Ritam Guha, Manosij Ghosh, Pawan Kumar Singh*, Ram Sarkar and Mita Nasipuri

M-HMOGA: A New Multi-Objective Feature Selection Algorithm for Handwritten Numeral Classification

https://doi.org/10.1515/jisys-2019-0064
Received March 7, 2019; previously published online June 14, 2019.

Abstract: The feature selection process is very important in the field of pattern recognition, which selects the informative features so as to reduce the curse of dimensionality, thus improving the overall classification accuracy. In this paper, a new feature selection approach named Memory-Based Histogram-Oriented Multi-objective Genetic Algorithm (M-HMOGA) is introduced to identify the informative feature subset to be used for a pattern classification problem. The proposed M-HMOGA approach is applied to two recently used feature sets, namely Mojette transform and *Regional Weighted Run Length* features. The experimentations are carried out on *Bangla*, *Devanagari*, and *Roman* numeral datasets, which are the three most popular scripts used in the Indian subcontinent. In-house *Bangla* and *Devanagari* script datasets and Competition on Handwritten Digit Recognition (HDRC) 2013 *Roman* numeral dataset are used for evaluating our model. Moreover, as proof of robustness, we have applied an innovative approach of using different datasets for training and testing. We have used in-house *Bangla* and *Devanagari* script datasets for training the model, and the trained model is then tested on Indian Statistical Institute numeral datasets. For *Roman* numerals, we have used the HDRC 2013 dataset for training and the Modified National Institute of Standards and Technology dataset for testing. Comparison of the results obtained by the proposed model with existing HMOGA and MOGA techniques clearly indicates the superiority of M-HMOGA over both of its ancestors. Moreover, use of K-nearest neighbor as well as multi-layer perceptron as classifiers speaks for the classifier-independent nature of M-HMOGA. The proposed M-HMOGA model uses only about 45–50% of the total feature set in order to achieve around 1% increase when the same datasets are partitioned for training-testing and a 2–3% increase in the classification ability while using only 35–45% features when different datasets are used for training-testing with respect to the situation when all the features are used for classification.

Keywords: M-HMOGA, handwritten numeral classification, feature selection, genetic algorithm, *Bangla* numerals, *Devanagari* numerals, *Roman* numerals.

1 Introduction

A handwritten numeral classification system can be defined as the ability of a computer to receive and interpret handwritten numerals from sources such as paper documents, photographs, touch screens, and other devices without human intervention [14]. India is a multi-script country having 12 official scripts, where *Devanagari* and *Bangla* stand among the top two renowned scripts (in terms of the number of speakers) used in the Indian subcontinent. *Bangla* script is used to write languages like *Bengali*, *Assamese*, etc., whereas *Devanagari* script is used to write languages such as *Nepali*, *Pali*, *Hindi*, *Marathi*, *Konkani*, *Sanskrit*, etc. Furthermore, *Devanagari* script is one of the oldest and most widely used Indian scripts since ancient times and is used by around 500 million people [30]. *Bangla* and *Devanagari*, being the base of many Indian languages, should be given special attention so that document retrieval and analysis of ancient and modern

***Corresponding author: Pawan Kumar Singh,** Department of Computer Science and Engineering, Jadavpur University, 188, Raja S.C. Mullick Road, Kolkata-700032, West Bengal, India, e-mail: pawansingh.ju@gmail.com
Ritam Guha, Manosij Ghosh, Ram Sarkar and Mita Nasipuri: Department of Computer Science and Engineering, Jadavpur University, 188, Raja S.C. Mullick Road, Kolkata-700032, West Bengal, India

Indian literature can be done effectively. In order to attain higher precision values for handwritten numeral recognition for *Bangla* and *Devanagari* scripts, there is a need to design effective techniques. *Roman* script, derived from *Latin* script, is used to write a number of languages such as *English*, *German*, *Spanish*, *Italian*, etc. *English* language, having around 341 million speakers, is the fourth most spoken language in the world. *English* language is used as a binding language in the Indian subcontinent because of its colonial past. Moreover, India stands second among all the countries in terms of the number of *English*-speaking population [33]. In such a multi-script environment, the main aim is to classify numerals written in any Indian script. This makes the problem of handwritten numeral classification even more interesting for the research community.

Classification of numerals written in different Indian scripts is always a complicated task due to the variation of size, shape, thickness, and style of writing of individuals. Many methods have been developed so far to solve the problem of handwritten numeral classification. However, many a time, the designed feature vector may have a large feature dimension among which some are redundant and/or non-informative. Here arises the need for feature selection (FS). The very purpose of FS is to identify the relevant and important features only, which in turn, increases the classification accuracy and also speeds up the classification process. However, it is also sometimes considered an optimization problem. Thus, researchers generally apply some meta-heuristic algorithms to solve the problem within an acceptable time limit.

One very useful and basic algorithm in this field is genetic algorithm (GA). An FS methodology called Memory-Based Histogram-Oriented Multi-objective GA (M-HMOGA) is introduced in the present work, where both multi-layer perceptron (MLP) and K-nearest neighbor (KNN)-based classifiers are used for classification. The main reason for forming this histogram is to make a trade-off between the randomness and exploration properties of any wrapper method. To be more specific, due to the random nature of GA, an important feature might be ignored by this algorithm. To achieve maximum exploration, GA should consider different combinations of features. Keeping in mind the importance of both issues, in this method, a histogram is formed considering the frequency of appearances of various features at different generations. The peaks in the histogram echo the features that appear in almost every generation proving their worth in the classification process. Then, a suitable threshold is calculated from the histogram to select the most optimal feature subset. In HMOGA, every feature present in a candidate solution has an associated value that is equal to the classification accuracy of the candidate solution. After getting the final population, weight is assigned to each feature by adding the values of corresponding features for all the candidate solutions present in the population and the threshold cutoff is calculated by taking the mean of the feature weights. Instead of using the mean, M-HMOGA utilizes a different cutoff value that is found to be more suitable for a handwritten numeral classification problem.

The remainder of the paper is structured as follows: Section 2 reviews the literature. In Section 3, feature extraction methodologies are described, whereas the implementation details regarding M-HMOGA are elaborated in Section 4. Section 5 discusses the dataset descriptions used for the evaluation of the proposed M-HMOGA methodology, whereas the experimental results are reported in Section 6. Finally, Section 7 concludes the work.

2 Literature Study

There are a good number of published methods used for the classification of handwritten numerals, and a number of reviews also exist where different feature sets and classification algorithms are explained in detail. In this section, we would describe some existing works on handwritten *Bangla*, *Devanagari*, and *Roman* numerals.

Khan et al. [28] presented a framework for *Bangla* digit recognition using a sparse representation classifier (SRC). They used the zone density feature extraction method for extracting only local information from differently sized zoning, and zone density features were calculated as the normalized number of foreground pixels in each zone. They finally employed SRC for digit classification and achieved an overall accuracy of 94% using 8×8 zoning on the *CMATERdb* 3.1.1 database [9]. Hashem et al. [22] proposed a local binary pattern (LBP)-based spectral texture descriptor for handwritten *Bangla* digits. Firstly, LBP image was computed from

all pixels of a particular digit, which was again split into a number of zones or blocks. The local histogram of each block was then calculated separately for feature extraction purpose. The technique achieved the highest accuracy of 96.7% for 8×8 blocks using the KNN classifier on the *CMATERdb* 3.1.1 database [9]. Alom et al. [2] used a deep learning-based neural network. They proposed a methodology for recognizing *Bangla* handwritten digits, which were constructed on several filter techniques including Deep Belief Network, Convolutional Neural Networks (CNN), CNN with dropout, CNN with dropout and Gaussian filters, and CNN with Gabor filters and dropout. Sarkhel et al. [38] proposed a cost-effective methodology for handwritten character and numeral recognition structure. A multi-objective region sampling technique was developed for the recognition of handwritten *Bangla* characters and digits in their work. A non-dominated sorting harmony search algorithm (NSHA)-based region sampling methodology and a non-dominated sorting GA (NSGA-II)-based region sampling methodology were developed. An axiomatic fuzzy set methodology based on fuzzy logic was applied to build a model for combining the Pareto optimal outcomes from two multi-objective heuristic algorithms.

Several works are also found in the literature for *Devanagari* handwritten numeral recognition. In Ref. [7], a technique of applying support vector machine (SVM) (having radial basis function kernel) for handwritten *Devanagari* numeral recognition was proposed. The features were extracted using principal component analysis. In Ref. [34], a simple profile and contour base triangular area representation technique for finding feature extraction was proposed. The classification was done using a majority voting scheme on back-propagation and cascade feed-forward neural networks. The performance of this technique was tested on 5030 handwritten numerals, and an accuracy of 94.16% was achieved. Arora et al. [4] proposed a system for recognizing handwritten *Devanagari* numerals using SVM and artificial neural network. The feature extraction was done using some structural information of the numerals, such as shadow-based features, zone-based directional features, zone-based centroid features, and view-based features. In the first stage, numerals were classified using the MLP classifier. Unrecognized numerals of the first stage were then classified in the second stage by SVM using the one-against-all technique, and the system achieved nearly 93.15% recognition rate. A set of 17 geometric features based on pixel, lines, holes, connectivity, line directions, eccentricity, area, perimeter, etc., was used for identifying numerals written in *Devanagari* script by Dongre and Mankar [12]. For classification purpose, they applied five discriminant functions, namely Quadratic, Linear, Diaglinear, Diagquadratic, and Mahalanobis distance, where they found that Linear, Quadratic, and Mahalanobis discriminant functions gave better accuracy than others. They executed a classifier ensemble based on majority voting to obtain a result of 81.67% accuracy using the outcomes of the three discriminant functions. Akhand et al. [1] used two CNN-based models to perform recognition of *Bangla* and *Devanagari* handwritten numerals. The obtained results were found to be better than that of some prominent existing methods. A novel hybrid deep learning model was proposed by Trivedi et al. [44]. The hybrid model uses GA and Limited-memory Broyden–Fletcher–Goldfarb–Shanno optimization algorithms to train the CNN model. The overall model was tested on *Devanagari* handwritten numeral datasets. From the results, it was evident that GA-assisted CNN outperformed non-GA-assisted CNN.

Compared to *Bangla* and *Devanagari* numerals, relatively a lot of research works have been reported for *Roman* numeral recognition. Kessab et al. [13] presented a comparison of square and triangular zoning methods for isolated handwritten *Roman* numeral recognition. Each numeral was transformed into a vector of 4, 6, and 9 components for square zoning and to a vector of 4, 6, and 8 components for triangular zoning. It can be observed from the experimental analysis that the zoning with 9 squares performed better than having 8 triangular zones. This work achieved approximately 85% success rate using the SVM classifier. Salouan et al. [36] used a set of three feature extraction methods: the zoning method combined firstly with the Krawtchouk moments, secondly with pseudo-Zernike invariant moments, and thirdly with the invariant analytical Fourier-Mellin transform. The classification was done using three different classifiers: SVM, naïve Bayes, and dynamic programming. The most precise classifier was SVM, then naïve Bayes, followed by dynamic programming. The authors in Ref. [37] compared four hybrid methods, as follows: (i) zoning technique combined with Radon transform, (ii) zoning technique combined with Hough transform, (iii) zoning technique combined with Gabor filter, and (iv) zoning technique combined with all three descriptors. Finally, in order to recognize each numeral image, they employed three classifiers, which are the MLP, Hidden Markov

Model (HMM), and the hybrid of MLP and HMM. Kulkarni and Vasambekar [29] proposed an efficient automatic recognition system for isolated handwritten *Roman* numerals. Fourier descriptor-based features were extracted and input to a feed-forward back-propagation neural network for classification. The numeral recognition was finally done by a template matching classifier based on a correlation metric on a dataset of 10,000 samples.

A number of studies [3, 8, 10, 26, 35, 39, 41] have already been done for solving the problem of handwritten numeral recognition using the FS procedure. However, some of the methods depend on some statistical FS methods such as principal component analysis, whereas most of them depend on directly applying some well-known optimization algorithms such as GA, particle swarm optimization, ant colony optimization, harmony search algorithm, etc. Most of these studies have been applied on numerals written in a single script, which is a mere constraint for countries using multiple scripts. However, the time required to train-test the classification models and the execution time taken by the FS methods are also not taken into account in many preceding research works.

Being a classic algorithm, many works have been proposed to date based on GA in the domain of FS. Also, GA and its variants have been applied for the selection of features in different domains, such as classification of schizophrenia using functional magnetic resonance imaging data [40], gene selection in microarray data [17, 18], text clustering and text classification [24], *Persian* font recognition [25], handwritten city name recognition [15, 19], detection of premature ventricular contractions [27], etc. Based on the generalness nature of the GA in FS, we introduce in this paper a method called M-HMOGA, which is applied to select an optimal feature subset to be used for handwritten numeral recognition for *Bangla*, *Devanagari*, and *Roman* scripts. The proposed approach has been applied to two formerly proposed feature vectors for handwritten numeral recognition, namely Mojette transform [43] and *Regional Weighted Run Length* (*RWRL*) features [42].

3 Feature Extraction

As mentioned before, the proposed FS technique has been applied to two state-of-the-art feature vectors introduced by Singh et al. [42, 43]. These feature vectors are extracted for recognition of numerals written in *Bangla*, *Devanagari*, and *Roman* scripts. As these feature vectors are already proposed in the literature, we here discuss them briefly.

3.1 Mojette Transform

Mojette transform [21], also called *projection histogram features*, is derived from "radon transform." The two-dimensional Mojette transform consists of projections where each calculated element called a bin is the sum of pixel values. It is calculated as a set of I discrete projections describing the discrete image f. The projection angles are considered along different orientations $\theta_i = \tan^{-1}(q_i/p_i)$, where $i \in I$ and p_i and q_i are prime integers such that $GCD(p_i, q_i) = 1$. The Mojette transform set is defined by the following equations [45]:

$$Mf = \{M_{p_i,q_i},\ f,\ i = 1,\ \ldots\ldots\ldots,\ I\} = \left\{\text{proj}_{p_i,\ q_i},\ i = 1,\ \ldots,\ I\right\}, \tag{1}$$

$$M_{p,\ q}f(x,\ y) = \text{proj}_{p,\ q}(m) = \text{proj}_{\theta}(m) = f(x,\ y)\Delta(m + qx - py), \tag{2}$$

where $\Delta\ (m)$ is the discrete Kronecker function. For a digital image of size $M \times N$, the number of the i^{th} projection is given by

$$\#\text{bins}_i = (M - 1).|q_i| + (N - 1).|p_i| + 1. \tag{3}$$

The total number of bins is then calculated as

$$\#\text{bins} = \sum_{i=1}^{I} \#\text{bins}_i. \tag{4}$$

Here, the size of handwritten numeral images is kept fixed to 32×32 pixels. As the values of orientations are taken as $\theta_i = 0°, 45°, 90°$, and $135°$, the number of bins corresponding to these projection angles become 32, 63, 32, and 63, respectively. Thus, a feature vector of size 190 ($32 + 63 + 32 + 63$) is extracted using the Mojette transform [43].

3.2 *RWRL* features

The *RWRL* feature extraction methodology consists of three major steps [42]. Firstly, estimation of the contour of the handwritten numeral images is performed. This is done in order to reduce the time complexity of feature computation. Secondly, four types of masks are considered based on four different orientations of the data pixels, such as *vertical*, *horizontal*, and the two oblique lines slanted at $\pm 45°$ to each black pixel using eight-connectivity neighborhood analysis. Finally, the input numeral image (of predefined size 32×32) is divided into imaginary grids of size 8×8. After that, a mask of prefixed size 16×16, formed by overlapping the neighboring 8 pixels in both horizontal and vertical directions, is made to slide over the numeral image. For each mask, four concentric regions, R_i, $i = 1, 2, 3, 4$, are considered as follows:

- R_1 covers a region of size 4×4 in the center of the corresponding subimage.
- R_2 consists of a region of size 8×8 excluding region R_1.
- R_3 is taken as a region of size 12×12, which excludes the preceding two regions.
- R_4 is a region of 16×16 dimensions excluding all three prior regions.

From each of these four regions, the binary transition count (from foreground to background pixels and vice versa) is taken as the feature value. As 16 features are extracted from each of the four regions, a 144 (16 * 9) dimensional feature vector is designed using *RWRL* features.

4 M-HMOGA

In this paper, we propose an upgraded version of HMOGA proposed in Ref. [16], which overcomes some of the shortcomings present in the existing HMOGA model. M-HMOGA includes a memory to remember the candidate solutions in order to keep track of the best solutions obtained over the generations.

In 1975, Holland drafted the basic model of GA [23], which closely represented the formation of child chromosomes from parent chromosomes. It incorporated genetic operations like crossover and mutation to produce the final solution. The basic GA was mainly applied to optimization problems, but it was still not applied to the FS problem. In 1996, Leardi [31] modified the basic GA to make it applicable to solve the FS problem. There were mainly two objectives in any FS problem – reduction in the number of features and increment of the classification accuracy. Hence, to perform efficient FS, MOGA adopted a new fitness function in order to encompass both the aspects of FS, which is shown in Eq. (5). The main concern regarding MOGA is its randomness. On one hand, this random nature gives MOGA a strong place in the meta-heuristic optimization world. On the other hand, it increases the volatility of the method. Hence, only the best solution produced by MOGA cannot clearly ascertain proper exploration of the search space. Ghosh et al. [16] have addressed this problem of MOGA and formed a histogram of the feature attributes from multiple runs of MOGA. As results from different runs are considered, the exploration ability of the entire model increases to some extent. Use of multiple test-train divisions in each run allows for better FS, independent of the divisions used. Depending on the threshold value, some features are selected and some are discarded. The authors have named the overall model as HMOGA. The proposed model makes an improvement over the existing HMOGA model by

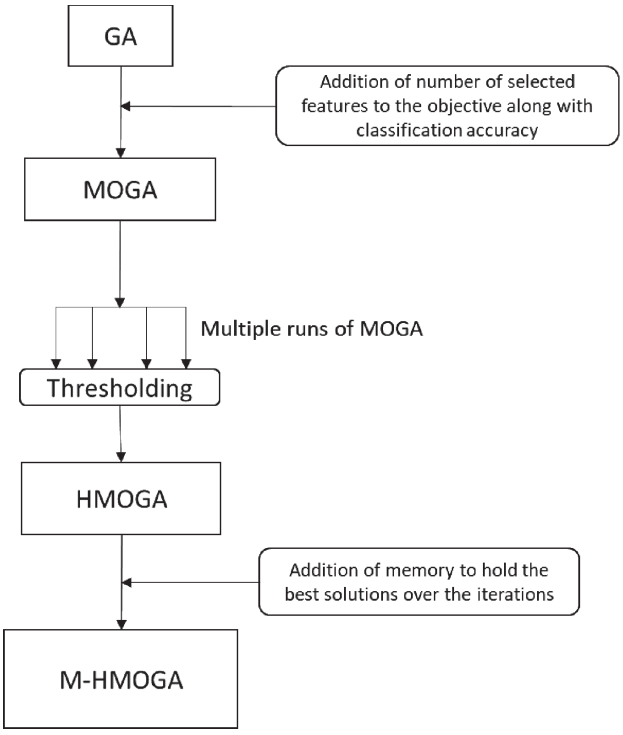

Figure 1: Flow Diagram of Our Proposed M-HMOGA Methodology Evolved Initially from GA.

storing the best set of candidate solutions throughout the generations. The flow from basic GA to M-HMOGA is shown in Figure 1 for better understanding.

In FS, each candidate solution is represented as a vector of 0 s and 1 s. A "0" indicates the exclusion and "1" indicates the inclusion of the corresponding feature in the candidate solution. Each candidate solution has a classification ability depending on the features it has selected from the total feature space. In GA and all its variants, each candidate solution is termed as a chromosome.

MOGA starts with a randomly initialized set of chromosomes called the initial population. The fitness values of the initial population are then calculated using the fitness function represented in Eq. (5).

$$\text{Fitness} = \text{accuracy} * w1 + (1 - c)*w2, \tag{5}$$

where the accuracy of each chromosome is obtained by passing it through a classifier, c is the ratio of the number of features selected to the total number of features present, and $w1$ and $w2$ are the weights assigned to classification ability and number of features selected by the chromosome, respectively. Depending on the fitness values, each chromosome is placed on a roulette wheel containing a pointer. The higher-valued chromosome gets more space on the wheel to increase the probability of its selection as a parent. After spinning the wheel, when it comes to a halt, the chromosome pointed to by the pointer becomes a parent chromosome. In a similar way, another parent chromosome is selected. These parent chromosomes then undergo crossover to create child chromosomes. Crossover helps the chromosomes to achieve much-needed exploration. Meanwhile, mutation helps in exploitation. The newly created child chromosomes then mutate in order to achieve exploitation to some extent. If the mutated child chromosomes exceed their parents in terms of quality, they replace their parents in the population, else the parents are carried over to the next generation. Thus, throughout multiple generations, the population of candidates evolve and proceed toward the final set of solutions.

The random nature of MOGA does not ascertain proper exploration of the search space. Hence, M-HMOGA uses the concept adopted in HMOGA to run MOGA multiple times and combine their final results to reach the ultimate solution. However, unlike HMOGA, which combines the candidate solutions of the final population of each MOGA run, M-HMOGA adds a memory module to each run that stores the best solutions obtained

throughout all the generations and combines the memory module for each MOGA run to reach the final solution. Depending on the replacement criteria of MOGA, there may be certain cases where the model accepts degrading solutions that are undesirable, and consequently the quality of solutions of upcoming generations decreases. The memory module in M-HMOGA circumvents this situation by adding the best solutions obtained over the generations of MOGA to their list.

After finishing all the runs, we finally have a memory module attached to each run of MOGA. The proposed model does not use the final solutions of each run. Instead, it uses the best chromosomes stored in the memory to form the histogram representation of the features. After appropriate thresholding, features exceeding the cutoff are selected and the rest are discarded.

Consider that m is the number of candidates in the memory of each MOGA run, n is the total number of features, and i is the total number of MOGA runs. After all the runs, we have $i * m$ candidate solutions. The weight of each feature is calculated by Eq. (6):

$$w_j = \sum_{x=1}^{i} \sum_{y=1}^{m} f_{xyj} * a_{xy}, \tag{6}$$

where $w_j$is the weight of j^{th} feature, f_{xyj} represents the feature state of j^{th} feature of y^{th} candidate solution of x^{th} MOGA run, and a_{xy} indicates the classification accuracy of y^{th} candidate solution of x^{th} MOGA run. After calculating weights for each feature, the cutoff is calculated using Eq. (7):

$$\text{Histogram cutoff} = \frac{\sqrt[3]{\sum_{j=1}^{n} w_j^3}}{n}. \tag{7}$$

The features that go beyond the histogram cutoff are selected in the final solution and the rest are discarded.

5 Dataset Description

For the evaluation of our proposed model, we have selected three scripts, namely *Bangla*, *Devanagari*, and *Roman*. The experimentations are performed in two stages to prove the efficiency of the method. In the first stage, the same datasets are partitioned to form training and testing samples for the model. In the next stage, training is done using the datasets used in the first stage, but testing is done using completely different datasets. The details of the datasets used for both the stages are provided in this section. At the first stage, we have used three datasets (one for each script). At the second stage, we have used six datasets (one train dataset and one test dataset for each script).

5.1 First Stage Datasets

A few benchmark databases consisting of handwritten *Bangla* and *Devanagari* numerals are available in the literature. However, we have also prepared two in-house numeral datasets written in *Bangla* and *Devanagari* scripts of size 10,000 numerals per script. A large number of people were involved in the data collection process, belonging to varying age, sex, profession, etc. They were asked to write one (or two) set(s) of numerals in A4-sized data sheet consisting of predefined rectangular boxes. Mostly, they used a black or blue ink pen. All datasheets were scanned using a flatbed HP scanner with 300 dpi resolution and stored in .bmp file format. Finally, the numeral images were cropped automatically from the scanned sheets to prepare the isolated digit samples for the experiment. For *Roman* numerals, a dataset of 10,000 samples is formed by random selection from the International Conference on Document Analysis and Recognition 2013 competition on Handwritten Digit Recognition (HDRC 2013) [11]. Figure 2 shows the handwritten samples of *Bangla*, *Devanagari*, and *Roman* numerals. Pre-processing techniques such as Gaussian filter [20] is used to eliminate noise or distortions present in the input numeral image that got introduced due to the poor quality of

Figure 2: Sample Numeral Images Written in (A) *Bangla*, (B) *Devanagari*, and (C) *Roman* Scripts.

Table 1: Outline of Datasets Used for Experiments E1 and E2.

Script	Dataset	Abbreviation	No. of samples	Experimentation setup in step E1	Experimentation setup in step E2
Roman	ICDAR 2013 *Roman*	I1	10,000	Split into train test	Used as train
	MNIST *Roman*	T1	10,000	–	Used as test
Devanagari	In-house *Devanagari*	I2	10,000	Split into train test	Used as train
	ISI *Devanagari*	T2	500	–	Used as test
Bangla	In-house *Bangla*	I3	10,000	Split into train test	Used as train
	ISI *Bangla*	T3	500	–	Used as test

writing instrument or paper on which the digits were written. Then, binarization (for converting the numeral images into two-tone images "0" and "1") is accomplished using an adaptive global threshold value [6]. In the first stage, the above-mentioned three datasets are used. Each dataset with 10,000 samples is partitioned to form 6000 training and 4000 testing samples. For easy reference, *Roman*, *Devanagari*, and *Bangla* numeral datasets are named as I1, I2, and I3, respectively, and this stage of experimentations is named as E1.

5.2 Second Stage Datasets

In the second stage of experimentation, the datasets used in the first stage (i.e. I1, I2, and I3) are used as training datasets, i.e. all 10,000 samples of I1, I2, and I3 are used to train the model in the second stage. For testing the trained model, we have used handwritten numeral datasets of three different scripts. For *Bangla* and *Devanagari*, we have used 500 handwritten numeral samples freely available online on the Indian Statistical Institute (ISI) Kolkata website [5]. For testing on *Roman* numerals, we have used 10,000 test samples of the popular Modified National Institute of Standards and Technology (MNIST) dataset [32]. This stage of experimentations is named as E2. For easy reference, *Roman*, *Devanagari*, and *Bangla* datasets, which are used only as test sets, are named as T1, T2, and T3, respectively. The overall summary of the datasets used in E1 and E2 can be found in Table 1.

6 Experimentations

As described in Section 5, we have performed a set of experiments in E1 to prove the efficiency of the proposed model. Then, to show the robustness of the approach, we have performed another set of experiments in E2. All the results provided for E1 and E2 are on test datasets.

6.1 Experiment E1

For E1, we have used datasets I1, I2, and I3 from which the Mojette transform and *RWRL* features are extracted. Therefore, in total, six feature sets are extracted and then optimized using M-HMOGA. The classifier-independent nature of our method is shown through the use of two of classifiers, namely KNN and MLP.

Table 2: Effect of Change in Population Size with (a) Classification Accuracy of Selected Features and (b) Percentage of Features Selected in E1.

Population size	Mojette transform [43]			*RWRL* [42]		
	Bangla	*Devanagari*	*Roman*	*Bangla*	*Devanagari*	*Roman*
(a) Classification accuracy (in %)						
40	93.925	92.25	93.7	90.825	95.575	85.4
35	94.15	92.5	93.525	91.175	95.375	85.55
30	96.2	93.325	94.75	91.975	96.225	86.35
25	95.775	93.15	94	91.5	95.75	85.4
20	95.975	92.675	94.525	91.55	95.675	85.725
(b) Percentage of selected features (in %)						
40	42.11	44.21	41.58	45.83	40.28	43.75
35	41.05	42.11	41.05	40.28	41.67	38.19
30	43.68	45.79	42.63	39.58	43.06	39.58
25	39.47	39.47	40.00	44.44	43.06	45.14
20	42.11	41.05	40.53	39.58	40.97	40.97

While KNN is faster, its classification ability is not as good as MLP. MLP, however, is computationally expensive and therefore also slower. The parameters of M-HMOGA, like number of iterations and population size, are experimentally fixed to ensure optimal performance. As observed in Table 2a, the population size of 30 yields the best accuracy. The percentage of selected features varies according to the population size; however, from Table 2b, we can see that no particular value yields the lowest percentage of features. As accuracy is a more pressing concern than dimension reduction, the value of 30 is chosen as the optimal size for the population.

Similarly, the variation of accuracy for different numbers of iterations is shown in Table 3a and the percentage of selected features is shown in Table 3b. In this case, the best accuracy occurs for 20 iterations, whereas, for the percentage of features selected, there is no clear value that has the lowest value for all datasets. The optimal value is therefore chosen to be 20 for the number of iterations. Tables 4 and 5 depict the variations of classifier parameters. In Table 4, the value of K in KNN is varied, while in Table 5 the number of neurons in hidden layers of the MLP classifier is varied. For both Tables 4 and 5, (a) is the change in accuracy and (b) is the variation in the percentage of features selected.

The results using KNN and MLP can be seen in Tables 6 and 7, respectively. The accuracies achieved by the proposed model are compared with MOGA and HMOGA, from which our algorithm is derived, and the accuracy without performing any FS. By using KNN as a classifier, we obtained the results presented in

Table 3: Effect of Change in Number of Iterations with (a) Classification Accuracy of Selected Features and (b) Percentage of Features Selected in E1.

No. of iterations	Mojette transform [43]			*RWRL* [42]		
	Bangla	*Devanagari*	*Roman*	*Bangla*	*Devanagari*	*Roman*
(a) Classification accuracy (in %)						
10	96.025	93.175	94.125	91.975	96.175	85.475
15	95.875	92.575	93.975	91.6	95.75	84.1
20	96.2	93.325	94.75	91.975	96.225	86.35
25	95.8	92.7	93.85	91.55	96.1	85.075
30	95.6	92.725	94.45	91.65	95.625	85.45
(b) Percentage of selected features (in %)						
10	37.89	44.74	38.95	44.44	43.06	42.36
15	45.26	39.47	38.42	40.97	40.28	43.75
20	43.68	45.79	42.63	39.58	43.06	39.58
25	37.89	42.11	38.95	40.97	43.75	43.06
30	37.89	41.58	38.95	42.36	40.28	41.67

Table 4: Effect of Change in the Value of K in KNN Classifier with (a) Classification Accuracy of Selected Features and (b) Percentage of Features Selected in E1.

Feature set	Numeral script	$K = 6$	$K = 9$	$K = 12$	$K = 15$	$K = 18$
(a) Obtained classification accuracy (in %) for varying K in KNN classifier						
Mojette transform [43]	*Bangla*	95.075	96.2	93.825	93.9	92.875
	Devanagari	92.65	93.325	92.275	91.725	91.325
	Roman	94.75	92.95	93.975	93.5	92.85
RWRL [42]	*Bangla*	91.975	91.35	90.75	90.925	90.75
	Devanagari	96.225	95.6	95.275	95.325	95.225
	Roman	84.625	86.025	86.35	85.55	85.5
(b) Percentage of selected features for varying K in KNN classifier						
Mojette transform [43]	*Bangla*	42.11	43.68	41.05	45.79	42.63
	Devanagari	44.74	45.79	41.05	43.68	44.21
	Roman	42.63	35.26	44.21	37.89	41.58
RWRL [42]	*Bangla*	39.58	47.92	37.50	43.75	42.36
	Devanagari	43.06	42.36	41.67	40.28	43.06
	Roman	36.81	43.06	39.58	40.97	43.06

Table 5: Effect of Change in Number of Neurons in the Hidden Layer of MLP Classifier with (a) Classification Accuracy of Selected Features and (b) Percentage of Features Selected in E1.

Feature set	Numeral script	No. of neurons					
		50	70	90	110	130	150
Obtained classification accuracy (in %) for varying numbers of neurons in MLP classifier							
Mojette transform [43]	*Bangla*	98.1	98.625	98.325	98.75	99.55	99.4
	Devanagari	92.25	93	93.3	92.725	92.975	93.175
	Roman	92.775	93.6	94.4	93.85	93.975	93.575
RWRL [42]	*Bangla*	93.6	93.275	93.975	93.3	93.475	93.35
	Devanagari	95.3	95.775	95.3	94.325	96.55	95.825
	Roman	86.1	84.85	86.525	86.7	86.98	86.6
Percentage of selected features for varying numbers of neurons in MLP classifier							
Mojette transform [43]	*Bangla*	38.42	44.21	42.63	45.79	41.05	42.63
	Devanagari	38.95	42.63	47.89	40.00	42.63	40.00
	Roman	42.63	44.21	47.89	40.53	42.63	42.11
RWRL [42]	*Bangla*	42.36	43.06	41.67	42.36	44.44	42.36
	Devanagari	45.83	43.75	42.36	46.53	45.83	42.36
	Roman	41.67	41.67	44.44	45.14	45.14	43.06

Table 6. The results show that both in the number of features and accuracy, M-HMOGA outperforms HMOGA as well as MOGA. M-HMOGA decreases the feature dimension by >50% in all the six cases, while improving accuracy by 0.5–1%. In Table 7, the results using the MLP classifier are provided. The results for MLP have a much higher accuracy as compared to KNN, especially in the case of *Bangla* numerals. Here, too, the accuracy without FS is lower than the accuracy obtained by M-HMOGA. Around 1% increase in accuracy can be seen in the case of Mojette transform. As before, a >50% decrease in the number of selected features can also be seen in the case of MLP. In all six feature sets, M-HMOGA outperforms HMOGA and MOGA. Thus, it can be concluded that the use of MOGA multiple times, inclusion of a memory, and using Eq. (7) to find the histogram cutoff help our FS model to perform better. The time required by M-HMOGA to perform FS over all the six datasets using the KNN and MLP classifiers are presented in Table 8a and b, respectively. While calculating the time requirements, all parameters are set to optimal values, such as population size (30), number of iterations (20), and the optimal classifier parameter values for different datasets [listed in Table 8a and b]. From the time requirements, it can be easily seen that using MLP as a classifier is a lot more time consuming than using KNN.

Table 6: Comparison of the Proposed FS Model Called M-HMOGA for Two Feature Sets with HMOGA and MOGA Using KNN as the Classifier in E1 (Best Results Are Made Bold).

Feature set	Method	*Bangla* numerals (dataset I3)		*Devanagari* numerals (dataset I2)		*Roman* numerals (dataset I1)	
		No. of features (in %)	Accuracy (in %)	No. of features	Accuracy (in %)	No. of features (in %)	Accuracy (in %)
Mojette transform [43]	Without using FS	100.00	95.87	100.00	92.93	100.00	94.13
	MOGA	66.31	94.00	56.31	91.40	60.53	93.02
	HMOGA [16]	56.31	95.93	52.63	92.73	48.95	94.10
	M-HMOGA	43.68	**96.20**	45.79	**93.33**	42.63	**94.75**
RWRL [42]	Without using FS	100.00	90.50	100.00	95.45	100.00	84.40
	MOGA	67.36	89.40	63.19	94.72	72.22	83.47
	HMOGA	45.83	91.07	49.31	95.13	50.00	84.40
	M-HMOGA	39.58	**91.98**	43.06	**96.23**	39.58	**86.35**

Table 7: Comparison of the Proposed FS Model Called M-HMOGA for Two Feature Sets with HMOGA and MOGA Using MLP as the Classifier in E1 (Best Results Are Made Bold).

Feature set	Method	*Bangla* numerals (dataset I3)		*Devanagari* numerals (dataset I2)		*Roman* numerals (dataset I1)	
		No. of features (in %)	Accuracy (in %)	No. of features	Accuracy (in %)	No. of features (in %)	Accuracy (in %)
Mojette transform	Without using FS	100.00	98.28	100.00	92.10	100.00	93.45
	MOGA	54.74	98.12	70.53	92.85	57.89	92.90
	HMOGA [16]	49.47	95.93	52.63	92.73	48.95	94.10
	M-HMOGA	41.05	**99.55**	47.37	**93.30**	47.89	**94.40**
RWRL	Without using FS	100.00	92.58	100.00	95.58	100.00	86.88
	MOGA	63.19	91.75	66.67	95.60	68.06	86.67
	HMOGA [16]	**45.14**	92.53	46.53	95.60	50.00	85.75
	M-HMOGA	62.50	**93.98**	45.83	**96.55**	45.14	**86.98**

Table 8: Time Required by M-HMOGA Using (A) KNN Classifier and (b) MLP Classifier Over Different Datasets for Optimal Parameter Settings in E1.

Feature set	Numeral script	Optimal value of *K* for KNN classifier	Execution time for M-HMOGA (in s)
(a)			
Mojette transform [43]	*Bangla*(I3)	9	91.50759
	Devanagari(I2)	9	90.35259
	Roman(I1)	6	188.063
RWRL [42]	*Bangla*(I3)	6	69.7117
	Devanagari(I2)	6	66.314152
	Roman(I1)	12	135.2059
Feature set	**Numeral script**	**Optimal number of neurons chosen for MLP classifier**	**Execution time for M-HMOGA (in s)**
(b)			
Mojette transform [43]	*Bangla*(I3)	130	940.33
	Devanagari(I2)	90	474.72
	Roman(I1)	90	883.67
RWRL [42]	*Bangla*(I3)	90	587.33
	Devanagari(I2)	130	668.39
	Roman(I1)	130	1034.65

Table 9: Comparison of the Proposed FS Model Called M-HMOGA for Two Feature Sets with MOGA and HMOGA Using KNN as the Classifier in E2 (Best Results Are Made Bold).

Feature set	Method	*Bangla* numerals (dataset T3)		*Devanagari* numerals (dataset T2)		*Roman* numerals (dataset T1)	
		No. of selected features (in %)	Accuracy (in %)	No. of selected features (in %)	Accuracy (in %)	No. of selected features (in %)	Accuracy (in %)
Mojette transform [43]	Without using FS	100.00	95.60	100.00	90.20	100.00	88.70
	MOGA	66.31	95.20	56.31	90.40	60.52	89.40
	HMOGA [16]	56.31	95.92	52.63	92.72	48.94	90.60
	M-HMOGA	42.10	**97.40**	42.10	**93.8**	43.63	**91.25**
RWRL [42]	Without using FS	100.00	95.40	100.00	95.45	100.00	83.20
	MOGA	67.36	96.20	63.19	94.72	72.20	83.47
	HMOGA [16]	45.83	95.80	49.30	95.13	50.00	84.40
	M-HMOGA	31.57	**98.40**	32.10	**97.60**	35.58	**84.55**

Table 10: Comparison of the Proposed FS Model Called M-HMOGA for Two Feature Sets with HMOGA and MOGA Using MLP as the Classifier in E2 (Best Results Are Made Bold).

Feature set	Method	*Bangla* numerals (dataset T3)		*Devanagari* numerals (dataset T2)		*Roman* numerals (dataset T1)	
		No. of selected features (in %)	Accuracy (in %)	No. of features (in %)	Accuracy (in %)	No. of selected features (in %)	Accuracy (in %)
Mojette transform [43]	Without using FS	100.00	96.28	100	90.10	100.00	91.45
	MOGA	53.43	96.12	69.22	90.85	56.59	90.9
	HMOGA [16]	48.17	93.93	51.33	90.73	**47.64**	92.10
	M-HMOGA	42.75	**98.05**	49.50	**91.80**	49.59	**92.90**
RWRL [42]	Without using FS	100.00	90.58	100.00	93.58	100.00	84.88
	MOGA	61.89	89.75	65.36	93.60	66.75	84.67
	HMOGA [16]	43.83	90.53	**45.22**	93.60	48.70	83.75
	M-HMOGA	43.36	**92.47**	47.53	**95.05**	46.83	**85.48**

6.2 Experiment E2

To display the robustness of the M-HMOGA model, in addition to E1, we have performed experiment E2 in which we have followed an interesting and innovative train-test approach. We have used the entire datasets I1, I2, and I3 as training datasets for E2, while datasets T1, T2, and T3 were used as test datasets. Again, Mojette transform and *RWRL* features are extracted from the datasets that are optimized using M-HMOGA. Selecting different datasets for training and testing can provide proper evaluation of the proposed model. We have used the same parameter settings [population size, 30; number of iterations, 20; values for K provided in Table 8a; and number of neurons as provided in Table 8b] as those used in experiment E1 for testing the model. Tables 9 and 10 present the obtained results for the proposed method using the KNN and MLP classifiers, respectively, in E2. From the results, we can see that M-HMOGA has achieved around 2–4% increase in classification accuracy while using only 40–50% of the features in experiment E2, which clearly establishes the applicability of the proposed model.

7 Conclusion

Handwritten numeral recognition is an interesting research domain with wide applications, and FS is an important aspect of machine learning. The use of FS in handwritten numeral recognition to enhance accuracy and minimize the computation time is a modern advancement in the field. Three widely used numeral

scripts, namely *Bangla*, *Devanagari*, and *Roman*, are used for the purpose of experimenting our FS model. Two well-established feature descriptors, Mojette transform and *RWRL*, are extracted from the handwritten numeral images written in the said scripts and FS is performed in order to reduce their dimension. In experiment E1, the FS capability of M-HMOGA is depicted using datasets I1, I2, and I3. One unique aspect regarding the experimentation part is that in experiment E2, we have used three handwritten numeral datasets (I1, I2, and I3) for training the model and three completely different script datasets (T1, T2, and T3) for testing it, to prove the robustness of our proposed model. Tests are carried out using both KNN and MLP as classifiers. The features are reduced by $>50\%$ while the accuracy also increases appreciably. The results obtained by M-HMOGA are compared with no FS, MOGA, and HMOGA. From the comparison, it is clear that our algorithm outperforms its ancestors. In the future, we plan to employ the proposed model to solve other pattern recognition problems like word and character recognition, facial emotion identification, etc.

Acknowledgment: The authors are thankful to the Center for Microprocessor Application for Training Education and Research and the Project on Storage Retrieval and Understanding of Video for Multimedia of the Computer Science and Engineering Department, Jadavpur University, for providing infrastructure facilities during the progress of the work. The authors of this paper are thankful to all those individuals who had given appropriate consents and contributed wholeheartedly in developing the handwritten numeral script database used in the current research.

Conflict of Interests: The authors declare that there are no conflict of interests regarding the publication of this paper.

Bibliography

[1] 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.* **64** (2018), 176–194.

[2] M. Z. Alom, P. Sidike, T. M. Taha and V. K. Asari, Handwritten Bangla digit recognition using deep learning, arXiv Prepr. arXiv1705.02680, 2017.

[3] A. Alsaafin and A. Elnagar, A minimal subset of features using feature selection for handwritten digit recognition, *J. Intell. Learn. Syst. Appl.* **9** (2017), 55.

[4] S. Arora, D. Bhattacharjee, M. Nasipuri, M. Kundu, D. K. Basu and L. Malik, Handwritten Devanagari numeral recognition using SVM & ANN, *Int. J. Comput. Sci. Emerg. Technol. (IJCSET)* **1** (2010), 40–46.

[5] U. Bhattacharya and B. B. Chaudhuri, Databases for research on recognition of handwritten characters of Indian scripts, in: *Proceedings of the Eighth IEEE International Conference on Document Analysis and Recognition, 2005*, pp. 789–793, 2005.

[6] S. Bhowmik, R. Sarkar, B. Das and D. Doermann, GiB: a game theory inspired binarization technique for degraded document images, *IEEE Trans. Image Process.* **28** (2019), 1443–1455.

[7] M. Chaudhary, M. H. Mirja and N. K. Mittal, Hindi numeral recognition using neural network, *Int. J. Sci. Eng. Res.* **5** (2014), 260–268.

[8] H. Chouaib, F. Cloppet and N. Vincent, Fast feature selection for handwritten digit recognition, in: *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 485–490, 2012.

[9] *CMATER* [Online], available at: http://www.cmaterju.org/cmaterdb.html, Accessed 26 September, 2018.

[10] C. De Stefano, F. Fontanella, C. Marrocco and A. S. Di Freca, A GA-based feature selection approach with an application to handwritten character recognition, *Pattern Recognit. Lett.* **35** (2014), 130–141.

[11] M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber and R. Sablatnig, ICDAR 2013 Competition on Handwritten Digit Recognition (HDRC 2013), in: *ICDAR*, pp. 1422–1427, 2013.

[12] V. J. Dongre and V. H. Mankar, Devanagari handwritten numeral recognition using geometric features and statistical combination classifier, arXiv Prepr. arXiv1310.5619, 2013.

[13] B. El Kessab, C. Daoui, B. Bouikhalene and R. Salouan, Isolated handwritten Roman numerals recognition using the zoning methods, *Int. J. Comput. Sci. Netw. Sol.* **3** (2015), 8–18.

[14] S. S. Gharde, R. J. Ramteke, V. A. Kotkar and D. D. Bage, Handwritten Devanagari numeral and vowel recognition using invariant moments, in: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pp. 255–260, 2016.

[15] M. Ghosh, S. Malakar, S. Bhowmik, R. Sarkar and M. Nasipuri, Memetic algorithm based feature selection for handwritten city name recognition, in: *International Conference on Computational Intelligence, Communications, and Business Analytics (CICBA 2017)*, J. Mandal, P. Dutta and S. Mukhopadhyay, eds., vol. 776, pp. 599–613, Springer, CCIS, 2017.
[16] M. Ghosh, R. Guha, R. Mondal, P. K. Singh, R. Sarkar and M. Nasipuri, Feature selection using histogram-based multi-objective GA for handwritten Devanagari numeral recognition, *Intell. Eng. Inform.* **695** (2018), 471–479.
[17] M. Ghosh, S. Adhikary, K. K. Ghosh, A. Sardar, S. Begum and R. Sarkar, Genetic algorithm based cancerous gene identification from microarray data using ensemble of filter methods, *Med. Biol. Eng. Comput.* **57** (2019), 159–176.
[18] M. Ghosh, S. Begum, R. Sarkar, D. Chakraborty and U. Maulik, Recursive memetic algorithm for gene selection in microarray data, *Expert Syst. Appl.* **116** (2019), 172–185.
[19] M. Ghosh, S. Malakar, S. Bhowmik, R. Sarkar and M. Nasipuri, Feature selection for handwritten word recognition using memetic algorithm, in: *International Conference on Computational Intelligence, Communications, and Business Analytics (CICBA 2017)*, J. Mandal, P. Dutta and S. Mukhopadhyay, eds., vol. 687, pp. 103–124, Springer, 2019.
[20] R. Gonzalez, *R. woods digital image processing*, Addison-Wesley, Reading, MA, 1992.
[21] J. Guédon and N. Normand, The Mojette transform: the first ten years, in: *International Conference on Discrete Geometry for Computer Imagery*, E. Andres, G. Damiand and P. Lienhardt, eds., LNCS, vol. 3429, pp. 79–91, 2005.
[22] T. Hashem, M. Asif and M. A.-A. Bhuiyan, Handwritten Bangla digit recognition employing hybrid neural network approach, in: *2013 16th International Conference on Computer and Information Technology (ICCIT)*, pp. 360–365, 2014.
[23] J. H. Holland, Genetic algorithms, *Sci. Am.* **1** (1992), 66–73.
[24] S.-S. Hong, W. Lee and M.-M. Han, The feature selection method based on genetic algorithm for efficient of text clustering and text classification, *Int. J. Adv. Soft Comput. Its Appl.* **7** (2015), 22–40.
[25] M. B. Imani, T. Pourhabibi, M. R. Keyvanpour and R. Azmi, A new feature selection method based on ant colony and genetic algorithm on Persian font recognition, *Int. J. Mach. Learn. Comput.* **2** (2012), 278.
[26] A. Kaushik, H. Gupta and D. S. Latwal, Impact of feature selection and engineering in the classification of handwritten text, in: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 2598–2601, 2016.
[27] Y. Kaya and H. Pehlivan, Feature selection using genetic algorithms for premature ventricular contraction classification, in: *2015 9th International Conference on Electrical and Electronics Engineering (ELECO)*, pp. 1229–1232, 2015.
[28] H. A. Khan, A. Al Helal and K. I. Ahmed, Handwritten Bangla digit recognition using sparse representation classifier, in: *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*, pp. 1–6, 2014.
[29] R. V. Kulkarni and P. N. Vasambekar, Isolated handwritten Latin and Devanagari numeral recognition using Fourier descriptors and correlation, in: *International Conference on Mechanical and Electrical Technology, 3rd (ICMET-China 2011)*, ASME Press, New York, vol. 1–3, 2011.
[30] Languages spoken by more than 10 million people, Available at: http://web.archive.org/web/20071203134724/, http://encarta.msn.com/media_701500404/Languages_Spoken_by_More_Than_10_Million_People.html, retrieved 2018-06-03.
[31] R. Leardi, Genetic algorithms in feature selection, in: *Genetic Algorithms in Molecular Modeling*, J. Devillers, ed., pp. 67–86, Academic Press, Elsevier, 1996.
[32] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE* **86** (1998), 2278–2324.
[33] List of countries by English-speaking population, available at: https://en.wikipedia.org/wiki/List_of_countries_by_English-speaking_population, retrieved 2018-06-15.
[34] A. Pandey, A. Kumar, R. Kumar and A. Tiwari, Handwritten Devanagari number recognition using majority voting scheme, *Int. J. Comput. Sci. Inf. Technol. Secur.* **2** (2012), 631–636.
[35] Z. Qing and X. He, Feature extraction and filter in handwritten numeral recognition, in: *Geo-Informatics in Resource Management and Sustainable Ecosystem*, F. Bian, Y. Xie, X. Cui and Y. Zeng, eds., CCIS, vol. 398, pp. 58–67, Springer, 2013.
[36] R. Salouan, S. Safi and B. Bouikhalene, Isolated handwritten Roman numerals recognition using dynamic programming, naïve Bayes and support vectors machines, *Int. J. Comput. Appl.* **113** (2015).
[37] R. Salouan, S. Safi and B. Bouikhalene, Isolated handwritten Roman numerals recognition using methods based on radon, Hough transforms and Gabor filter, *Int. J. Hybrid Inf. Technol.* **8** (2015), 181–194.
[38] R. Sarkhel, N. Das, A. K. Saha and M. Nasipuri, A multi-objective approach towards cost effective isolated handwritten Bangla character and digit recognition, *Pattern Recognit.* **58** (2016), 172–189.
[39] L. M. Seijas, R. F. Carneiro, C. J. Santana, L. S. L. Soares, S. G. T. A. Bezerra and C. J. A. Bastos-Filho, Metaheuristics for feature selection in handwritten digit recognition, in: *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, pp. 1–6, 2015.
[40] H. Shahamat and A. A. Pouyan, Feature selection using genetic algorithm for classification of schizophrenia using fMRI data, *J. AI Data Min.* **3** (2015), 30–37.
[41] P. Singh, A. Verma and N. S. Chaudhari, Feature selection based classifier combination approach for handwritten Devanagari numeral recognition, *Sadhana* **40** (2015), 1701–1714.

[42] P. K. Singh, S. Das, R. Sarkar and M. Nasipuri, Recognition of offline handwritten Devanagari numerals using regional weighted run length features, in: *2016 IEEE International Conference on Computer, Electrical & Communication Engineering (ICCECE)*, pp. 1–6, 2016.

[43] P. K. Singh, S. Das, R. Sarkar and M. Nasipuri, Recognition of handwritten Indic script numerals using Mojette transform, in: *Proceedings of the First International Conference on Intelligent Computing and Communication. Advances in Intelligent Systems and Computing*, J. Mandal, S. Satapathy, M. Sanyal and V. Bhateja, eds., vol. 458, pp. 459–466, Springer, Singapore, 2017.

[44] A. Trivedi, S. Srivastava, A. Mishra, A. Shukla and R. Tiwari, Hybrid evolutionary approach for Devanagari handwritten numeral recognition using Convolutional Neural Network, *Proc. Comput. Sci.* **125** (2018), 525–532.

[45] J. Vásárhelyi and P. Serfözö, Analysis of Mojette transform implementation on reconfigurable hardware, in: *Dagstuhl Seminar Proceedings 06141, Dynamically Reconfigurable Architectures*, P. M. Athanas, J. Becker, G. J. Brebner and J. Teich, eds., 2006, Available at: http://drops.dagstuhl.de/opus/volltexte/2006/746.