BASED MODELS

In this section, we discuss the custom CNN-based approaches proposed in the literature. A summary of the proposed architectures is shown in Table 9.

Reference [35] proposed one of the earliest works in Bengali Handwritten Digit Recognition that showed how the handcrafted feature extraction phase can be replaced with a CNN architecture. The authors achieved 97.93% accuracy on a self-curated dataset consisting of 17,000 images using a CNN model consisting of two convolution layers and one fully-connected (FC) layer. However, despite working with a relatively simple dataset, the model failed to differentiate the inter-class similarities of certain classes such as (০, ৩), (৫, ৬), etc. A similar CNN model was used by [27] on a self-curated dataset of 70,000 images containing samples from different age and gender groups. An exhaustive grid search was applied to find the appropriate set of hyperparameters. Although the authors achieved a low validation error of 1.22% with such a simple model with two convolution layers, the likely reason might be due to the absence of challenging images in the dataset.

In another work, [38] applied the same architecture proposed in [35] with the same set of hyperparameters on 22,000 images of the ISI-HBN dataset and achieved 98.98% accuracy. They utilized rotation-based artificially generated patterns, which resulted in the improvement of performance compared to other works on the dataset till that time. However, the merit of this technique is questionable, since the authors applied a fixed degree of rotation to the images to create the patterns and reported the best result observed by repeating several such experiments. This is equivalent to hand fitting for the testing set. Moreover, the effect of other augmentation techniques was unexplored. Using the same architecture, the authors published another work exploring the possibility of Bangla-English mixed numeral recognition [37]. However, the authors did not provide any directions on separating similar Bengali and English digits (For example, Four in Bengali '৪' and Eight in English '8').

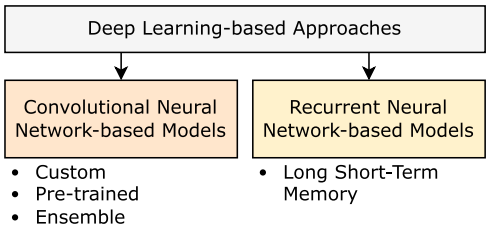


FIGURE 21. Taxonomy of the deep learning-based techniques used in Bengali Handwritten Digit Recognition.

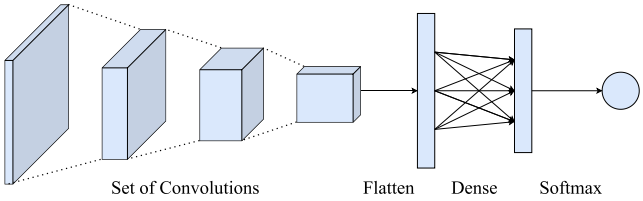


FIGURE 22. Generalized convolutional neural network-based architecture used in handwritten digit recognition.

In [67], the authors experimented with model ensembling techniques to determine its effect on the recognition performance. They trained three separate instances of the same CNN architecture used in [35]; one using the original images of the ISI-HBN dataset; one using the clockwise rotated version of the dataset (within a range of 10◦ to 40◦) and one using counterclockwise rotated version (within a range of 10◦ to 40◦). The authors reported an accuracy of 98.80% using the ensembled model. However, instead of training the three instances, a single model could be trained with all three of the variations of the dataset, which has been found to produce slightly higher accuracy in [68]. This helps to reduce the time complexity during training and inference. Moreover, the effects of other augmentation techniques were unexplored.

Reference [39] proposed an image augmentation method on the ISI-HBN dataset with a CNN model, which achieved 99.42% validation accuracy. The augmentation method downsamples the images into lower dimensions and then upsamples them again into the original dimension. This introduces a set of blurred images into the training set, reducing the overall quality, and enabling the model to learn to recognize poor-quality images. The best result was achieved by applying their proposed augmentation along with random rotation and shifting. Applying the proposed augmentation method, the authors achieved an increment of 0.17% accuracy compared to the 99.25% achieved by only applying random rotation and shifting. However, the effectiveness of this augmentation technique is debatable considering that a similar effect can be achieved by applying random blur on images, which will have a lesser computational requirement.

Recently, [58] proposed a densely connected deep CNN architecture, inspired by DenseNet architecture [145], called 'BDNet', which achieved 99.78% accuracy on the ISI-HBN dataset. The input is passed through a series of 'Dense' and 'Transition' blocks consisting of several 2D convolutions (Conv2D), ReLU, and Batch Normalization (BN) layers. The performance of this model was also tested on a self-curated dataset of 1,000 images where the model achieved an accuracy of 98.80%.

Reference [36] presented a hybrid model combining both hand-crafted and automatic feature extraction techniques that used the ISI-HBN dataset for training and validation sets, and the CMATERdb 3.1.1 dataset as the test set. The images were passed through a HOG feature extractor in one stream and through a two-layer CNN architecture in another. Then, the extracted features from both streams were combined in a larger feature vector and passed through an FC layer. This hybrid architecture successfully achieved 99.02% and 99.17% accuracy on the validation and test set, respectively. At that time, this was the state-of-the-art result. Since the test set consisted of a different dataset, the achieved result proved the generalization capability of the model in a completely unseen scenario. However, even though the hybrid model outperformed the CNN-only model mentioned in the paper, the CNN architecture was fairly simple. And this simple CNN-only architecture alone was able to achieve an accuracy of 98.87% compared to the 99.02% of the hybrid model on the ISI-HBN dataset. That means HOG had a minor contribution in increasing the accuracy, which might have been achieved by tweaking the layers in the CNN stream. Moreover, working with the HOG features might not turn out to be useful in challenging datasets like NumtaDB which contains lots of noise and variation [146].

Authors in [82] experimented with both the ISI-HBN and CMATERDB 3.1.1 datasets, where they formed four combinations of the train-test sets. The proposed pipeline consisted of Autoencoders and CNN which was able to provide high performance on all four of the combinations. The autoencoders helped in unsupervised pretraining and the CNN part utilized the learned weights for further extraction of meaningful features. In another work, the authors used the same architecture along with a newly introduced augmentation technique named 'Blocky Artifact' and achieved slight improvement in the performance [147]. A similar Autoencoder-CNN-based approach was proposed by [148]. However, since the encoder consisted of simple feed-forward layers, the model produced lower performance compared to the CNN-only approaches.

Another work on CMATERdb 3.1.1 dataset introduced a CNN model consisting of two convolutions and two FC layers [50]. The ability of the convolution layers in generating robust features invariant to shifting, rotation, scaling, etc. enabled the model to achieve an accuracy of 98.78%. Inspired by LeNet architecture [149], authors in [55] proposed a model consisting of three convolution layers, three average pooling, and one fully connected layer. The model achieved 97.60% accuracy on the same dataset. However, the accuracy can be improved with deeper networks and by applying more data augmentation techniques.

The introduction of the NumtaDB dataset presented the researchers with new challenges to handle highly diversified and heavily augmented noisy images. Reference [40] demonstrated how careful preprocessing techniques like

cropping, noise removal, thresholding, color inversion, etc. can boost the accuracy and also enable lighter architectures to achieve better performance in this dataset. The preprocessed dataset was fed into several ML and DL-based algorithms and resulted in improved performance compared to using the original dataset. The work also showed that CNN outperforms traditional ML-based techniques. On the same dataset, [54] proposed a six-layer CNN architecture to tackle the challenging NumtaDB dataset and achieved an accuracy of 92.72%. The performance of the network was validated in other English datasets as well. However, the model performed poorly on highly augmented images. The authors achieved 96.44% accuracy on the non-augmented portion of the dataset. Hence, efficient preprocessing and augmentation techniques can be undertaken to uplift the overall performance. Reference [62] achieved 96.79% accuracy on this dataset by ensembling two CNN architectures consisting of 12 and 9 convolutional layers respectively. The number of samples in the training set was increased by applying different augmentation techniques to tackle the noisy images. This resulted in a performance boost compared to the model trained with non-augmented images. However, the 12-layer CNN architecture was able to achieve 96.33% accuracy, which is comparable to the 96.73% accuracy achieved by the ensemble model. This 0.46% improvement comes at a cost of increased computational resources used by the second model.

One of the most recent works on the NumtaDB dataset has proposed a custom CNN model named 'DeepBanglaNet' which introduces two special types of operations named 'Analogous' and 'Changer' block for better recognition [63]. The Analogous block extracts features from a deeper level keeping the dimension of the activation map the same and can also ensure proper gradient flow using the residual connections. On the other hand, the Changer block can modify the dimension of the feature map to extract more generalized features. This work currently holds the state-of-the-art position with an accuracy value of 99.43%.

Reference [41] proposed a lightweight model consisting of 4 convolutional layers, which achieved a competitive accuracy with a faster execution time. The authors achieved an accuracy of 99.58%, 98.58%, and 92.65% on the ISI-HBN, BanglaLekha-Isolated, and CMATERdb 3.1.1 datasets, respectively. However, the authors tampered with the dataset by fixing some of the incorrect labelings and deleting some images which might have boosted the performance of the models. This makes it difficult to compare it with other works on these datasets. Reference [42] compared the performance of different ML algorithms such as *k*NN, SVM, Random Forest, Multilayer Perceptron (MLP), and a custom-designed CNN architecture on the Ekush, CMATERdb, NumtaDB, and BDRW datasets. All the algorithms were fine-tuned to get the best possible hyperparameters. The best performance was achieved by the six-layer CNN model inspired by the LeNet architecture [149]. The CNN model outperformed the traditional ML algorithms by at least 4% across multiple datasets. Yet, there was room for improvement in the performance of the NumtaDB and BDRW datasets as shown in Table 9. Specifically, all the algorithms faced difficulties recognizing the samples of the BDRW dataset due to its high variance and diversified background.

Reference [150] achieved 99.4% accuracy on a self-curated dataset using a CNN model consisting of two convolutions and one FC layer. Such a high accuracy with this simple model can be attributed to the absence of challenging and diversified digit samples in the dataset. In another work [57], authors proposed a CNN model containing six blocks, each consisting of convolution, pooling, and local normalization layers. The authors achieved 99.44% validation accuracy on the BHAND dataset, outperforming some of the state-of-the-art CNN architectures in the literature.

2) PRE-TRAINED CNN-BASED MODELS

Pre-pretrained CNN-based architectures are used in BHDR via transfer learning. Transfer learning refers to the use of an architecture that is trained to extract features using the training data of one domain and reused as a feature extractor in another domain [151]. Most of the transfer learning-based feature extractors that are used in image classification, namely AlexNet [152], VGG [153], GoogLeNet [154], ResNet [155], Xception [156] originated from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [157]. Along with them, MobileNet [158] and CapsuleNet [159] have also been used in BHDR. Studies have shown that fine-tuning these architectures for downstream tasks such as digit and character recognition can achieve satisfactory performance [160], [161]. A summary of the proposed architectures are shown in Table 10.

Although the pretrained models from ImageNet Challenge are available for quite a while, [61] was one of the first to study the use of transfer learning methods in BHDR. The authors used the VGG-16 network and achieved an accuracy of 97.09% on the NumtaDB dataset. The number of trainable parameters was reduced by freezing some of the layers of the architecture. The authors experimented by freezing five different combinations of layers from the 21 layers available in VGG-16. They are: freezing all layers except the final one, freezing none of the layers, freezing only the softmax layer, freezing everything except layer 16-20, and freezing only layer 16-20. The fifth combination achieved the highest accuracy of 97.09%. Freezing layers in this manner reduced the number of trainable parameters up to half of the original model.

On the same dataset, [162] experimented with four pretrained architectures, namely, AlexNet, MobileNet, GoogLeNet, and CapsuleNet. Among them, AlexNet was able to achieve the highest accuracy of 99.01%. According to the authors, although AlexNet had the highest number of parameters among the architectures considered, it required the least computation time to recognize digits.

Reference [163] compared the performance of LeNet-5, ResNet-50, VGG-11, and VGG-16 and found the modified VGG-11 architecture to have the best performance achieving

TABLE 9. Performances of the custom convolutional neural network-based architectures.

Reference	Year	Number of Layers and Filters	Dataset	Accuracy (%)
[35]	2015	2x Conv (6, 12) and 1x FC	Self-curated	97.93
[27]	2016	2x Conv (35, 45) and 1x FC	Self-curated	98.78
[38]	2016	2x Conv (6, 12) and 1x FC	ISI-HBN	98.98
[37]	2016	2x Conv (6, 12) and 1x FC	ISI-HBN	98.45
[67]	2016	2x Conv (6, 12) and 1x FC	ISI-HBN	98.80
[39]	2017	5x Conv (32, 32, 32, 32, 32) and 1x FC	ISI-HBN	99.42*
[58]	2020	Transition blocks and Dense Blocks	ISI-HBN	99.78
[36]	2016	HOG Features with 2x Conv (32, 32) and 1x FC	ISI-HBN	99.02*
			CMATERdb 3.1.1	99.17
[82]	2016	Autoencoders with 3x Conv (32, 32) and 1x FC	ISI-HBN	98.29
			CMATERdb 3.1.1	99.50
[68]	2017	2x Conv (6, 12) and 1x FC	ISI-HBN	98.98
			CMATERdb 3.1.1	99.65
[50]	2017	2x Conv (32, 64) and 2x FC	CMATERdb 3.1.1	97.60
[55]	2019	3x Conv (32, 64, 128) and 2x FC	CMATERdb 3.1.1	97.60
[40]	2018	2x Conv (32, 64) and 1x FC	NumtaDB	91.30
[54]	2018	6x Conv (2x 32, 2x 128, 2x 256) and 2x FC	NumtaDB	92.72
[62]	2018	Model 1: 12x Conv (2x 32, 2x 64, 2x 128, 3x 256, 521, 2x 512) and 2x FC. Model 2: 9x Conv (2x 32, 2x 64, 2x 128, 3x 256) and 2x FC	NumtaDB	96.79
[63]	2020	Series of Analogous and Changer blocks (13x Conv) and 2x FC	NumtaDB	99.43
[150]	2017	2x Conv (32, 64) and 1x FC	Self-curated	99.40
[57]	2020	13x Conv (2x 32, 3x 64, 2x 128, 2x 256, 2x 384, 2x 512) and 3x FC	BHAND	99.44*
[41]	2019	4x Conv (2x 32, 2x 64) and 2x FC	ISI-HBN	99.58
			CMATERdb 3.1.1	92.65
			BLI	98.65
[42]	2019	6x Conv (2x 28, 2x 128, 2x 256) and 2x FC	Ekush	99.71
			CMATERdb 3.1.1	99.25
			NumtaDB	98.94
			BDRW	96.65

* denotes validation accuracy

99.80%, 99.66%, 99.25% accuracy on ISI-HBN, CMATERdb, and NumtaDB datasets, respectively. They introduced batch normalization and zero-padding layers to the vanilla VGG-11 architecture. The number of filters in the convolution layers and the neurons in the fully-connected layers were tuned to reduce the number of trainable parameters. The modified architecture had 7.7M parameters, which is significantly less compared to the 28M parameters in the vanilla architecture.

3) ENSEMBLE OF MULTIPLE MODELS

Combining the results obtained by individual classifiers in an ensemble architecture can be done in several manners. A few of the usual approaches are max-voting, probability average, etc. In a max-voting ensemble, among the class labels obtained from the individual classifiers, the highest occurring label is selected as the final label for the input sample. In a classifier network, the probabilities for each label for a particular sample are passed through an argmax layer which provides this class label. These probabilities are utilized by the probability averaging ensemble. The class-wise probability values across all models are aggregated for each class and divided by the total number of classifier networks. This provides the average probability for each class, combining the contribution from all the models. Then, the probability values are passed through an argmax layer to predict the final class label. A summary of the ensemble-based approaches can be found in Table 10.

Reference [71] used a max-voting ensemble architecture of three pretrained Xception architectures on the NumtaDB dataset. According to the depthwise separable convolution blocks used in Xception architecture were suitable for reducing the parameter count. Additionally, considering the large size of the model, dropout and L2 regularization were used to avoid overfitting. To train the individual architectures, different preprocessed training images were used as input. The authors achieved an accuracy of 96.694% using the ensemble architecture, whereas the individual models achieved 96.499%, 96.305%, and 96.125%. Nevertheless, the authors reported weighted average accuracy, where the significantly distorted images of the test set were weighted significantly lower, misrepresenting the actual performance of the models.

Reference [73] used an ensemble architecture of the popular pretrained models, ResNet34 and ResNet-50, that achieved an accuracy of 99.3359% on the NumtaDB dataset. Each of the models individually achieved an accuracy of 98.686% and 99.094%, respectively. To achieve the final accuracy, the authors used a weighted voting ensemble of six architectures, where five of which were ResNet34 and one of them was ResNet50. Each model had a different set of hyperparameters, training, and validation images. However, the accuracy achieved by a single ResNet-50 being close

to the ensembled model negates the necessity of the huge increase in computational resources to achieve a meager 0.2419% increase in accuracy. However, both of the works have ensembled the variants of the same baseline CNN model. The impact of ensembling different types of architecture can also be investigated. Analysis should be conducted on how such ensembling techniques impact the computation and time constraints compared to the performance improvement it provides.

4) CHOICE OF HYPERPARAMETERS

Hyperparameters are the parameters whose values are set before the learning process and not updated during training. Properly tuning them can result in faster convergence to the global minima [164]. The trend in hyperparameters selection during the training of the CNN models is shown in Table 11.

a: INPUT DIMENSION

A common practice that has been observed throughout the existing literature in this domain is to convert the input images of RGB color space into binary or grayscale to reduce the channel dimension [35], [36], [37], [38], [41], [42], [50], [54], [150]. This results in a reduction of computational cost during training and inference without affecting the accuracy. Another common practice was to reduce the input dimension at the very beginning of the model. Here, the input dimension should be selected in such a way that it contains as fewer pixels as possible but still maintains the clarity to convey enough information to the model for digit recognition. Taking the resolution too high can increase the computational cost, and too low can distort the digits. So this trade-off should be considered when choosing the input size.

b: NUMBER OF CONVOLUTION LAYERS

The main building block of a CNN architecture is the convolution layer which applies a set of filters to create activation maps containing meaningful information utilizing the spatial information of the digit image. Starting with the low-level features produced by the earlier layers, the deeper convolution layers combine them to produce a deeper meaning of the data [165]. In the BHDR literature, the number of convolution layers in different models varied with the complexity of the dataset. If the dataset is simpler, i.e., had comparatively simple backgrounds and fewer variations in handwriting, meaningful features could be learned with only a few convolution layers [35], [36], [37], [38], [50], [67], [150]. Besides, some works manually cropped the digit regions, which also enabled the simpler CNN models to gain higher accuracy [37], [38], [67]. However, such rigorous preprocessing is not recommended considering the variety in real-life handwritten numerals. As the size, variation, and background of the samples in the dataset got diversified, the authors used complex models with more layers. For example, since the NumtaDB dataset contains different challenging samples, most of the works have utilized CNN models with more than six layers to train their models to achieve acceptable performance on this dataset [42], [54], [56], [57], [58], [62], [63], [166].

c: NUMBER OF FULLY CONNECTED (FC) LAYERS

The fully-connected layers take the flattened output produced by the CNN block, representing meaningful features extracted from the input. The number of FC layers varied in the BHDR literature based on the size of the flattened layer. If the number of nodes in the flatten layer was larger, one or two dense layers were added before the final prediction layer to consider further combinations of these features for robust digit recognition [39], [41], [42], [50], [54], [56], [57], [62], [63], [166]. However, if the number of nodes in the flatten layer was smaller, it was directly fed to the final output layer [35], [37], [38], [67], [150], [167].

d: POOLING AND BATCH NORMALIZATION LAYERS

Pooling layers reduce the number of parameters and computations in the network by downsampling the spatial size of the activation. It works on each feature map independently. In most of the BHDR models, the authors have used a Pooling layer after every one or two convolution layers. It was observed that if the number of convolution layers was higher, one pooling layer was used after every two layers [42], [54], [57], [62]. Otherwise, pooling was applied after every convolution layer [35], [68]. Applying pooling layers after every convolution layer in deeper models aggressively downsamples the feature maps, hampering the overall performance of the digit recognition models. Max pooling was the most popular choice in the literature, but a few works also used average pooling [37], [63]. Some works also used Batch Normalization layers [168] before convolution layers to speed up the learning process by normalizing the input layers [54], [57].

e: SIZE OF FILTERS

Filters are the learned weights of CNN architecture that are applied over the input to capture patterns. Most of the existing CNN architectures for Bengali numeral recognition have gradually increased the number of filters as they went towards the deeper layers to capture combinations of the features as much as possible. Common choices for filter size were 3×3 and 5×5. The use of bigger filters incurs more computation, while also aggressively downsampling the data. On the contrary, a filter of any size can be mimicked using a consecutive number of small filters like 3×3, resulting in less amount of computation [169]. In the case of a larger input dimension, a 7×7 filter has also been utilized to rigorously downsample the feature space only in the first layer [63]. The authors set the size of the filter in the pooling layer to be 2×2.

f: DROPOUT RATE

The dropout technique is used to introduce regularization in the model by randomly switching off nodes during the training phase [170]. Some of the works used dropout layers before the final output layer [42], [54], [56], [62]. In some other works, if the model contained multiple FC layers,

TABLE 10. Performances of the transfer learning and ensemble-based architectures.

Reference	Year	Best Performing Model	Number of Layers	Parameter Count (millions)	Dataset	Accuracy (%)
[61]	2018	VGG16	21	134.7	NumtaDB	97.09
[162]	2020	AlexNet	8	61	NumtaDB	99.01
[163]	2019	Modified VGG-11	32	7.7	NumtaDB	99.25
					ISI-HBN	99.80
					CMATERdb 3.1.1	99.66
[71]	2018	Ensemble of Xception models	126	126	NumtaDB	96.69
[73]	2018	Ensemble of ResNet-34 and ResNet-50	34 and 50	21.5 and 23.9	NumtaDB	99.34

TABLE 11. Hyperparameters of the CNN-based architectures.

Reference	Input Dimension	Pooling	Dropout	Batch Size	Epoch	Learning Rate	Optimizer
[35]	$28 \times 28 \times 1$	–	N/A	–	80	–	–
[27]	$32 \times 32 \times 1$	Max	N/A	–	–	–	–
[38]	$28 \times 28 \times 1$	–	N/A	100	310	–	–
[37]	$28 \times 28 \times 1$	Average	N/A	50	300	1	–
[67]	$28 \times 28 \times 1$	–	N/A	50	500	–	–
[68]	$28 \times 28 \times 1$	–	N/A	50	300	1	–
[39]	$28 \times 28 \times 1$	Max	0.5 after every conv and FC layer	100	100	–	Adadelta
[58]	$32 \times 32 \times 3$	Average	0.9	32	150/125	0.009	SGD
[36]	$28 \times 28 \times 1$	Max	N/A	–	100	–	Adadelta
[50]	$32 \times 32 \times 1$	Max	0.5 in between convolution layers	–	30	–	–
[55]	$32 \times 32 \times 3$	Max	N/A	–	40	–	Adam
[40]	$28 \times 28 \times 1$	–	0.5 in between convolution layers	64	30	0.003	–
[54]	$32 \times 32 \times 1$	Max	0.2 after FC1	64	30	0.001	Adam
[62]	$64 \times 64 \times 1$	Max	0.2 before softmax	–	30 and 6	0.001 and 0.0001	SGD in Model1 and Adam in Model2
[63]	$128 \times 128 \times 3$	Average	N/A	32	100	0.0001	Adam
[150]	$28 \times 28 \times 1$	Max	N/A	–	4000	–	–
[57]	$32 \times 32 \times 1$	Max	0.5 after first 2 FC layers	–	19550	0.001	RMSprop
[41]	$28 \times 28 \times 1$	Max	0.25 after convs and 0.5 before softmax	86	30	0.001 with LR Reduction	Adam
[42]	$28 \times 28 \times 1$	Max	0.2 before output	–	–	–	Adam
[71]	71×71	Max	0.25 before every residual block	124	–	0.001	Adam
[61]	41×41	Max	N/A	16	50	0.0001	AdaDelta
[73]	–	–	N/A	–	2-3	0.0055 - 0.01	–
[163]	32×32	Max, Global Average	0.2 after 6th conv layer, 0.25 and 0.4 respectively after the two FC layers	128	200	0.001	Adam

– denotes information was not available in the cited literature
N/A denotes dropout layer was not used in the cited literature

dropout was used after each of them with different probabilities [57]. In both scenarios, the dropout layer was not used in the intermediate layers. The possible reason for omission might be to preserve high-level features in the intermediate layers [171]. However, several works also achieved a positive result by utilizing dropout after both the convolution layers and FC layers. For example, [41] applied a dropout of 25% after every two convolution layers and 50% before the final FC layer. [39] applied 50% dropout after every convolution and FC layers. [50] and [40] also found the usage of dropout useful in between the convolution layers.

g: OPTIMIZER

Optimizers play a vital role to update the weights of the model in order to reduce the overall loss. In the present literature, the most popular choice has been the Adam optimizer [172], since it provides faster convergence in most of the cases [41], [42], [54], [55], [63]. However, other optimizers have also shown promising results, such as Adadelta in [36], [39], [61], RMSprop in [57], and SGD in [39]. In the ensemble model proposed in [62], one model was trained with Adam, and another model was trained with an SGD optimizer.

h: ACTIVATION FUNCTION

The activation function helps the performance of the architecture by introducing non-linearity into the output of a neuron [173]. In the existing literature of BHDR, ReLU [174] has been widely used as the activation function [54], [55], [56], [63]. It removes the negative values from the activation map by setting them to zero. Thus, it increases the nonlinear

properties of the decision function and removes the chances of vanishing gradient without affecting the receptive fields of the convolution layers [175].

i: CHOICE OF LEARNING RATE

Learning Rate (LR) helps the optimization algorithm to control the step size for each iteration along the downward slope. Large learning rates might not hold for deep neural architecture since the model learns suboptimal solutions and might not converge to the global minima [176]. Using an adaptive learning rate might be handy in such scenarios [177]. Some popular choices of LR in BHDR literature are 0.001 [41], [54], [57], [71], [163], 0.0001 [56], [61], [63], [166], etc. Reference [41] manually monitored the results up to 30 epochs, manually reduced the learning rate, and continued training for 5-10 more epochs. This process was repeated a couple of times. Reference [58] used an LR of 0.009 till 80 epochs and then reduced it by a factor of 0.15 until the model converged. Reference [62] used two different LRs (0.001 and 0.0001) to train the ensembled CNN networks.

j: BATCH SIZE

Batch size denotes the number of samples that will be passed through the model before updating weights once. Mini-batch is usually preferred for large datasets. If the optimization algorithms work with the entire batch for every update, the overall training process slows down [178]. In addition, smaller batch size is used due to the limitation of memory. Batch size does not contribute much to the performance, apart from increasing the training speed. Some of the common choices of batch sizes in the present literature are 32 in [58], [63], 50 in [37], [67], 64 in [54], [56], 86 in [41], 100 in [39], etc.

k: NUMBER OF EPOCHS

The number of epochs denotes the count of completed passes through the dataset by the optimizer during training. Choosing the optimal value for this hyperparameter depends highly on the architecture [164]. An inverse proportional relationship between model depth and the number of epochs can be seen in the existing BHDR literature. For example, models with more than six convolution layers required 30 epochs to converge [54], [56], [62], [166]. However, several authors trained deeper models for even more epochs, such as [55] trained for 40 epochs, [35] for 80 epochs, [39], [63] for 100 epochs. On the other hand, if the model is shallow, it took longer epochs to converge. For example, [37], [38] and [67] trained the model for around 300 epochs to converge. Similarly, [150] trained the model for 4000 epochs since the model had to learn from around 80000 images with a simple 2-layer CNN. In the case of transfer learning-based approaches, it is observed that they usually require a lower number of epochs than custom CNN-based architecture that is trained from scratch. This is because these architectures are already pretrained to extract features from a huge set of image samples.

Instead of training the model for a fixed number of epochs, the early stopping approach can be considered to limit the maximum number of epochs required [179]. In this method, regardless of the number of epochs selected, the training is stopped as soon as the model converges. Such an approach was adopted by [36], where the authors initially set the number of epochs to be 100, however, the training was stopped at 55 epochs as soon as it converged. Similarly, [58] set the number of epochs to be 150, but the model converged at 125 epochs. Another approach can be varying the number of epochs based on the learning rate, giving the model enough opportunity to learn as long as possible [41].

B. RECURRENT NEURAL NETWORK-BASED ARCHITECTURES

A neural network that is specialized for processing a sequence of data for tasks involving sequential inputs such as speech and language is referred to as a recurrent neural network (RNN). The output of a specific element in an RNN-based architecture is dependent not only on that element but also on the previous elements in the sequence, which together serve as a 'memory' for the architecture as shown in Figure 23.

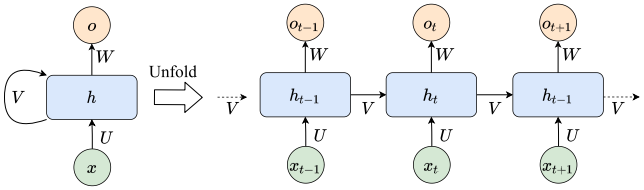


FIGURE 23. Generalized Recurrent Neural Network-based architecture for handwritten digit recognition.

Previous research works have proven CNN-based architectures to be very effective for image classification tasks. However, various authors reported that CNN-based architectures tend to misclassify specific Bengali digits where the shape and size are similar (such as '৯' and '৯') [69], [162], [180]. To improve the overall performance, the efficiency of RNN-based architectures has also been explored for its ability to learn contextual information through recurrent connections. Specifically, a variation of RNN named Long Short Term Memory (LSTM) based networks is mostly used in this specific task. Unlike RNN and other DL methods which use gradient-based learning, LSTM is proven to be less prone to the vanishing gradient problem which contributes to the overall performance [181], [182]. Although LSTM is well-known in classifying time-series data, several authors have reported satisfactory performance of LSTM-based networks in this domain as well [183], [184]. These models have the ability to extract features utilizing handwriting strokes which can be useful to differentiate the numerals having similar shapes and sizes. A summary of the RNN-based architectures is provided in Table 12.

Reference [185] proposed one of the first works in BHDR using LSTM-based architectures. Their comparison showed that the proposed pipeline outperformed some pre-dominant architectures till that time, namely, SVM, MLP, CNN, etc.,

TABLE 12. Performances of the recurrent neural network-based architectures.

Reference	Year	LSTM Layers (Hidden Units)	Dropout	Epoch	Batch Size	Learning Rate	Optimizer	Loss Function	Accuracy (%)
[185]	2016	2x (50, 100)	N/A	460	50	–	–	–	98.25
[69]	2019	2x (100, 50)	0.2	370	50	0.02	Adadelta	Categorical Cross-Entropy	98.25
[180]	2020	2x (50, 30)	N/A	1500	–	–	–	–	98.03

– denotes information was not available in the cited literature
N/A denotes dropout layer was not used in the cited literature

and achieved an accuracy of 98.25% on the ISI-HBN dataset. The proposed architecture consisted of two LSTM layers with 50 and 100 hidden units respectively, with a feed-forward layer in between them. Both layers used Tanh activation functions. The LSTM layers were followed by a reshape layer and a dense layer. A subsequent work by the same authors also experimented with three configurations of LSTM architectures and showed that an LSTM with two layers performed better than the one having a single layer [69]. After experimenting with different choices of learning rates and batch sizes, the authors concluded that smaller values for these hyperparameters resulted in better performance. The smaller learning rate allowed the optimizer to take smaller steps to apply the gradients. On the other hand, the smaller batch size provided regularizing effect [186], [187], [188].

Although the intention of both the works was to demonstrate the superiority of LSTM-based architectures, several existing works using custom CNN-based architectures on the same dataset achieved a better performance [37], [38], [67]. Again, in addition to experimentation with single-layer and two-layer LSTMs in [69], the authors could have tested how the models perform when the number of LSTM layers is increased. The impact on computational resources compared to other architectures could also have been explored.

In another work, the authors took a unique preprocessing approach that converted the input image into a directed acyclic graph with a fixed number of equidistant points [180]. Then it was fed to an LSTM architecture containing two LSTM layers with 50 and 30 units respectively.

The proposed architecture achieved 98.03% accuracy on the ISI-HBN dataset, which was 2.58% higher than the performance achieved by applying a single layer. Nevertheless, exploring the impact of LSTM-based architectures on reducing computation resources along with ensuring a comparable performance can be useful. On the other hand, the claim that LSTM can alleviate the difficulty of correctly classifying Bengali digits with similar shapes and sizes is not put to test in any of the existing literature. Future research can focus on these things as well.

VIII. BROADER APPLICATIONS OF HANDWRITTEN DIGIT RECOGNITION

In recent times, some works have proposed handwritten digit recognition models with a broader scope than only classifying numerals. Such systems can be utilized to build useful tools focusing on different real-life applications.

A. HANDWRITTEN MATHEMATICAL EXPRESSION EVALUATION

Reference [78] proposed a CNN-based architecture named 'MathNet' to recognize different components of handwritten mathematical equations containing Bengali digits. The authors trained the model using a dataset of 32,400 images consisting of 10 classes of Bengali handwritten digits and 44 other classes belonging to operators from algebra, set-symbols, limit, calculus, symbols of comparison, delimiters, etc. The dataset included 6,000 numeral images from the 'Ekush' dataset and the symbol images were self-collected. Although the model achieved high performance, the classes were assumed to be isolated and no segmentation was proposed, which leaves the opportunity for future improvements. Besides, the task could use more challenging images for numeral detection like 'NumtaDB' instead of 'Ekush' for digit images, which is more likely to represent real-world scenarios.

This work was further extended in [189] by proposing a complete pipeline for handwritten Bengali mathematical expression recognition and simplification. It utilized an existing line and character segmentation technique and applied the 'MathNet' model for classification. Nevertheless, the work is still in a very preliminary state and less likely to generalize in real-world examples due to certain strict assumptions by the authors. For example, the line segmentation technique assumes that there will be a certain amount of gap between two lines. But in a real-world scenario, the handwriting might not be in such an ideal form.

However, this area has a long way to go compared to the present state of mathematical expression recognition on other scripts [190]. Advanced techniques like stroke extraction [191] are found to be useful on the publicly available mathematical expression dataset CROHME [192]. Curating such benchmark datasets for Bengali handwritten mathematical expression recognition along with a robust pipeline might be useful for many practical applications.

B. HANDWRITTEN DIGIT GENERATION

Despite the existence of several works on handwritten character generation in recent times in different languages, the field of Bengali handwritten digit generation is still comparatively unexplored [193], [194], [195]. [57] proposed a semi-supervised Generative Adversarial Network (SGAN) [196] for Bengali handwritten digit generation. Both the generator and the discriminator blocks had a series of convolution and fully-connected layers. The model had a high

loss of 0.368 and 0.694 in the discriminator and the generator, respectively.

Reference [197] trained a DCGAN-based architecture [198] to generate Bengla handwritten digits from random noise. The authors experimented with four publicly available datasets: CMATERdb, BanglaLekha-Isolated, ISI-HBN, and Ekush for this task. Once the model was trained with a merged dataset of 58,827 images, it achieved a loss of 0.647. The final output was promising, but still, there is plenty of room for improvement.

One negative aspect of both works is that the generated datasets were not tested for digit recognition. Experiments should be designed where these generated images are used in the training phase of a digit recognition model to understand whether these systems are capable of generating adverse samples to improve the performance of the recognition models.

C. HANDWRITTEN DIGIT DETECTION

Reference [199] proposed a pipeline to detect Bengali handwritten digits using a region proposal network. An architecture combining Faster-RCNN [200] with InceptionV2 [201] was used for the detection and classification of digits. Due to the absence of any publicly available dataset suitable for this task, the authors generated a dataset using isolated images from the NumtaDB dataset to simulate the effect of a page with handwritten digits on it.

Despite the model achieving high performance, its effectiveness of this in real-life applications remains untested. The images in the generated dataset were placed randomly with the strict assumption that the digits should not be too densely packed, which might not be the case in real-world handwriting. Again, the images were taken from a comparatively less challenging portion of the NumtaDB dataset. Further experiments are needed to understand how it performs on images with a high degree of variability. Additionally, strict distance-based rules were applied to understand if two digits belong to the same number, which might not work in real-life scenarios.

D. MULTILINGUAL DIGIT RECOGNITION

At present, most of the solutions are designed for single script numeral recognition. However, with the ever-growing international correspondence, especially in countries where multiple languages are being used in daily activities, the capability of a model to recognize multiple scripts is highly beneficial.

Reference [202] worked to recognize different Indic numerals in a single document page. The authors concentrated on the four most frequent scripts in the Indian subcontinent: Bengali, English, Hindi, and Oriya. Different state-of-the-art Deep CNN architectures were employed to solve the task, and Inception-v4 was the best performing model among them. However, the experiments were performed on individual datasets of each script, which conflicts with the goal of building a script invariant handwritten digit identification system. Similarly, [203] proposed a CNN-based approach for multilingual numeral recognition, but the experiments were limited to applying the model separately to each dataset. Future endeavors should concentrate on mixing samples from all the scripts to generate a combined dataset and then judge the performance of the model. Since different scripts have high inter-class similarities (such as Eight (8) in English and Four (৪) in Bengali), it can pose a huge challenge for the models to recognize such occurrences.

In this regard, [59] proposed a novel feature extraction technique named 'Symbolization of binary images (SBI)' for the recognition of handwritten numerals in different scripts. The authors achieved above 95% accuracy with the SVM classifier on each of the four popular scripts: Arabic, Bengali, Devanagari, and Latin. When one of the scripts was mixed pairwise with Latin, the performance was above 91%. Finally, once all four scripts were mixed, the model still had a 90.98% recognition rate, justifying its capability to recognize the numerals invariant of the script class. In another work, the authors introduced the 'Quadrangular Transition Count' feature extraction technique which resulted in satisfactory performance in script invariant handwritten digit recognition [111].

However, this domain of mixed numeral recognition has a long way to go since distinguishing these high inter-script similarities poses an extremely difficult challenge even for a human being. Using separate models for language detection and digit recognition has been proven to be useful in multi-script images [204]. Such approaches can be adopted for mixed numeral detection of Bengali and other scripts. Furthermore, benchmark datasets containing mixed script handwritten documents can be proposed in this regard to provide baselines and make the performance of different models comparable [205].

E. HANDWRITTEN DIGIT STRING RECOGNITION (HDSR)

Recognizing handwritten digits from different strings is a challenging task since it has to deal with several complex scenarios such as the presence of noise, broken digits, digits touching each other, etc. In this regard, some recent works have introduced segmentation-free approaches [206], [207], [208]. Reference [209] utilized different techniques of HDSR to detect and recognize handwritten English digits from historical handwritten document images. However, this field is completely unexplored in the context of Bengali handwritten string recognition. Developing such systems can be extremely useful in application domains like processing bank checks, postal codes, handwritten forms, etc. Application in these domains can help build assistive technology for visually impaired people [210].

IX. RESEARCH GAPS AND FUTURE DIRECTIONS

Based on the experience of detailed literature review, some research gaps have been identified in this field, which requires intensive research to contribute more. Therefore, a list of future research directions is suggested below:

- Most of the existing works assume the digits to be in isolated form. A complete pipeline is necessary to

check how this can be applied to entire documents. Even though a few works in BHDR have focused on it, they are constrained by a number of challenges. To extract digits from a handwritten document, a well-designed line and digit segmentation algorithm can be proposed. Grouping the digits belonging to a single number is another challenge in this regard.

- Although lots of preprocessing techniques have been proposed in the existing literature, they are mostly considered localized solutions for specific datasets. These techniques might not hold up against real-life samples if they contain different types of noises, background clutter, occlusion, etc. For this reason, instead of preprocessing the dataset, it would be better to focus on augmentation to ensure that the models learn to recognize digits in various scenarios.
- Despite the main focus of most of the works have been on improving the accuracy, lightweight models can be proposed that are suitable for low-powered devices such as smartphones and embedded systems. An in-depth analysis of training time and inference time for such models is yet to be explored. Such analysis of the time complexities of different methodologies can be extremely useful while picking the suitable model satisfying the resource constraints across different applications.
- Instead of focusing on digit classification only, future research efforts can be concentrated on extracting age, gender, location, educational status, etc. information from handwritten digits. These endeavors can be facilitated by various modalities provided in Ekush and BanglaLekha-Isolated datasets. Extraction of such information can facilitate forensic investigation of different handwritings.
- Most of the existing literature in BHDR has not considered a detailed ablation study of the proposed pipeline. An in-depth analysis of the contribution of different components of the model to the overall result may not only shed light on the pros and cons of the proposed model but also provide insights for future researchers to work on. Furthermore, proper benchmarking attempts are required to compare the models with previous literature. The heterogeneity of reported results makes it difficult to position a model with respect to earlier works.
- To further understand the generalization capability of the models, one approach can be training the model on one dataset and testing it on another. Few such works can be seen in the existing literature. The merit of this experiment is that samples of the same dataset can have similar biases within them. Even when the test set remains completely unseen, the model can get a sense of the variety of the images in the training phase, which can affect the classification accuracy. Validating the model on a completely different dataset can alleviate this issue.
- Several existing works on BHDR utilize multiple classifiers to provide a comparative understanding of their performance of them with a view to identifying the best one. However, while working with multiple datasets across different classifiers, it is difficult to provide a generalized comparison. To address the issue, various parametric and non-parametric statistical analyses can be performed to standardize the experimental results of the classifiers which can provide better insight into their performance [211], [212].
- With the rapid growth in the usage of smart devices, storing data from online handwriting has become a common phenomenon. To convert this written form of information into editable electronic content, Online Handwritten Recognition has become an active area of research in recent times [3], [213]. Although several research works have been done on several scripts such as, Arabic [8], Chinese [214], Devanagari [215], etc, it is yet to be explored with Bengali numeral. Research attempts can be made to propose such online handwritten digit recognition systems suitable for end-level smart devices.
- One of the key challenges in ensuring the adoption of the deep learning-based BHDR pipelines in various applications is the lack of transparency [216]. In the case of the machine learning-based models, the use of handcrafted features usually provided better visualization of what worked and what did not. On the other hand, DL-based works on BHDR often treat the model architecture as a ''black boxes''. In this regard, adopting recent trends in explainable artificial intelligence can provide better insight into how the models identify the digits. This interpretability of the models can ensure effective performance [217] and improve the user experience by helping the end users to trust the decision-making process of the BHDR pipelines [218].
- Research attempts can be concentrated on generating handwritten digits, which can add value to the overall performance of the classifier. On this note, in recent times, there has been a trend in proposing models having fewer dependencies on very large datasets [219]. The usability of different few-shot learning-based approaches can be explored in the field of Bengali handwritten digit recognition [220], [221].
- In recent times, some broader applications of BHDR such as mathematical expression recognition, numeral recognition with a mixture of digits from multiple scripts, detecting digits from handwritten strings, etc., have gained interest. The status quo of such works in the field of BHDR is still in a preliminary state. Such application fields can further be explored by curating publicly available benchmark datasets, preparing baselines, and proposing robust models for each of such use cases.

X. CONCLUSION

In this paper, the characteristics and inherent ambiguities of Bengali handwritten digits along with comprehensive insights on the state-of-the-art works that are proposed in the last two decades have been analyzed. Even though a lot of

work has been done in this domain, addressing these ambiguities still remains a major challenge for future researchers. In this regard, the use of contextual information such as texts, images, etc. surrounding the digits to be recognized might be an interesting research avenue. The discussion regarding the available benchmark datasets highlights the usefulness of different modalities such as gender, age, aesthetic quality index, etc. available with the dataset in various biometric applications. Again, utilization of these benchmark datasets in future works can provide better insight into the comparative performance of different models. Furthermore, the assessment of the preprocessing and augmentation techniques used over the years indicates heavy reliability in targeted preprocessing resulting in BHDR pipelines with reduced generalization capabilities. This issue requisites a shift from preprocessing to augmentation that can result in robust pipelines capable of performing well in unknown scenarios.

The last two decades have seen a huge shift in the recognition process transforming from classical machine learning-based approaches to the deep learning-based techniques. Despite the fact that deep learning-based approaches helped get rid of cumbersome handcrafted feature-based approaches increasing the generalization capability of the recognition pipelines, there are still a lot to explore in terms of model architectures and hyperparameters of the deep learning-based approaches. Careful consideration should be provided to the overall training and testing process to better analyze the performance of the models. In addition to exploring the newer and better architectures, exploitation of the existing ones are also necessary to harness their full potential.

Finally, the use of BHDR pipelines in real-life scenarios, such as mathematical expression evaluation, multilingual digit, recognition, handwritten digit string recognition, etc. can further empower seamless connection and cooperation between the physical and the digital world. To facilitate that, this paper will serve the purpose of a compendium for researchers who are interested in the science behind offline BHDR, instigating the exploration of newer avenues of relevant research, which may lead to better recognition of offline Bengali handwritten digits in different application areas.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

ACKNOWLEDGMENT

The authors are thankful to Mohammad Ishrak Abedin, Department of Computer Science and Engineering, Islamic University of Technology, for his valuable time and support in proofreading the article.

REFERENCES

[1] A. Khatun, M. S. Shahriar, M. H. Hasan, K. Das, S. Ahmed, and M. S. Islam, "A systematic review on the chronological development of Bangla sign language recognition systems," in *Proc. Joint 10th Int. Conf. Informat., Electron. Vis. (ICIEV) 5th Int. Conf. Imag., Vis. Pattern Recognit. (icIVPR)*, Aug. 2021, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9564157

[2] U. Pal, R. Jayadevan, and N. Sharma, "Handwriting recognition in Indian regional scripts: A survey of offline techniques," *ACM Trans. Asian Lang. Inf. Process.*, vol. 11, no. 1, pp. 1–35, Mar. 2012, doi: 10.1145/2090176.2090177.

[3] H. Singh, R. K. Sharma, and V. P. Singh, "Online handwriting recognition systems for indic and non-indic scripts: A review," *Artif. Intell. Rev.*, vol. 54, no. 2, pp. 1525–1579, Feb. 2021, doi: 10.1007/s10462-020-09886-7.

[4] P. K. Singh, R. Sarkar, and M. Nasipuri, "A comprehensive survey on Bangla handwritten numeral recognition," *Int. J. Appl. Pattern Recognit.*, vol. 5, no. 1, pp. 55–71, 2018, doi: 10.1504/IJAPR.2018.090516.

[5] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63–84, Jan. 2000. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/824821

[6] M. Silfverberg, "Historical overview of consumer text entry technologies," in *Text Entry Systems*, I. S. MacKenzie and K. Tanaka-Ishii, Eds. San Mateo, CA, USA: Morgan Kaufmann, 2007, pp. 3–25, ch. 1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123735911500012

[7] M. T. Parvez and S. A. Mahmoud, "Offline Arabic handwritten text recognition: A survey," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 1–35, 2013, doi: 10.1145/2431211.2431222.

[8] B. M. Al-Helali and S. A. Mahmoud, "Arabic online handwriting recognition (AOHR): A survey," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 1–35, Oct. 2017, doi: 10.1145/3060620.

[9] A. Singh and A. S. Bist, "A wide scale survey on handwritten character recognition using machine learning," *Int. J. Comput. Sci. Eng.*, vol. 7, no. 6, pp. 124–134, Jun. 2019. [Online]. Available: https://www.ijcseonline.org/full_paper_view.php?paper_id=4518

[10] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with MNIST and EMNIST," *Appl. Sci.*, vol. 9, no. 15, p. 3169, Aug. 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/15/3169

[11] J. Memon, M. Sami, R. A. Khan, and M. Uddin, "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)," *IEEE Access*, vol. 8, pp. 142642–142668, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9151144/

[12] K. Cheng, R. Tahir, L. K. Eric, and M. Li, "An analysis of generative adversarial networks and variants for image synthesis on MNIST dataset," *Multimedia Tools Appl.*, vol. 79, nos. 19–20, pp. 13725–13752, May 2020, doi: 10.1007/s11042-019-08600-2.

[13] Y. A. Nanehkaran, D. Zhang, S. Salimi, J. Chen, Y. Tian, and N. Al-Nabhan, "Analysis and comparison of machine learning classifiers and deep neural networks techniques for recognition of Farsi handwritten digits," *J. Supercomput.*, vol. 77, no. 4, pp. 3193–3222, Apr. 2021, doi: 10.1007/s11227-020-03388-7.

[14] N. Arica and F. T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting," *IEEE Trans. Syst., Man, C, Appl. Rev.*, vol. 31, no. 2, pp. 216–233, May 2001. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/941845

[15] M. Patel and S. P. Thakkar, "Handwritten character recognition in English: A survey," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, pp. 345–350, Feb. 2015. [Online]. Available: https://ijarcce.com/wp-content/uploads/2015/03/IJARCCE6C.pdf

[16] A. N. Azmi, D. Nasien, and S. M. Shamsuddin, "A review on handwritten character and numeral recognition for Roman, Arabic, Chinese and Indian scripts," *Int. J. Adv. Stud. Comput., Sci. Eng.*, vol. 2, no. 4, pp. 1–8, 2013.

[17] M. Yadav, R. K. Purwar, and M. Mittal, "Handwritten Hindi character recognition: A review," *IET Image Process.*, vol. 12, no. 11, pp. 1919–1933, Nov. 2018, doi: 10.1049/iet-ipr.2017.0184.

[18] M. Kumar, M. K. Jindal, R. K. Sharma, and S. R. Jindal, "Character and numeral recognition for non-indic and indic scripts: A survey," *Artif. Intell. Rev.*, vol. 52, no. 4, pp. 2235–2261, Dec. 2019, doi: 10.1007/s10462-017-9607-x.

[19] D. M. Eberhard, G. F. Simons, and C. D. Fennings. (Jun. 2022). *What are the Top 200 Most Spoken Languages?*. [Online]. Available: https://www.ethnologue.com/guides/ethnologue200

[20] C. Halder and K. Roy, "Individuality of isolated Bangla numerals," *J. Netw. Innov. Comput.*, vol. 1, pp. 33–42, Jan. 2013.

[21] A. Holme, *Geometry: Our Cultural Heritage*. New York, NY, USA: Springer-Verlag, 2010, doi: 10.1007/978-3-642-14441-7.

[22] K. Roy, S. Vajda, U. Pal, and B. Chaudhuri, "A system towards Indian postal automation," in *Proc. 9th Int. Workshop Frontiers Handwriting Recognit.*, Oct. 2004, pp. 580–585. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1363974

[23] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data–AI integration perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1328–1347, Apr. 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8862913

[24] M. Biswas, R. Islam, G. K. Shom, M. Shopon, N. Mohammed, S. Momen, and A. Abedin, "BanglaLekha-isolated: A multi-purpose comprehensive dataset of handwritten Bangla isolated characters," *Data Brief*, vol. 12, pp. 103–107, Jun. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340917301117

[25] S. Alam, T. Reasat, R. M. Doha, and A. I. Humayun, "NumtaDB–assembled Bengali handwritten digits," 2018, *arXiv:1806.02452*.

[26] A. S. A. Rabby, S. Haque, M. S. Islam, S. Abujar, and S. A. Hossain, "Ekush: A multipurpose and multitype comprehensive database for online off-line Bangla handwritten characters," in *Recent Trends in Image Processing and Pattern Recognition*, K. C. Santosh and R. S. Hegadi, Eds. Singapore: Springer Singapore, 2019, pp. 149–158, doi: 10.1007/978-981-13-9187-3_14.

[27] A. M. S. Chowdhury and M. S. Rahman, "Towards optimal convolutional neural network parameters for Bengali handwritten numerals recognition," in *Proc. 19th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2016, pp. 431–436. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7860237

[28] U. Bhattacharya and B. B. Chaudhuri, "Handwritten numeral databases of Indian scripts and multistage recognition of mixed numerals," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 3, pp. 444–457, Mar. 2009. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4492784/

[29] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A statistical–topological feature combination for recognition of handwritten numerals," *Appl. Soft Comput.*, vol. 12, no. 8, pp. 2486–2495, Aug. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494612001378

[30] N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application," *Appl. Soft Comput.*, vol. 12, no. 5, pp. 1592–1606, May 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494611004728

[31] S. A. Kamran, A. I. Humayun, S. Alam, R. M. Doha, M. K. Mandal, T. Reasat, and F. Rahman, "AI learns to recognize Bengali handwritten digits: Bengali.AI computer vision challenge 2018," 2018, *arXiv:1810.04452*.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/726791

[33] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=Bygh9j09KX

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org.

[35] M. A. H. Akhand, M. M. Rahman, P. C. Shill, S. Islam, and M. M. H. Rahman, "Bangla handwritten numeral recognition using convolutional neural network," in *Proc. Int. Conf. Electr. Eng. Inf. Commun. Technol. (ICEEICT)*, May 2015, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7307467

[36] S. M. A. Sharif, N. Mohammed, N. Mansoor, and S. Momen, "A hybrid deep model with HOG features for Bangla handwritten numeral classification," in *Proc. 9th Int. Conf. Electr. Comput. Eng. (ICECE)*, Dec. 2016, pp. 463–466. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7853957

[37] M. Akhand, M. Ahmed, and M. H. Rahman, "Convolutional neural network based handwritten Bengali and Bengali-English mixed numeral recognition," *Int. J. Image, Graph. Signal Process.*, vol. 8, no. 9, p. 40, 2016. [Online]. Available: https://www.mecs-press.org/ijigsp/ijigsp-v8-n9/v8n9-6.html

[38] M. A. H. Akhand, M. Ahmed, and M. M. H. Rahman, "Convolutional neural network training with artificial pattern for Bangla handwritten numeral recognition," in *Proc. 5th Int. Conf. Informat., Electron. Vis. (ICIEV)*, May 2016, pp. 625–630. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7760077

[39] S. M. A. Sharif and M. Mahboob, "Evil method: A deep CNN model for Bangla handwritten numeral classification," in *Proc. 4th Int. Conf. Adv. Electr. Eng. (ICAEE)*, Sep. 2017, pp. 217–222. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8255356

[40] O. Paul, "Image pre-processing on NumtaDB for Bengali handwritten digit recognition," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8554910

[41] A. S. A. Rabby, S. Abujar, S. Haque, and S. A. Hossain, "Bangla handwritten digit recognition using convolutional neural network," in *Emerging Technologies in Data Mining and Information Security*, A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, and S. Dutta, Eds. Singapore: Springer, 2019, pp. 111–122, doi: 10.1007/978-981-13-1951-8_11.

[42] A. Islam, F. Rahman, and A. S. A. Rabby, "Sankhya: An unbiased benchmark for Bangla handwritten digits recognition," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 4676–4683. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9005696

[43] M. F. Wahid, M. F. Shahriar, and M. S. I. Sobuj, "A classical approach to handcrafted feature extraction techniques for Bangla handwritten digit recognition," in *Proc. Int. Conf. Electron., Commun. Inf. Technol. (ICECIT)*, Sep. 2021, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9641406

[44] S. Basu, R. Sarkar, N. Das, M. Kundu, M. Nasipuri, and D. K. Basu, "Handwritten Bangla digit recognition using classifier combination through DS technique," in *Pattern Recognition and Machine Intelligence*, S. K. Pal, S. Bandyopadhyay, and S. Biswas, Eds. Berlin, Germany: Springer, 2005, pp. 236–241, doi: 10.1007/11590316_32.

[45] S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri, and D. K. Basu, "An MLP based approach for recognition of handwritten 'Bangla' numerals," in *Proc. 2nd Indian Int. Conf. Artif. Intell.*, B. Prasad, Ed. Pune, India, Dec. 2005, pp. 407–417.

[46] J.-W. Xu, J. Xu, and Y. Lu, "Handwritten Bangla digit recognition using hierarchical Bayesian network," in *Proc. 3rd Int. Conf. Intell. Syst. Knowl. Eng.*, vol. 1, Nov. 2008, pp. 1096–1099. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4731093

[47] H. A. Khan, A. A. Helal, and K. I. Ahmed, "Handwritten Bangla digit recognition using sparse representation classifier," in *Proc. Int. Conf. Informat., Electron. Vis. (ICIEV)*, May 2014, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6850817

[48] T. Hassan and H. A. Khan, "Handwritten Bangla numeral recognition using local binary pattern," in *Proc. Int. Conf. Electr. Eng. Inf. Commun. Technol. (ICEEICT)*, May 2015, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7307371

[49] A. M. S. Chowdhury and M. S. Rahman, "Towards optimal shallow ANN for recognizing isolated handwritten Bengali numerals," in *Proc. 9th Int. Conf. Electr. Comput. Eng. (ICECE)*, Dec. 2016, pp. 194–197. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7853889

[50] M. Z. Alom, P. Sidike, T. M. Taha, and V. K. Asari, "Handwritten Bangla digit recognition using deep learning," 2017, *arXiv:1705.02680*.

[51] T. I. Aziz, A. S. Rubel, M. S. Salekin, and R. Kushol, "Bangla handwritten numeral character recognition using directional pattern," in *Proc. 20th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2017, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8281820

[52] H. Rehana, "Bangla handwritten digit classification and recognition using SVM algorithm with HOG features," in *Proc. 3rd Int. Conf. Electr. Inf. Commun. Technol. (EICT)*, Dec. 2017, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8275203

[53] A. Choudhury, H. S. Rana, and T. Bhowmik, "Handwritten Bengali numeral recognition using HOG based feature extraction algorithm," in *Proc. 5th Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Feb. 2018, pp. 687–690. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8474215

[54] A. Shawon, M. J.-U. Rahman, F. Mahmud, and M. M. A. Zaman, "Bangla handwritten digit recognition using deep CNN for large and unbiased dataset," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8554900

[55] C. Saha, R. H. Faisal, and M. M. Rahman, "Bangla handwritten digit recognition using an improved deep convolutional neural network architecture," in *Proc. Int. Conf. Electr., Comput. Commun. Eng. (ECCE)*, Feb. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8679309

[56] M. J. Hasan, M. F. Wahid, M. S. Alom, and M. M. A. Mia, "A new state of art deep learning approach for Bangla handwritten digit recognition using SVM classifier," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9225367

[57] M. Fahim Sikder, "Bangla handwritten digit recognition and generation," in *Proc. Int. Joint Conf. Comput. Intell.*, M. S. Uddin and J. C. Bansal, Eds. Singapore: Springer, 2020, pp. 547–556, doi: 10.1007/978-981-13-7564-4_46.

[58] A. Sufian, A. Ghosh, A. Naskar, F. Sultana, J. Sil, and M. M. H. Rahman, "BDNet: Bengali handwritten numeral digit recognition based on densely connected convolutional neural networks," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2610–2620, Jun. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157820303220

[59] P. K. Singh, I. Chatterjee, R. Sarkar, E. B. Smith, and M. Nasipuri, "A new feature extraction approach for script invariant handwritten numeral recognition," *Expert Syst.*, vol. 38, no. 6, p. e12699, 2021, doi: 10.1111/exsy.12699.

[60] Y. Wen, Y. Lu, and P. Shi, "Handwritten Bangla numeral recognition system and its application to postal automation," *Pattern Recognit.*, vol. 40, no. 1, pp. 99–107, Jan. 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320306003141

[61] H. Zunair, N. Mohammed, and S. Momen, "Unconventional wisdom: A new transfer learning approach applied to Bengali numeral classification," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8554435/

[62] R. Noor, K. M. Islam, and M. J. Rahimi, "Handwritten Bangla numeral recognition using ensembling of convolutional neural network," in *Proc. 21st Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8631944

[63] T. Mahmud, A. R. Hossain, and S. A. Fattah, "DeepBanglaNet: A deep convolutional neural network to recognize Bengali handwritten digits," in *Proc. IEEE Region 10th Symp. (TENSYMP)*, 2020, pp. 742–745. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9230922

[64] W.-C. Siu and K.-W. Hung, "Review of image interpolation and super-resolution," in *Proc. Asia Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, Dec. 2012, pp. 1–10. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6411957

[65] A. Majumdar and B. B. Chaudhuri, "A MLP classifier for both printed and handwritten Bangla numeral recognition," in *Computer Vision, Graphics and Image Processing*, P. K. Kalra and S. Peleg, Eds. Berlin, Germany: Springer, 2006, pp. 796–804, doi: 10.1007/11949619_71.

[66] O. Surinta, L. Schomaker, and M. Wiering, "A comparison of feature and pixel-based methods for recognizing handwritten Bangla digits," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Aug. 2013, pp. 165–169. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6628605

[67] M. A. H. Akhand, M. Ahmed, and M. M. H. Rahman, "Multiple convolutional neural network training for Bangla handwritten numeral recognition," in *Proc. Int. Conf. Comput. Commun. Eng. (ICCCE)*, Jul. 2016, pp. 311–315. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7808331

[68] M. A. H. Akhand, M. Ahmed, M. M. H. Rahman, and M. M. Islam, "Convolutional neural network training incorporating rotation-based generated patterns and handwritten numeral recognition of major Indian scripts," *IETE J. Res.*, vol. 64, no. 2, pp. 176–194, Mar. 2018, doi: 10.1080/03772063.2017.1351322.

[69] M. Ahmed, M. Akhand, and M. H. Rahman, "Recognizing Bangla handwritten numeral utilizing deep long short term memory," *Int. J. Image, Graph. Signal Process.*, vol. 11, no. 1, pp. 23–32, 2019. [Online]. Available: https://www.mecs-press.org/ijigsp/ijigsp-v11-n1/v11n1-3.html

[70] F. Kimura, M. Shridhar, and Z. Chen, "Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words," in *Proc. 2nd Int. Conf. Document Anal. Recognit. (ICDAR)*, Oct. 1993, pp. 18–22. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/395791

[71] M. R. Mamun, Z. Al Nazi, and M. S. U. Yusuf, "Bangla handwritten digit recognition approach with an ensemble of deep residual networks," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2018, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8554674

[72] C.-L. Liu and C. Y. Suen, "A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters," *Pattern Recognit.*, vol. 42, no. 12, pp. 3287–3295, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320308004457

[73] M. M. Hasan, M. R. U. Islam, and M. T. Mahmood, "Recognition of Bengali handwritten digits using convolutional neural network architectures," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8554753

[74] S. Haque, A. S. A. Rabby, M. S. Islam, and S. A. Hossain, "ShonkhaNet: A dynamic routing for Bangla handwritten digit recognition using capsule network," in *Recent Trends in Image Processing and Pattern Recognition*, K. C. Santosh and R. S. Hegadi, Eds. Singapore: Springer, 2019, pp. 159–170, doi: 10.1007/978-981-13-9187-3_15.

[75] T. Hashem, M. Asif, and M. A.-A. Bhuiyan, "Handwritten Bangla digit recognition employing hybrid neural network approach," in *Proc. 16th Int. Conf. Comput. Inf. Technol.*, Mar. 2014, pp. 360–365. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6997353

[76] M. M. I. Shovon, M. Kamruzzaman, and M. K. Kundu, "Recognition of handwritten Bangla number using multi layer convolutional neural network," in *Proc. IEEE Region 10th Symp. (TENSYMP)*, Jun. 2020, pp. 783–786. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9230703

[77] K. M. Islam, R. Noor, C. Saha, and J. Rahimi, "A deep convolutional neural network for Bangla handwritten numeral recognition," in *Proc. IEEE Int. WIE Conf. Electr. Comput. Eng. (WIECON-ECE)*, Dec. 2018, pp. 45–50. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8783151

[78] S. N. Shuvo, F. Hasan, M. U. Ahmed, S. A. Hossain, and S. Abujar, "MathNET: Using CNN Bangla handwritten digit, mathematical symbols, and trigonometric function recognition," in *Soft Computing Techniques and Applications*, S. Borah, R. Pradhan, N. Dey, and P. Gupta, Eds. Singapore: Springer, 2021, pp. 515–523, doi: 10.1007/978-981-15-7394-1_47.

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

[80] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Dec. 2002. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10302

[81] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *J. Big Data*, vol. 5, no. 1, pp. 1–30, Dec. 2018, doi: 10.1186/s40537-018-0151-6.

[82] M. Shopon, N. Mohammed, and M. A. Abedin, "Bangla handwritten digit recognition using autoencoder and deep convolutional neural network," in *Proc. Int. Workshop Comput. Intell. (IWCI)*, Dec. 2016, pp. 64–68. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7860340/

[83] M. Abdullah-Al-Wadud, M. H. Kabir, M. A. A. Dewan, and O. Chae, "A dynamic histogram equalization for image contrast enhancement," *IEEE Trans. Consum. Electron.*, vol. 53, no. 2, pp. 593–600, Jul. 2007.

[84] M. H. Kabir, M. Abdullah-Al-Wadud, and O. Chae, "Brightness preserving image contrast enhancement using weighted mixture of global and local transformation functions," *Int. Arab J. Inf. Technol.*, vol. 7, no. 4, pp. 403–410, 2010. [Online]. Available: https://iajit.org/portal/index.php/archive/volume-7-2010/october-2010-no-4/item/1848-brightness-preserving-image-contrast-enhancement-using-weighted-mixture-of-global-and-local-transfor

[85] L. Wang and D.-C. He, "Texture classification using texture spectrum," *Pattern Recognit.*, vol. 23, no. 8, pp. 905–910, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0031320390901358

[86] T. Jabid, M. H. Kabir, and O. Chae, "Robust facial expression recognition based on local directional pattern," *ETRI J.*, vol. 32, no. 5, pp. 784–794, 2010, doi: 10.4218/etrij.10.1510.0132.

[87] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, no. 1, Jun. 2005, pp. 886–893. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1467360

[88] I. Sobel, "An isotropic 3×3 gradient operator," Presentation Stanford A.I. Project, Tech. Rep., 1968.
[89] R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Comput. Biomed. Res.*, vol. 4, no. 3, pp. 315–328, Jun. 1971. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0010480971900346
[90] L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Dept. Elect. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, 1963. [Online]. Available: https://hdl.handle.net/1721.1/11589
[91] T. Jabid, M. H. Kabir, and O. Chae, "Local directional pattern (LDP)—A robust image descriptor for object recognition," in *Proc. 7th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Aug. 2010, pp. 482–487. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5597095
[92] F. Ahmed, "Gradient directional pattern: A robust feature descriptor for facial expression recognition," *Electron. Lett.*, vol. 48, no. 19, pp. 1203–1204, Sep. 2012, doi: 10.1049/el.2012.1841.
[93] T. Jabid, M. H. Kabir, and O. Chae, "Local directional pattern (LDP) for face recognition," in *Dig. Tech. Papers Int. Conf. Consum. Electron. (ICCE)*, Jan. 2010, pp. 329–330. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5418801
[94] N. Kingsbury, "Image processing with complex wavelets," *Phil. Trans. Roy. Soc. London A, Math., Phys. Eng. Sci.*, vol. 357, no. 1760, pp. 2543–2560, 1999.
[95] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Trans. Inf. Theory*, vol. 36, no. 5, pp. 961–1005, Sep. 1990.
[96] R. Mehrotra, K. R. Namuduri, and N. Ranganathan, "Gabor filter-based edge detection," *Pattern Recognit.*, vol. 25, no. 12, pp. 1479–1494, Dec. 1992.
[97] M. Z. Hossain, M. A. Amin, and H. Yan, "Rapid feature extraction for Bangla handwritten digit recognition," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Jul. 2011, pp. 1832–1837. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6017001
[98] P. K. Singh, S. Das, R. Sarkar, and M. Nasipuri, "Recognition of handwritten indic script numerals using Mojette transform," in *Proc. 1st Int. Conf. Intell. Comput. Commun.*, J. K. Mandal, S. C. Satapathy, M. K. Sanyal, and V. Bhateja, Eds. Singapore: Springer, 2017, pp. 459–466, doi: 10.1007/978-981-10-2035-3_47.
[99] S. Ghosh, A. Chatterjee, P. K. Singh, S. Bhowmik, and R. Sarkar, "Language-invariant novel feature descriptors for handwritten numeral recognition," *Vis. Comput.*, vol. 37, no. 7, pp. 1781–1803, Jul. 2021, doi: 10.1007/s00371-020-01938-x.
[100] M. Bulacu and L. Schomaker, "Text-independent writer identification and verification using textural and allographic features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 701–717, Apr. 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4250467
[101] F. Hausdorff, "Summationsmethoden und Momentfolgen. I," *Mathematische Zeitschrift*, vol. 9, nos. 1–2, pp. 74–109, Mar. 1921, doi: 10.1007/BF01378337.
[102] P. K. Singh, R. Sarkar, and M. Nasipuri, "A study of moment based features on handwritten digit recognition," *Appl. Comput. Intell. Soft Comput.*, vol. 2016, pp. 1–17, Mar. 2016. [Online]. Available: https://www.hindawi.com/journals/acisc/2016/2796863/
[103] Y. S. Abu-Mostafa and D. Psaltis, "Recognitive aspects of moment invariants," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 6, pp. 698–706, Nov. 1984. [Online]. Available: https://ieeexplore.ieee.org/document/4767594
[104] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 5, pp. 489–497, May 1990. [Online]. Available: https://ieeexplore.ieee.org/document/55109
[105] P.-T. Yap and R. Paramesran, "An efficient method for the computation of Legendre moments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1996–2002, Dec. 2005. [Online]. Available: https://ieeexplore.ieee.org/document/1524992
[106] R. C. Gonzalez and R. Woods, *Digital Image Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 2008. [Online]. Available: https://imageprocessingplace.com/DIP-3E/dip3e_main_page.htm
[107] M. Petrou and A. Kadyrov, "Affine invariant features from the trace transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 30–44, Jan. 2004. [Online]. Available: https://ieeexplore.ieee.org/document/1261077
[108] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inf. Theory*, vol. 8, no. 2, pp. 179–187, Feb. 1962. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1057692
[109] M. S. Azmi, M. F. Nasrudin, K. Omar, C. W. S. B. C. W. Ahmad, and K. W. M. Ghazali, "Exploiting features from triangle geometry for digit recognition," in *Proc. Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, May 2013, pp. 876–880. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6689658
[110] A. Roy, N. Mazumder, N. Das, R. Sarkar, S. Basu, and M. Nasipuri, "A new quad tree based feature set for recognition of handwritten Bangla numerals," in *Proc. IEEE Int. Conf. Eng. Educ., Innov. Practices Future Trends (AICERA)*, Jul. 2012, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6306727
[111] P. K. Singh, S. Das, R. Sarkar, and M. Nasipuri, "Script invariant handwritten digit recognition using a simple feature descriptor," *Int. J. Comput. Vis. Robot.*, vol. 8, no. 5, pp. 543–560, 2018, doi: 10.1504/IJCVR.2018.095005.
[112] U. Pal and B. B. Chaudhuri, "Automatic recognition of unconstrained off-line Bangla handwritten numerals," in *Advances in Multimodal Interfaces*, T. Tan, Y. Shi, and W. Gao, Eds. Berlin, Germany: Springer, 2000, pp. 371–378, doi: 10.1007/3-540-40063-X_49.
[113] U. Pal, B. B. Chaudhuri, and A. Belaid, "A complete system for Bangla handwritten numeral recognition," *IETE J. Res.*, vol. 52, no. 1, pp. 27–34, Jan. 2006, doi: 10.1080/03772063.2006.11416437.
[114] S. Ahmed, M. R. Islam, and M. S. Azam, "Bangla hand written digit recognition using supervised locally linear embedding algorithm and support vector machine," in *Proc. 12th Int. Conf. Comput. Inf. Technol.*, Dec. 2009, pp. 390–393. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5407269
[115] D. Keysers, T. Deselaers, C. Gollan, and H. Ney, "Deformation models for image recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1422–1435, Aug. 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4250467
[116] H. Cecotti, "Handwritten digit recognition of Indian scripts: A cascade of distances approach," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/7280451
[117] M. M. Hoque, M. M. Islam, and M. M. Ali, "An efficient fuzzy method for Bangla handwritten numerals recognition," in *Proc. Int. Conf. Electr. Comput. Eng.*, Dec. 2006, pp. 197–200. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4178442
[118] U. Pal, R. K. Roy, K. Roy, and F. Kimura, "Indian multi-script full pin-code string recognition for postal automation," in *Proc. 10th Int. Conf. Document Anal. Recognit.*, 2009, pp. 290–295. http://www.cenparmi.concordia.ca/ICFHR2008/Proceedings/papers/cr1076.pdf
[119] Y. Wen and L. He, "A classifier for Bangla handwritten numeral recognition," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 948–953, Jan. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417411010542
[120] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philos. Mag.*, vol. 2, no. 6, pp. 559–572, 1901, doi: 10.1080/14786440109462720.
[121] D. de Ridder, O. Kouropteva, O. Okun, M. Pietikäinen, and R. P. W. Duin, "Supervised locally linear embedding," in *Artificial Neural Networks and Neural Information Processing*, O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, Eds. Berlin, Germany: Springer, 2003, pp. 333–341, doi: 10.1007/3-540-44989-2_40.
[122] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 1, pp. 49–60, Jan./Feb. 1988. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/87054/
[123] R. Guha, M. Ghosh, P. K. Singh, R. Sarkar, and M. Nasipuri, "M-HMOGA: A new multi-objective feature selection algorithm for handwritten numeral classification," *J. Intell. Syst.*, vol. 29, no. 1, pp. 1453–1467, Jun. 2019, doi: 10.1515/jisys-2019-0064.
[124] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Apr. 1995, doi: 10.1007/BF00994018.
[125] K.-B. Duan and S. S. Keerthi, "Which is the best multiclass SVM method? An empirical study," in *Multiple Classifier Systems*, N. C. Oza, R. Polikar, J. Kittler, and F. Roli, Eds. Berlin, Germany: Springer, 2005, pp. 278–285, doi: 10.1007/11494683_28.

[126] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2001. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/991427

[127] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958. [Online]. Available: https://psycnet.apa.org/record/1959-09865-001

[128] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943, doi: 10.1007/BF02478259.

[129] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ. San Diego La Jolla Inst. Cognitive Sci., CA, Tech. Rep. ADA164453, 1985. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA164453

[130] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, no. 35, pp. 1–17, 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5721660/

[131] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," in *Classic Works Dempster-Shafer Theory Belief Functions*, R. R. Yager and L. Liu, Eds. Berlin, Germany: Springer, 2008, pp. 57–72, doi: 10.1007/978-3-540-44792-4_3.

[132] J. Pearl, "Bayesian networks: A model of self-activated memory for evidential reasoning," in *Proc. 7th Conf. Cognit. Sci. Soc.*, 1985, pp. 329–334. [Online]. Available: https://ftp.cs.ucla.edu/pub/stat_ser/r43-1985.pdf

[133] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artif. Intell.*, vol. 42, nos. 2–3, pp. 393–405, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/000437029090060D

[134] E. Fix and J. L. Hodges, "Discriminatory analysis. Nonparametric discrimination: Consistency properties," *Int. Stat. Rev.*, vol. 57, no. 3, pp. 238–247, 1989. [Online]. Available: https://www.jstor.org/stable/1403797

[135] M. N. Hoq, M. M. Islam, N. A. Nipa, and M. M. Akbar, "A comparative overview of classification algorithm for Bangla handwritten digit recognition," in *Proc. Int. Joint Conf. Comput. Intell.*, M. S. Uddin and J. C. Bansal, Eds. Singapore: Springer, 2020, pp. 265–277, doi: 10.1007/978-981-13-7564-4_24.

[136] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4483511

[137] E. B. Hunt, J. Marin, and P. J. Stone, *Experiments in Induction*. New York, NY, USA: Academic, 1966. [Online]. Available: https://books.google.com.bd/books?id=C9UKAAAAMAAJ

[138] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, Aug. 1995, pp. 278–282. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/598994

[139] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Microsoft, Washington, DC, USA, Tech. Rep. MSR-TR-98–14, Apr. 1998. [Online]. Available: https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/

[140] P.-F. Verhulst, "Notice sur la loi que la population suit dans son accroissement," *Correspondance Mathématique et Phys.*, vol. 10, pp. 113–126, 1838.

[141] P. Nemenyi, "Distribution-free multiple comparisons," M.S. thesis, Dept. Math., Princeton Univ., Princeton, NJ, USA, 1963. [Online]. Available: https://www.proquest.com/openview/c1f3e8829e8351e9c2a1c5e51778c6cf/1?pq-origsite=gscholar&cbl=18750&diss=y

[142] C. W. Dunnett, "A multiple comparison procedure for comparing several treatments with a control," *J. Amer. Stat. Assoc.*, vol. 50, pp. 1096–1121, Dec. 1955, doi: 10.1080/01621459.1955.10501294.

[143] O. J. Dunn, "Multiple comparisons among means," *J. Amer. Statist. Assoc.*, vol. 56, no. 293, pp. 52–64, Mar. 1961, doi: 10.1080/01621459.1961.10482090.

[144] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C Van Esesn, A. A S. Awwal, and V. K. Asari, "The history began from AlexNet: A comprehensive survey on deep learning approaches," 2018, *arXiv:1803.01164*.

[145] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8099726

[146] M. F. Aslan, A. Durdu, K. Sabanci, and M. A. Mutluer, "CNN and HOG based comparison study for complete occlusion handling in human tracking," *Measurement*, vol. 158, Jul. 2020, Art. no. 107704. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224120302426

[147] M. Shopon, N. Mohammed, and M. A. Abedin, "Image augmentation by blocky artifact in deep convolutional neural network for handwritten digit recognition," in *Proc. IEEE Int. Conf. Imag., Vis. Pattern Recognit. (icIVPR)*, Feb. 2017, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7890867

[148] M. Ahmed, A. K. Paul, and M. A. H. Akhand, "Stacked auto encoder training incorporating printed text data for handwritten Bangla numeral recognition," in *Proc. 19th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2016, pp. 437–442. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7860238

[149] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and D. andL Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6795724

[150] S. Razik, E. Hossain, S. Ismail, and M. S. Islam, "SUST-BHND: A database of Bangla handwritten numerals," in *Proc. IEEE Int. Conf. Imag., Vis. Pattern Recognit. (icIVPR)*, Feb. 2017, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7890891

[151] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham, Switzerland: Springer, 2018, pp. 270–279, doi: 10.1007/978-3-030-01424-7_27.

[152] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1106–1114. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[153] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds. San Diego, CA, USA, May 2015, pp. 1–14.

[154] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/7298594

[155] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778. [Online]. Available: https://ieeexplore.ieee.org/document/7780459

[156] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8099678/

[157] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015. [Online]. Available: https://image-net.org/challenges/LSVRC/

[158] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[159] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 3856–3866. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/2cad8fa47bbef282badbb8de5374b894-Abstract.html

[160] R. Pramanik, P. Dansena, and S. Bag, "A study on the effect of CNN-based transfer learning on handwritten indic and mixed numeral recognition," in *Document Analysis and Recognition*, S. Sundaram and G. Harit, Eds. Singapore: Springer, 2019, pp. 41–51, doi: 10.1007/978-981-13-9361-7_4.

[161] T. Ghosh, M.-H.-Z. Abedin, H. Al Banna, N. Mumenin, and M. A. Yousuf, "Performance analysis of state of the art convolutional neural network architectures in Bangla handwritten character recognition," *Pattern Recognit. Image Anal.*, vol. 31, no. 1, pp. 60–71, Jan. 2021, doi: 10.1134/S1054661821010089.

[162] R. Basri, M. R. Haque, M. Akter, and M. S. Uddin, "Bangla handwritten digit recognition using deep convolutional neural network," in *Proc. Int. Conf. Comput. Advancements*, New York, NY, USA, 2020, pp. 339–346, doi: 10.1145/3377049.3377077.

[163] M. M. Rahman, M. S. Islam, R. Sassi, and M. Aktaruzzaman, "Convolutional neural networks performance comparison for handwritten Bengali numerals recognition," *Social Netw. Appl. Sci.*, vol. 1, no. 12, pp. 1–11, Dec. 2019, doi: 10.1007/s42452-019-1682-y.

[164] L. Liao, H. Li, W. Shang, and L. Ma, "An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 3, pp. 1–40, Jul. 2022, doi: 10.1145/3506695.

[165] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8016501/

[166] M. A. Al Nasim, R. E. Ferdous, M. A. H. Pantho, and A. I. Chowdhury, "A comparative analysis on Bangla handwritten digit recognition with data augmentation and non-augmentation process," in *Proc. Int. Congr. Hum.-Comput. Interact., Optim. Robotic Appl. (HORA)*, Jun. 2020, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9152905

[167] J. Mukhoti, S. Dutta, and R. Sarkar, "Handwritten digit classification in Bangla and Hindi using deep learning," *Appl. Artif. Intell.*, vol. 34, no. 14, pp. 1074–1099, Dec. 2020, doi: 10.1080/08839514.2020.1804228.

[168] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, F. Bach and D. Blei, Eds. Lille, France, Jul. 2015, pp. 448–456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html

[169] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 29, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/hash/c8067ad1937f728f51288b3eb986afaa-Abstract.html

[170] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[171] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: An empirical study of their impact to deep learning," *Multimedia Tools Appl.*, vol. 79, nos. 19–20, pp. 12777–12815, May 2020, doi: 10.1007/s11042-019-08453-9.

[172] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2015, pp. 1–15.

[173] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *Proc. Chin. Control And Decis. Conf. (CCDC)*, Jun. 2018, pp. 1836–1841. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8407425/

[174] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, Madison, WI, USA, 2010, pp. 807–814.

[175] A. K. Dubey and V. Jain, "Comparative study of convolution neural network's ReLU and leaky-ReLU activation functions," in *Applications of Computing, Automation and Wireless Systems in Electrical Engineering*, S. Mishra, Y. R. Sood, and A. Tomar, Eds. Singapore: Springer, 2019, pp. 873–880, doi: 10.1007/978-981-13-6772-4_76.

[176] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang, "Demystifying learning rate policies for high accuracy training of deep neural networks," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 1971–1980. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9006104

[177] L. Wen, L. Gao, X. Li, and B. Zeng, "Convolutional neural network with automatic learning rate scheduler for fault classification," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9312168

[178] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Exp.*, vol. 6, no. 4, pp. 312–315, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959519303455

[179] Y. Bai, E. Yang, B. Han, Y. Yang, J. Li, Y. Mao, G. Niu, and T. Liu, "Understanding and improving early stopping for learning with noisy labels," in *Advances in Neural Information Processing Systems*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Red Hook, NY, USA: Curran Associates, 2021, pp. 24392–24403. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html

[180] S. Hye, M. Rahat-Uz-Zaman, and M. A. H. Akhand, "Extraction of sequence from Bangla handwritten numerals and recognition using LSTM," in *Proc. IEEE Region 10th Symp. (TENSYMP)*, Jun. 2020, pp. 1261–1264. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9230921

[181] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Mach. Learn.*, vol. 28, no. 3, S. Dasgupta and D. McAllester, Eds. Atlanta, Georgia, USA, Jun. 2013, pp. 1310–1318. [Online]. Available: https://proceedings.mlr.press/v28/pascanu13.html

[182] J. S. Bayer, "Learning sequence representations," Ph.D. dissertation, Fakultät für Informatik, Technische Universität München, Germany, 2015. [Online]. Available: https://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20151102-1256381-1-

[183] L. Sun, T. Su, C. Liu, and R. Wang, "Deep LSTM networks for online Chinese handwriting recognition," in *Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Oct. 2016, pp. 271–276. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7814075/

[184] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen, "Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 639–645. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8270041

[185] M. Ahmed, M. A. H. Akhand, and M. M. H. Rahman, "Handwritten Bangla numeral recognition using deep long short term memory," in *Proc. 6th Int. Conf. Inf. Commun. Technol. Muslim World (ICTM)*, Nov. 2016, pp. 310–315. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7814922

[186] Y. Bengio, *Practical Recommendations for Gradient-Based Training Deep Architectures*. Berlin, Germany: Springer, 2012, pp. 437–478, ch. 19, doi: 10.1007/978-3-642-35289-8_26.

[187] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=H1oyRlYgg

[188] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," 2018, *arXiv:1804.07612*.

[189] F. Hasan, S. N. Shuvo, S. Abujar, and S. A. Hossain, "Bangla handwritten math recognition and simplification using convolutional neural network," in *Emerging Technologies in Data Mining and Information Security*, A. E. Hassanien, S. Bhattacharyya, S. Chakrabati, A. Bhattacharya, and S. Dutta, Eds. Singapore: Springer, 2021, pp. 133–141, doi: 10.1007/978-981-33-4367-2_14.

[190] D. Zhelezniakov, V. Zaytsev, and O. Radyvonenko, "Online handwritten mathematical expression recognition and applications: A survey," *IEEE Access*, vol. 9, pp. 38352–38373, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9367185/

[191] C. Chan, "Stroke extraction for offline handwritten mathematical expression recognition," *IEEE Access*, vol. 8, pp. 61565–61575, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9051736

[192] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "ICFHR2016 CROHME: Competition on recognition of online handwritten mathematical expressions," in *Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Oct. 2016, pp. 607–612. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7814132

[193] B. Chang, Q. Zhang, S. Pan, and L. Meng, "Generating handwritten Chinese characters using CycleGAN," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 199–207. [Online]. Available: https://ieeexplore.ieee.org/document/8354132

[194] J. Liu, C. Gu, J. Wang, G. Youn, and J.-U. Kim, "Multi-scale multi-class conditional generative adversarial network for handwritten character generation," *J. Supercomput.*, vol. 75, no. 4, pp. 1922–1940, Apr. 2019, doi: 10.1007/s11227-017-2218-0.

[195] E. Alonso, B. Moysset, and R. Messina, "Adversarial generation of handwritten text images conditioned on sequences," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 481–486. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8977950/

[196] A. Odena, "Semi-supervised learning with generative adversarial networks," 2016, *arXiv:1606.01583*.

[197] S. Haque, S. A. Shahinoor, A. S. A. Rabby, S. Abujar, and S. A. Hossain, "OnkoGan: Bangla handwritten digit generation with deep convolutional generative adversarial networks," in *Recent Trends in Image Processing and Pattern Recognition*, K. C. Santosh and R. S. Hegadi, Eds. Singapore: Springer, 2019, pp. 108–117, doi: 10.1007/978-981-13-9187-3_10.

[198] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[199] S. Tajrean and M. A. Yousuf, "Handwritten Bengali number detection using region proposal network," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9084035

[200] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[201] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2016, pp. 2818–2826. [Online]. Available: https://ieeexplore.ieee.org/document/7780677

[202] M. R. Haque, M. G. Azam, S. M. Milon, M. S. Hossain, M. A.-A. Molla, and M. S. Uddin, "Quantitative analysis of deep CNNs for multilingual handwritten digit recognition," in *Proc. Int. Conf. Trends Comput. Cognit. Eng.*, M. S. Kaiser, A. Bandyopadhyay, M. Mahmud, and K. Ray, Eds. Singapore: Springer, 2021, pp. 15–25, doi: 10.1007/978-981-33-4673-4_2.

[203] D. Gupta and S. Bag, "CNN-based multilingual handwritten numeral recognition: A fusion-free approach," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113784. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420306084

[204] A. Fateh, M. Fateh, and V. Abolghasemi, "Multilingual handwritten numeral recognition using a robust deep network joint with transfer learning," *Inf. Sci.*, vol. 581, pp. 479–494, Dec. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521009828

[205] P. K. Singh, R. Sarkar, N. Das, S. Basu, M. Kundu, and M. Nasipuri, "Benchmark databases of handwritten bangla-roman and devanagari-roman mixed-script document images," *Multimedia Tools Appl.*, vol. 77, no. 7, pp. 8441–8473, Apr. 2018, doi: 10.1007/s11042-017-4745-3.

[206] A. F. De Sousa Neto, B. L. D. Bezerra, E. B. Lima, and A. H. Toselli, "HDSR-flor: A robust end-to-end system to solve the handwritten digit string recognition problem in real complex scenarios," *IEEE Access*, vol. 8, pp. 208543–208553, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9262842

[207] A. G. Hochuli, A. S. Britto, D. A. Saji, J. M. Saavedra, R. Sabourin, and L. S. Oliveira, "A comprehensive comparison of end-to-end approaches for handwritten digit string recognition," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 114196. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420309271

[208] S. Aly and A. Mohamed, "Unknown-length handwritten numeral string recognition using cascade of PCA-SVMNet classifiers," *IEEE Access*, vol. 7, pp. 52024–52034, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8693718

[209] H. Kusetogullari, A. Yavariabdi, J. Hall, and N. Lavesson, "DIGITNET: A deep handwritten digit detection and recognition method using a new historical handwritten digit dataset," *Big Data Res.*, vol. 23, Feb. 2021, Art. no. 100182. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214579620300502

[210] S. A. Sabab and M. H. Ashmafee, "Blind reader: An intelligent assistant for blind," in *Proc. 19th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2016, pp. 229–234. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7860200

[211] P. Singh, R. Sarkar, and M. Nasipuri, "Significance of non-parametric statistical tests for comparison of classifiers over multiple datasets," *Int. J. Comput. Sci. Math.*, vol. 7, no. 5, pp. 410–442, 2016. [Online]. Available: https://www.inderscience.com/info/inarticle.php?artid=80073

[212] P. K. Singh, R. Sarkar, and M. Nasipuri, "Statistical validation of multiple classifiers over multiple datasets in the field of pattern recognition," *Int. J. Appl. Pattern Recognit.*, vol. 2, no. 1, pp. 1–23, 2015, doi: 10.1504/IJAPR.2015.068929.

[213] S. Sen, A. Bhattacharyya, P. K. Singh, R. Sarkar, K. Roy, and D. Doermann, "Application of structural and topological features to recognize online handwritten Bangla characters," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 17, no. 3, pp. 1–16, May 2018, doi: 10.1145/3178457.

[214] H. Ren, W. Wang, and C. Liu, "Recognizing online handwritten Chinese characters using RNNs with new computing architectures," *Pattern Recognit.*, vol. 93, pp. 179–192, Sep. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132031930161X

[215] R. Ghosh and P. P. Roy, "A novel feature extraction approach for online Bengali and Devanagari character recognition," in *Proc. 2nd Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Feb. 2015, pp. 483–488. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7095313

[216] W. Samek and K.-R. Müller, *Towards Explainable Artificial Intelligence*. Cham, Switzerland: Springer, 2019, pp. 5–22, doi: 10.1007/978-3-030-28954-6_1.

[217] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G. Yang, "XAI—Explainable artificial intelligence," *Sci. Robot.*, vol. 4, no. 37, Dec. 2019, Art. no. eaay7120, doi: 10.1126/scirobotics.aay7120.

[218] F. Alizadeh, G. Stevens, and M. Esau, "I don't know, is AI also used in airbags?: An empirical study of folk concepts and people's expectations of current and future artificial intelligence," *I-Com*, vol. 20, no. 1, pp. 3–17, Apr. 2021, doi: 10.1515/icom-2021-0009.

[219] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, May 2021, doi: 10.1145/3386252.

[220] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, "A baseline for few-shot image classification," 2019, *arXiv:1909.02729*.

[221] M. A. Souibgui, A. Fornés, Y. Kessentini, and B. Megyesi, "Few shots are all you need: A progressive few shot learning approach for low resource handwritten text recognition," 2021, *arXiv:2107.10064*.

A. B. M. ASHIKUR RAHMAN received the B.Sc. and M.Sc. degrees in computer science and engineering (CSE) from the Islamic University of Technology (IUT), in 2014 and 2018, respectively.

He is currently working as an Assistant Professor with the Department of Computer Science and Engineering, IUT. His research interest includes the application of computer vision techniques in medical image analysis and human biometrics.

MD. BAKHTIAR HASAN received the B.Sc.Engg. and M.Sc.Engg. degrees in computer science and engineering (CSE) from the Islamic University of Technology (IUT), in 2018 and 2022, respectively.

Since 2019, he has been working as a Lecturer with the Department of Computer Science and Engineering, IUT. His research interest includes the use of deep learning and computer vision techniques in human biometrics and smart agriculture.

Mr. Hasan received the Huawei Seeds for the Future Scholarship in 2018. He was awarded the IUT Gold Medal in recognition of his outstanding performance in the pursuit of the B.Sc.Engg. in CSE degree in 2018.

SABBIR AHMED was born in Dhaka, Bangladesh, in 1996. He received the B.Sc.Engg. degree in computer science (CS) from the Islamic University of Technology (IUT), Gazipur, Bangladesh, in 2017, where he is currently pursuing the M.Sc. degree in CS.

Since 2018, he has been working as a Lecturer with the Department of Computer Science and Engineering, IUT. His research interests include pattern recognition, deep learning in computer vision, and intelligent agriculture.

TASNIM AHMED was born in Kushtia, Bangladesh, in 1997. He received the B.Sc. degree in computer science and engineering from the Islamic University of Technology, Gazipur, Bangladesh, where he is currently pursuing the M.Sc. degree.

Since 2020, he has been working as a full-time Lecturer with the Computer Science and Engineering Department, Islamic University of Technology. His research interests include computer vision, natural language processing, bioinformatics, and software engineering.

MD. HAMJAJUL ASHMAFEE received the B.Sc.Engg. and M.Sc.Engg. degrees in computer science and engineering (CSE) from the Islamic University of Technology (IUT), in 2015 and 2022, respectively.

Since 2016, he has been working as a full-time Lecturer with the Computer Science and Engineering Department, IUT. His research interests include data analytics and machine learning.

MOHAMMAD RIDWAN KABIR received the B.Sc.Engg. and M.Sc.Engg. degrees in computer science and engineering from the Islamic University of Technology (IUT), Boardbazar, Gazipur, Bangladesh, in 2017 and 2022, respectively.

Since 2018, he has been working as a Lecturer with the Department of Computer Science and Engineering, IUT. He worked as a Lecturer with the Department of Computer Science Engineering, BRAC University, Dhaka, Bangladesh. His research interests include human–computer interaction, computer vision, assistive technology, machine learning, data analysis and visualization, embedded system development, and wearable devices.

Mr. Kabir and his team received the Runners Up Title (Project Showcasing) at the National ICT Fest in 2016, the Champions title (Project Showcasing) at Esonance in 2017, the Top 5 Innovative Projects Awards at BASIS Soft Expo, Bangladesh, in 2020, and the BASIS National ICT Awards, Bangladesh, in 2021. Furthermore, he has received the Winners Title (Research and Development) in the prestigious APICTA Awards 2020–2021 (host country: Malaysia) from Bangladesh.

MD. HASANUL KABIR (Member, IEEE) received the B.Sc. degree in computer science and information technology from the Islamic University of Technology, Bangladesh, and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea.

He is currently a Professor with the Department of Computer Science and Engineering, Islamic University of Technology. His research interests include feature extraction, visual question answering, and sign language interpretation by combining image processing, machine learning, and computer vision.