Hierarchical K-Nearest Neighbor with GPUs and a High Performance Cluster: application to Handwritten Character Recognition

International Journal of Pattern Recognition and Artificial Intelligence

Hierarchical K-Nearest Neighbor with GPUs and a High Performance Cluster: application to Handwritten Character Recognition

Hubert Cecotti

Intelligent Systems Research Centre
Ulster University (Magee Campus)
Londonderry BT48 7JL, Northern Ireland, UK
h.cecotti@ulster.ac.uk

The accelerating progress and availability of low cost computers, high speed networks, and software for high performance distributed computing allow us to reconsider computationally expensive techniques in image processing and pattern recognition. We propose a two-level hierarchical K-Nearest Neighbor classifier where the first level uses graphics processor units (GPUs) and the second level uses a High Performance Cluster (HPC). The system is evaluated on the problem of character recognition with nine databases (Arabic digits, Indian digits (Bangla, Devnagari, and Oriya), Bangla characters, Indonesian characters, Arabic characters, Farsi characters and digits). Contrary to many approaches that tune the model for different scripts, the proposed image classification method is unchanged throughout the evaluation on the nine databases. We show that a hierarchical combination of decisions based on two distances, using GPUs and a High Performance Cluster provides state-of-the-art performances on several scripts, and provides a better accuracy than more complex systems.

Keywords: k-nearest neighbor; GPU; high performance cluster; handwritten character recognition.

1. Introduction

The evolution of pattern recognition techniques over the years has benefited from advances in both theoretical and hardware. For instance, classifiers based on artificial neural networks have exploited graphics processing units (GPUs) [16], highlighting the benefit of parallel and distributed systems in pattern recognition applications. The corresponding improvements found in theoretical and hardware developments suggest the perpetual evaluation and reconsideration of the methods to use for classification [10,41]. Furthermore, while artificial neural networks have provided state-of-the-art performance since their introduction [34,35], the use of GPUs has significantly increased their interest. One of the key challenges in machine learning techniques applied on large databases, such as natural images or handwritten characters, is to find the best trade-off between the efficiency of the method, the possibility of implementing the method on computer clusters and/or GPUs, and the possibility of updating the system with new data in an incremental way.

The recognition of handwritten characters is a problem that has almost been

overcome thanks to several decades of research [46,22,50,38]. It is mainly due to new methods of machine learning and feature extraction. Yet, the accuracy of single handwritten character recognition remains below 100%, and methods are typically customized for a particular script or database, from the pre-processing steps to the classification. Furthermore, the high accuracy is only available for certain scripts, *e.g.* Latin script. It is therefore pertinent to propose new methods to reach a perfect score. In addition, there are often documents with noisy and/or deformed characters, such as in ancient documents, which cannot be recognized with current optical character recognition (OCR) technologies [2]. The recognition of some characters can be impossible without any contextual information.

In this paper, we propose *(i)* a hierarchical combination of k-nearest neighbor (k-nn) with two distances, and *(ii)* a benchmark of k-nn with nine distances for character recognition in different scripts. While the image distortion model distance is an efficient distance that can be applied on several scripts, its computation time remains important [28]. This paper explores solutions to reduce the number of images processed by a time consuming distance, and the number of prototypes for the comparisons. To solve this problem, we use a parallel implementation, and propose rejection rules to select a limited number of images to test in order to decrease the computation time while maintaining a high accuracy of the non-rejected images. In addition, we evaluate the relationship between the number of prototypes and the error rate. This paper focuses on isolated handwritten character and digit recognition in different scripts. Compared to methods that optimize parameters in relation to only a particular database, the proposed approach is evaluated with the same parameters on all the databases, in order to show the genericity of the approach. This evaluation approach is chosen because it must be known whether a method can be used with different scripts without the selection of specific parameters for each script.

The remainder of the paper is organized as follows: First, we give an overview of the k-nearest neighbor and image matching techniques in character recognition. Then, we present a solution based on a sequential combination of classifiers. In Section 4, we present nine handwritten databases, which are evaluated in Section 5 with the different distances, and the proposed classifiers combination. Finally, the results are discussed in relation to the state-of-the-art in Section 6.

2. Related work

2.1. *Isolated character recognition*

The field of isolated character recognition is largely dominated by deep learning architecture such as convolutional neural networks [15,54], and Support Vector Machines (SVM) [19], or by combining neural networks and SVM [32,47]. The recognition of isolated handwritten characters is a typical benchmark task that is used to evaluate new machine learning methods. Despite the low difference in term of error rate that can occur across methods, this problem remains common in pattern

recognition and machine learning. In addition, while this problem has a key place in the pattern recognition and the document analysis and recognition community, the classification of isolated characters is not perfect, and remains challenging for non-Latin scripts.

2.2. *K-nearest neighbor*

K-nn is a search problem that is often used as a baseline for machine-learning classification algorithms and data mining applications. k-nn classifiers involve the choice of the parameter k, and an appropriate distance for the problem. In pattern recognition, k-nn is one of the oldest methods, and it is often regarded as an inefficient classifier compared to classifiers such as Support Vector Machines or artificial neural networks because of two main problems: first, the inability to obtain an efficient distance for the classification of images, second the high computation cost related to the number of prototypes to compare, and the distance itself. However, several studies have tackled the issues of the computational time by considering data structures (*e.g.* KD-trees [30]) that are more suited to the problem [44]. The probability of error of k-nn is bounded above by twice the Bayes probability of error [17]. Furthermore, k-nn can be easily transferred into a parallel implementation. Both the training database and the test database can be cut into different blocks that can be processed independently as the main goal is to compute the distance between all the possible couples of training/test patterns. This can be achieved with a shared-nothing cluster on a number of commodity machines using MapReduce [18]. Recent studies have proposed novel algorithms in MapReduce to perform efficient parallel k-nn joins on large data [39].

2.3. *Image matching*

In handwritten character recognition, image matching techniques are often poorly regarded due to their high processing time. Image matching techniques are mainly used in relatively small databases such as MPEG7 CE Shape-1 Part B, which contains 1400 binary shape image [31]. For this type of database, shape descriptors must be invariant to scale and rotation [25]. Elastic matching techniques, as an optimization problem of two-dimensional warping (2DW), can be classified into two categories: parametric and non-parametric [55]. This problem is directly related to point matching, which has to cope with the existence of outliers and geometric transformations that may require high dimensional non-rigid mappings [23]. Deformations in handwritten characters can be of two types: first, the global or large deformations such as rotation (with limited angles), scaling, translation; and second, the local deformations that include changes of stroke direction, curvature, and length of the lines. The local deformations depend on the pen/pencil that is used, as the thickness of the lines will depend on what is used to write a character. Due to the different type of deformations that can occur within the same character, it is difficult to determine models of deformations. Image matching can be applied at

the pixel level or by using shape descriptors such as Radon features [52] or Histogram of Oriented Gradients (HOG) [1].

Keysers et al. presented an efficient distance for the classification of handwritten digits based on image distortion models [28]. Moreover, they determined that more complex models (*e.g.* 2-dimensional warping) do not necessarily represent better models compared to the simple image distortion model. Despite the high accuracy on MNIST, with an error rate of only 0.54%, without extending the size of the training database, the method based on the image distortion model has a high computational cost. To reduce its computational cost, it is possible to limit the number of images to process with this distance, and it is also possible to reduce the number of prototypes that should be used during the comparisons of images.

We define the distance L_p between two images I_1 and I_2 of size $N_s \times N_s$ by:

$$L_p = \left(\sum_{i=1}^{N_s} \sum_{j=1}^{N_s} |(I_1(i,j) - I_2(i,j))^p| \right)^{1/p} \quad (1)$$

The Weighted Euclidean distance is defined by:

$$L_{w2} = \sqrt{\sum_{i=1}^{N_s} \sum_{j=1}^{N_s} \frac{1}{s^2(i,j)} (I_1(i,j) - I_2(i,j))^2} \quad (2)$$

where $s^2(i,j)$ is the variance of the point (i,j).

The image distortion model distance (IDMD) takes as input two images I_1 and I_2 of size $N_s \times N_s$. Due to the use of the filters and the displacement fields, the image has to be placed in a larger image with a border of the background color to include the possible shifts. The distance is then computed through a range of pixels from N_{min} to N_{max}. The IDMD is described in the algorithm 1. For each pixel, we consider a displacement field of size w_0 in each direction, corresponding to the elements of a window of size $2w_0 + 1$, a window for the consideration of the neighborhood pixels of size $2w_1 + 1$, and the sum of w_2 values, which are extracted from the application of w_2 transformations on the image. It is worth noting that the *min* function corresponds to the invariance to local deformations, the same way that the invariance to local deformations is achieved through layers with a pooling function, *i.e. max* function in convolutional neural network [45]. For instance, with Sobel filters on the horizontal and vertical direction, $w_2 = 2$. If $w_0 = 0$, $w_1 = 0$, $w_2 = 1$, $p = 2$, it corresponds to the Euclidean distance (L_2) between I_1 and I_2.

The cost factor (c) between the Euclidean distance and IDMD is approximately:

$$c \approx (w_0 + 1)^2 (w_1 + 1)^2 w_2 \quad (3)$$

In the next sections, we consider the following parameters for IDMD: $w_0 = 2$, $w_1 = 1$, and $p = 2$. We evaluate three pre-processing procedures: First, by considering only the pixel value of the image ($w_2 = 1$), then by using Canny descriptors of the image [9], and third, by using images after Sobel filtering. We consider two

Input: images I_1, I_2
Parameters: w_0, w_1, w_2, p
$d \leftarrow 0$
for $i_1 \leftarrow N_{min}, N_{max}$ **do**
  for $j_1 \leftarrow N_{min}, N_{max}$ **do**
    $i_5 \leftarrow 1$
    for $i_2 \leftarrow -w_0, w_0$ **do**
      for $j_2 \leftarrow -w_0, w_0$ **do**
        $s_2 \leftarrow 0$
        for $i_3 \leftarrow -w_1, w_1$ **do**
          for $j_3 \leftarrow -w_1, w_1$ **do**
            for $i_4 \leftarrow 1, w_2$ **do**
              $v_1 \leftarrow I_1(i_1 + i_3, j_1 + j_3, i_4)$
              $v_2 \leftarrow I_2(i_1 + i_3 + i_2, j_1 + j_3 + j_2, i_4)$
              $s_2 \leftarrow s_2 + |v_1 - v_2|^p$
            end for
          end for
        end for
        $s_1(i5) \leftarrow s_2$
        $i_5 \leftarrow i_5 + 1$
      end for
    end for
    $d \leftarrow d + min(s_1)$
  end for
end for
return d

Algorithm 1: Image Deformation Model Distance (IDMD).

sets of filters: horizontal and vertical filters only ($w_2 = 2$), and with the diagonals ($w_2 = 4$). The filters are precised thereafter:

$$f_1 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad f_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{4}$$

$$f_3 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad f_4 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \tag{5}$$

We also consider the method $L_2^{Sobel(4)}$ that considers the distance L_2 with inputs after Sobel filtering (f_1, f_2, f_3, and f_4), and after decreasing the size of the resulting images by two. Hence, the number of features with $L_2^{Sobel(4)}$ is the same as in L_2.

3. Hierarchical processing

Multi-classifier systems can be used to optimize the performance by combining the decision of several classifiers [24,51]. Multi-classifiers with sequential architectures in multi-class classification are typically used with general classifiers at the top of the chain, and more specialized classifiers in the next steps. This type of combination can be used to reduce the number of classes to process, *i.e.* the classifier provides as output a subset of potential classes, then a more dedicated classifier is used for the classification of those subset of classes [12]. In the proposed system, the combination of classifiers is justified by the computational cost that is needed to classify an image. While it would be possible to use the most reliable distance for the evaluation of the whole database, its computational cost may not allow the processing of high volumes of images. We consider a cascade of k-nn classifiers based on two distances. The goal of this classifiers combination is to reduce the processing time for the evaluation of a database of images. A simple distance (L_2) is considered at the first stage, then a more computationally expensive distance (IDMD) is used for the patterns that could not reach a strong decision at the previous level.

Let c_1 and c_2 be the computational cost for two distances, $c_1 << c_2$. We define τ_r by the average rejection rate of method c_1. The total cost for processing an image is estimated by:

$$c_{total} = c_1(1 - \tau_r) + (c_1 + c_2)\tau_r \tag{6}$$

In the worst case, all the images are processed in both stages, $\tau_r = 1$, and $c_{total} = c_1 + c_2$.

The problem is to find a rejection rule that limits the number of images while keeping a high accuracy for the recognized images. First, we compute k-nn with the distance L_2 and k=500, for the selection of the prototypes that will be used in the next step. The evaluation of k-nn with distance L_2 and k=10, which is based on the distances computed for k=500, includes a rejection rule: all the k best answers must belong to the same class in order to accept the decision. If it is not the case, then the image is evaluated with IDMD, where the training prototypes are the 500 best answers obtained with distance L_2 at the previous stage, with k=3. At the second level, images can be rejected with the same decision rule as in the first level: all the k best answers must belong to the same class. The proposed approach is depicted in Fig. 1.

4. Databases

Nine databases of handwritten digits and/or characters have been chosen for the analysis of the performance. Each database is evaluated separately. The first two databases, MNIST and CVL, contain images of ten Arabic numerals (Latin script). The next three databases contain images of digits in three Indian scripts: Bangla, Devnagari, and Oriya. The next two databases have characters of Bangla and Lampung. The HACDB database contains handwritten Arabic characters, and the

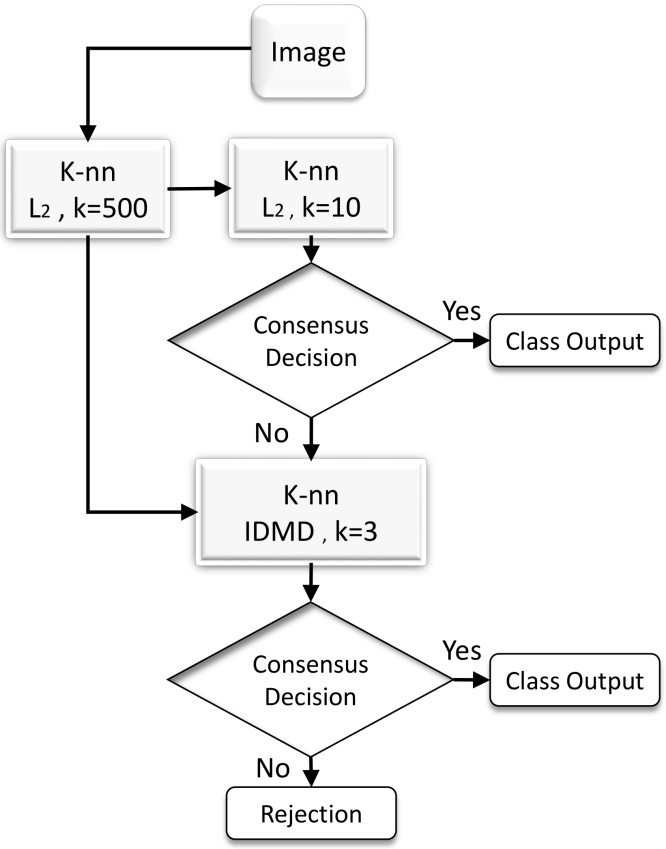


Fig. 1. Multi-classifier system overview.

IFHCDB database has handwritten Farsi digits and characters. The content of each database is described in the following subsections. Samples of digits are presented in Fig. 2. As we want to obtain a method that can be applied on different data sets of single handwritten characters, all the images from the different databases were normalized the same way. The images were preprocessed the same way as in the original images in the MNIST database. Because some databases have very noisy images and/or images in color, images were first binarized with the Otsu method at their original size [48], then they were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain 8 bit gray levels due to the bicubic interpolation for resizing the images. Finally, all the images were centered in a 28x28 pixel box field by computing the center of mass of the pixels, and translating the gravity center of the image to the center of the 28x28 field. Tables 1 and 2 present for each database the number of classes, the total number of images in the database, and the number of images per class, for both training and the test.

Table 1. Characteristics of the handwritten databases (digits, 10 classes).

Database	MNIST	CVL	Devnagari	Oriya	Bangla$_{digit}$
			Training		
# samples	60000	14000	18783	4970	19392
# per class	6000 ± 339	1400	1878 ± 15	497 ± 3	360
size (x)	28	47 ± 12	65 ± 16	73 ± 25	58 ± 16
size (y)	28	104 ± 30	62 ± 19	73 ± 26	54 ± 16
			Test		
# samples	10000	21780	3763	1000	4000
# per class	1000 ± 62	2178	376 ± 3	100	400
size (x)	28	50 ± 12	66 ± 17	75 ± 25	59 ± 17
size (y)	28	106 ± 31	62 ± 20	74 ± 26	54 ± 18

Table 2. Characteristics of the handwritten databases (characters).

Database	Bangla$_{char}$	Lampung	HACDB	IFHCDB
# classes	50	18	66	47
			Training	
# samples	18000	23447	5280	49101
# per class	1939 ± 9	1302 ± 825	80	1045 ± 1355
size (x)	78 ± 25	32	128 ± 2	77
size (y)	74 ± 27	32	128 ± 12	95
			Test	
# samples	12859	7853	1320	20653
# per class	257 ± 224	436 ± 276	20	440 ± 576
size (x)	74 ± 24	32	128	77
size (y)	66 ± 24	32	128	95

4.1. *Western Arabic digits*

The MNIST database is a benchmark in supervised classifiers [35,54]. It includes a training and test database of 60000 and 10000 images, respectively. For the comparison of techniques, two characteristics are typically precised if they are used or not: the addition of distorted images in the database, and the type of normalization of the images. The error rate reaches quasi human performance level of 0.23% with a combination of 35 convolutional neural networks, where each network requires almost 14h to be trained [15]. With k-nn classifiers, the best error rate is 0.52% by using a Pseudo 2D Hidden Markov Models [28], followed by Image Deformation Model with an error rate of 0.54%, and 0.63% with shape matching using shape contexts [4]. While the Pseudo 2D Hidden Markov Models or the Image Deformation Model are interesting approaches, their computation time remains an issue. With the undistorted and unprocessed MNIST data-set, the lowest error rate is 0.53%

(a) MNIST (b) Bangla (c) CVL (d) Devnagari (e) Oriya (f) Farsi

Fig. 2. Representative handwritten digits for the different databases (from zero to nine).

(a) Bangla (b) Lampung (c) Arabic (d) Farsi

Fig. 3. Handwritten characters for the different scripts. **(a)** Bangla basic characters (first row: vowels; rows 2, 3, 4 and 5: consonants; last row: others). **(b)** Representative samples for the 18 classes of the Lampung database. **(c)** The 52 different shapes of Arabic characters without diacritics. **(d)** Representative samples for each of the 35 classes of Farsi characters.

with a large convolutional neural network with unsupervised pre-training [26].

The Computer Vision Lab of the Vienna University of Technology, Austria (CVL) database was used during the International Conference on Document Analysis and Recognition (ICDAR) 2013 Competition on Handwritten Digit Recognition [21]. The images in the CVL database are in RGB color, not size-normalized, and in original size with a resolution of 300 dpi. For this competition, the best methods were based on Finite Impulse Response Multilayer Perceptron (Fir-MLP) partially and fully connected with four layers, and by including affine deformations of the input patterns [3]. In this neural network, the static weights (synapses) were replaced with finite impulse response filters. With one Fir-MLP and an ensemble of four Fir-MLPs, the error rate was 3.28% and 2.26%, respectively.

4.2. *Indian digits and characters*

The databases of Indian digits were created at the Indian Statistical Institute, Kolkata, India [13,49,5]. The databases contain Bangla digits. Bangla is the fourth most popular script in the world, used by more than 200 million people [14,57]. In this database, class-specific feature polynomial classifier with input features [36] based on 8 gradient direction histogram with 5x5 sampling provided an accuracy of 99.40% [37]. The second database corresponds to Devnagari digits, which is part of the Brahmic family of scripts of India, Nepal, Tibet, and South-East Asia [53]. The third database contains Oriya digits [8]. This script is one of the many descendants of the Brahmi script of ancient India. Bhowmik et al. obtain an accuracy of 90.50% by using Hidden Markov Models [8].

The fourth database contains only handwritten Bangla characters [7]. Whereas there are about 300 characters, a large number is based on the combination of basic characters. The 50 basic characters, with their corresponding Harvard-Kyoto transcription that are used for classification, are depicted in Fig. 3(a). In [7], they propose a two-stage framework that combines modified quadratic discriminant function (MQDF) [29] and MLPs, and obtain an accuracy of 95.84%. On other databases of the same problem, Mandal et al. use features based on the combination of gradient features and Haar wavelet coefficients at different scales with a k-nn classifier to reach an accuracy of 88.95% [40]. Bhattacharya et al. establish a method based on a chain code histogram feature with an MLP classifier, and obtain an accuracy of 88.95% [6].

4.3. *Lampung characters*

Lampung is a non-cursive script used in Indonesia. Images of characters are depicted in Fig. 3(b). The database contains 32, 140 separated characters that were extracted from 82 documents. For the evaluation of the methods, 23, 447 characters were considered for training, 7, 853 characters were considered for test and the remaining 840 characters served as validation. Altogether 18 different character classes were identified. The database was already preprocessed, and characters were initially normalized into 32x32 grayscale images. A more detailed description of the data

set can be found in [56,27]. For the classification of characters, Junaidi et al. obtain an accuracy of 94.27% by considering a multi-layer perceptron with input features based on the skeleton of images, and water reservoirs in order to exploit the different cavities and the subsequent measures of the character shapes.

4.4. *Arabic characters*

The handwritten Arabic characters database for automatic character recognition (HACDB) contains 66 classes of different Arabic characters, or overlapping characters collected from 50 writers [33]. Arabic script is composed of 28 characters, with no capital or lower case. Each character has two or four shapes, which depends on its position in the word (beginning, middle, end, or isolated). Hence, 81 different shapes are possible. However, the shape of some characters is similar, because the only differences come from the number and position of dots. Without the dots, there are 52 different shapes in Arabic characters, as presented in Fig. 3(c). In addition, some overlapping characters added into the HACDB as one shape, which leads to 66 classes.

4.5. *Farsi/Persian characters*

The IFHCDB database includes grayscale images of isolated Farsi/Persian characters (35 classes), and numerals (12 classes) gathered from real-life documents in Farsi language [42,43]. The documents were scanned from Iranian school entrance exam forms, between the years of 2004 and 2006, at 300 dpi. The ten digits have twelve separated shapes, two for the digit '4' and '6', which are considered as separated classes (see Fig. 2(f)). Samples of Farsi characters for each class are provided in Fig. 3(d).

5. Results

5.1. *Evaluation*

The proposed approach and different distances have been evaluated independently across nine databases. For the evaluation of the performance, we consider the mean error rate across the whole database. For pairwise comparisons, we consider a Bonferroni correction, and we use the non-parametric Wilcoxon signed-rank test [20], which does not assume a normal distribution, for the estimation of the p-values (p).

5.2. *Distance evaluation*

The error rate for the nine databases, by using only the distances L_1, L_2, L_3, L_{w2}, and $L_2^{Sobel(4)}$ is presented in Tables 3 and 4; the best results for each database are in bold. The input of the k-nn classifiers are the images in gray level of size 28x28.

Table 3. Error rate (in %) with only the distance L_1, L_2, and L_3. Inputs are pixel values.

D	k	MNIST	CVL	Devnagari	Oriya	Bangla$_{digit}$
L_1	1	3.69	11.67	3.96	9.40	6.75
	3	3.67	11.90	4.04	9.30	6.68
	5	3.82	11.90	3.93	7.70	6.70
L_2	1	3.09	10.48	3.91	8.80	6.53
	3	2.95	10.67	3.75	8.00	6.30
	5	3.12	10.73	3.61	**7.50**	6.45
L_3	1	2.83	**10.16**	4.07	9.20	6.58
	3	2.82	10.32	3.72	9.10	**6.00**
	5	**2.81**	10.39	3.67	8.40	6.38
L_{w2}	1	5.66	13.31	9.15	6.70	11.90
	3	5.48	13.28	8.97	7.39	11.30
	5	5.57	13.09	9.12	7.89	10.60
$L_w^{Sobel(4)}$	1	2.98	11.20	3.70	8.80	6.37
	3	2.99	11.45	**3.51**	8.20	11.45
	5	2.96	11.49	3.67	8.00	11.49

Pairwise comparisons revealed that L_1 gives a lower performance than L_2 (p <10e-4). There is no difference between L_2 and L_3, and between L_1 and L_3, across the databases. For the choice of k in k-nn, the only differences are between k=5 and k=10 (p <10e-4), and between k=3 and k=10 (p <10e-2).

The performance obtained with only IDMD is presented in Tables 5 and 6. The parameters $w_0 = 2$ and $w_1 = 1$ were estimated on the training database of MNIST by comparing IDMD scores between all the images, and their closest neighbor of the same class. Pairwise comparisons revealed that the best method is IDMD$_{sobel}$, followed by IDMD$_{pixel}$, and IDMD$_{canny}$. In addition, the impact of the addition of diagonal filters (f_3 and f_4), 4 directions, is not significant compared to only horizontal and vertical Sobel filters. The worst performance is obtained with k=10. As there is no significant difference between k=1 and k=3, we consider k=3 in the remaining evaluation of the paper. With an error rate of around 0.65% on MNIST, IDMD$_{sobel}$ belongs to the set of the best efficient methods for MNIST. With IDMD$_{sobel}$, the error rate (2.75%) is significantly lower than the score obtained in [27] for the Lampung database (5.73%).

The error rate decreases with the addition of more prototypes with k-nn (k=3). The speed of the classification is linearly dependent on the number of prototypes used during the comparisons. For all the databases, we observe the same pattern of performance with an error decreasing significantly with the increase of prototypes, until 200, then it reaches a plateau where the error decreases slowly. For instance, the error decreases from 0.92% with 100 prototypes to 0.65% with 300 prototypes, *i.e.* by multiplying the processing time by three.

Table 4. Error rate (in %) with only the distance L_1, L_2, and L_3. Inputs are pixel values.

D	k	Bangla$_{char}$	Lampung	HACDB	IFHCDB
L_1	1	30.21	15.55	**34.32**	13.07
	3	33.74	15.55	36.14	11.68
	5	32.87	16.36	36.44	11.14
L_2	1	**29.48**	12.20	35.76	12.53
	3	33.89	12.40	36.82	11.11
	5	33.02	12.20	36.14	**10.69**
L_3	1	30.22	**10.89**	37.42	12.36
	3	35.41	11.18	38.64	11.07
	5	35.14	11.42	37.35	10.74
L_{w2}	1	40.18	18.15	38.26	14.95
	3	46.14	18.03	40.68	13.63
	5	45.52	18.55	39.77	13.38
$L_w^{Sobel(4)}$	1	31.19	11.42	34.92	12.69
	3	36.51	11.61	36.29	11.29
	5	36.01	12.09	36.44	10.88

Table 5. Error rate for the handwritten digit databases (in %) with only IDMD ($w_0 = 2, w_1 = 1$), and considering the 500 best prototypes from k-nn with L_2.

Input	k	MNIST	CVL	Devnagari	Oriya	Bangla$_{digit}$
Pixel	1	1.02	4.26	1.01	2.70	2.08
	3	0.90	3.87	0.82	3.50	2.20
	5	0.93	3.87	0.82	3.50	2.20
Canny	1	1.90	4.61	1.36	3.20	2.08
	3	1.69	4.31	1.30	3.10	2.28
	5	1.69	4.10	1.41	3.30	2.08
Sobel (2 dir.)	1	0.73	3.14	0.88	2.00	1.45
	3	**0.64**	**3.00**	0.80	2.10	1.72
	5	0.68	3.06	0.80	2.50	1.77
Sobel (4 dir.)	1	0.71	3.14	0.85	**1.90**	**1.45**
	3	0.66	3.02	**0.77**	2.00	1.70
	5	0.65	3.06	0.77	2.30	1.88

The error rate in relation to the n best answers provided with k-nn ($k \geq n$) is given in Tables 7 and 8. If the correct label is present among the n best answers, then the image is considered to be successfully detected. This evaluation allows us to determine to what extent some confusion may impair the overall classification accuracy. While it is unlikely to find contextual information for the detection of single digits, *e.g.* in technical maps, zipcodes, we observe a decrease of the error

Table 6. Error rate for the handwritten character databases (in %) with only IDMD ($w_0 = 2, w_1 = 1$), and considering the 500 best prototypes from k-nn with L_2.

Input	k	Bangla$_{char}$	Lampung	HACDB	IFHCDB
Pixel	1	13.03	5.95	19.02	10.18
	3	14.41	5.01	19.55	8.41
	5	14.26	5.57	19.32	8.10
Canny	1	17.58	6.53	23.71	12.42
	3	18.31	5.77	22.96	10.03
	5	17.37	5.58	21.67	9.54
Sobel (2 dir.)	1	**10.80**	3.20	17.05	9.54
	3	11.76	2.93	16.89	8.05
	5	12.35	2.79	**15.76**	**7.62**
Sobel (4 dir.)	1	10.89	3.18	16.97	9.57
	3	11.97	2.80	17.05	8.67
	5	12.44	**2.75**	15.91	7.71

Table 7. Error rate (in %) for the top n best answer with IDMD$_{sobel}$, k=3.

Top n	MNIST	CVL	Devnagari	Oriya	Bangla$_{digit}$
1	0.66	3.02	0.77	2.00	1.70
2	0.31	1.53	0.29	1.00	0.72
3	0.28	1.29	0.19	0.90	0.45

Table 8. Error rate (in %) for the top n best answer with IDMD$_{sobel}$, k=5.

Top n	Bangla$_{char}$	Lampung	HACDB	IFHCDB
1	12.43	2.75	15.91	7.71
2	6.03	1.09	8.63	2.28
3	4.34	0.80	6.21	2.06
4	3.76	0.75	5.30	2.01
5	3.61	0.71	5.07	1.98

rate from 0.66% to 0.31%, by considering the two best answers. The same pattern of performance is observed for the other databases of handwritten digits. For the databases of handwritten characters, the error drops from 12.43 to 6.03, 2.75 to 1.09, 15.91 to 8.63, and 7.71 to 2.28, for Bangla$_{char}$, Lampung, HACDB, and IFHCDB, respectively. The most important decrease of the error is obtained for the databases with a large amount of classes, where the scripts suggest possible confusions between characters, particularly for Arabic characters, but also for Bangla characters due to their size normalization.

The impact of the number of prototypes based on the results of k-nn with L_2 on

Table 9. Performance (in %) for the combination of L_2 (level 1) and IDMD$_{sobel}$ (level 2), ($w_0 = 2, w_1 = 1$), with and without rejection in the second level.

		MNIST	CVL	Devnagari	Oriya	Bangla$_{digit}$
	lvl-1 error	0.44	0.82	0.14	0.65	0.28
	lvl-1 rejection	15.81	38.69	24.10	38.30	29.92
Without	lvl-2 error	3.10	6.91	2.98	4.96	5.35
rejection	total error	0.86	3.17	0.82	2.30	1.80
With	lvl-2 error	1.10	2.55	0.83	1.75	1.84
rejection	lvl-2 rejection	8.10	13.85	6.72	10.70	9.36
	total error	0.54	1.42	0.30	1.04	0.72
	total rejection	1.28	5.36	1.62	4.10	2.80

the error rate is depicted in Fig. 6. In most of the databases, the error rate reaches a plateau with more than 100 prototypes. The evolution of the error rate in relation to the rejection rate of k-nn based on a consensus decision is depicted in Fig. 4 for L_2, and Fig. 5 for $IDMD_{sobel}$. Each point in the curves correspond to the choice of k in the decision. For the MNIST, the error is 0.21%, which is approximately acknowledged as the human error rate, with a rejection rate of 2.06%. For a perfect accuracy, 8.51% of the patterns should be rejected.

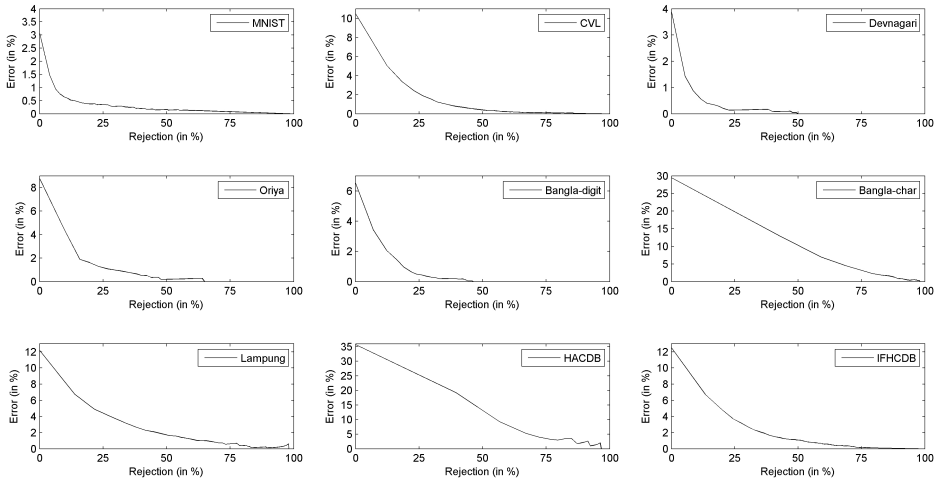


Fig. 4. Evolution of the error rate in relation to the rejection rate based on a consensus decision given by k, with L_2.

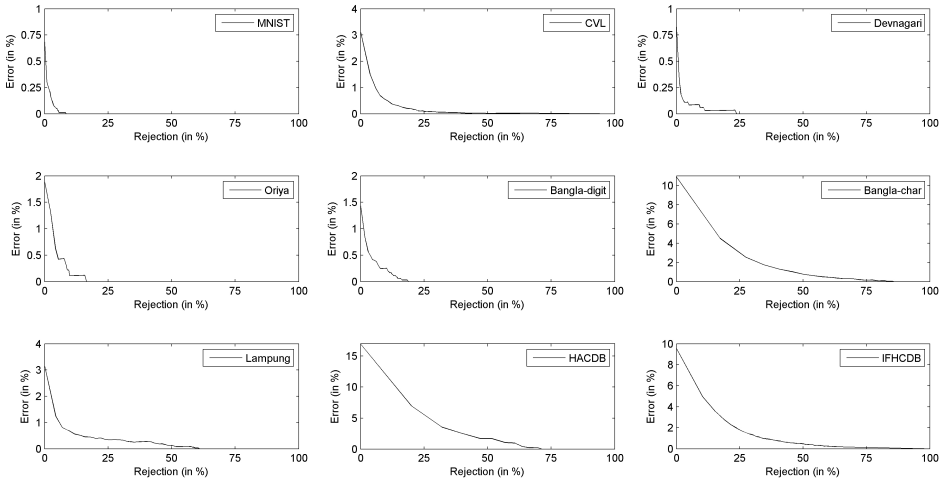


Fig. 5. Evolution of the error rate in relation to the rejection rate based on a consensus decision given by k, with IDMD_{sobel}.

Table 10. Performance (in %) for the combination of L_2 (level 1) and IDMD_{sobel} (level 2), ($w_0 = 2, w_1 = 1$), with and without rejection in the second level.

		Bangla_{char}	Lampung	HACDB	IFHCDB
	lvl-1 error	0.83	3.63	1.76	1.71
	lvl-1 rejection	89.71	58.66	87.12	38.53
Without	lvl-2 error	12.21	4.36	18.17	18.24
rejection	total error	11.04	4.06	16.06	8.08
With	lvl-2 error	2.86	1.11	4.50	9.50
rejection	lvl-2 rejection	30.29	10.44	36.17	34.91
	total error	2.57	2.22	3.98	3.96
	total rejection	27.17	6.12	31.51	13.45

5.3. *Rejection and subset of prototypes*

The results corresponding to the cascade of decisions between L_2 and IDMD_{sobel} are presented in Tables 9 and 10. First, despite a high threshold for the decision, there remains an error rate of 0.44% with the distance L_2 on MNIST. With the rejection option of the first level, 15.81% of the images are rejected to be processed by the second level. In the second level, which is intended to process the most difficult images, the error rate is 3.10%. Finally, the overall error rate on MNIST reaches 0.86%, showing that the combination of distances is as effective as a convolutional neural network using huge distortions [35]. The same pattern of performance is observed on other databases of handwritten digits. However, for the other databases of characters, which contain noisy images and a large intraclass variability, the re-

Table 11. Comparison with other methods on MNIST with no distortion in the training database. ([1]: it depends on the chosen trade-off between performance and processing time.)

Method	Error rate (%)
IDMD [28]	0.54
P2DHMM [28]	0.52
Shape Matching [4]	0.63
Large Conv. Net (unsup. pretraining) [26]	0.52
Conv. Net+SVM [32]	0.83
This paper (IDMD)	0.64
This paper[1] (L2+IDMD)	0.86

jection rate is high, involving the processing of most of the images at the second level. When we consider the possibility of rejecting images in the last stage of the combination (level 2), an error rate of 0.54% is obtained with a rejection rate of 1.28% of the test database.

5.4. *Implementation and Computational demands*

The methods were implemented with Matlab R2013a using a dedicated parallel implementation (Fig. 7). With a parallel implementation on three GPU cards (NVIDIA Tesla C1060), it takes about 237 seconds to get the distances for k-nn with the Euclidean distance (L_2) and k=500, for processing the MNIST test database. The training database is split into three blocks that are processed by each GPU. In this case, the Euclidean distance between a test image and the block of training images is directly computed using the GPU. With a parallel implementation on a cluster using 50 cores (Intel Xeon X5650 2.66 Ghz) where the test database is split into 50 blocks, it takes about 700 seconds to process the MNIST test database with k-nn, IDMD$_{sobel}$, 500 prototypes, and k=3. For IDMD, contrary to the Euclidean distance with GPUs, the test database is split. Each core processes a subset of test examples using the selection of prototypes that were obtained in the previous stage. The processing time is linearly dependent on the number of prototypes, and the number of images for the test. By applying the rejection rule at only the second level, the current implementation allows us to process MNIST under 6 min, with an error rate of 0.86% (there is no training as it is a k-nn classifier). This error rate is close to the combination of a convolutional neural network and SVM proposed in [32], with an error rate of 0.83%. State of the art results are given in Table 11. It is worth noting that the main strength of the approach is the ratio processing time/performance, and not the performance only.

6. Discussion

The goal of this paper was to propose an efficient combination of distances that can be easily implemented on a computer cluster, and aims at becoming a stan-

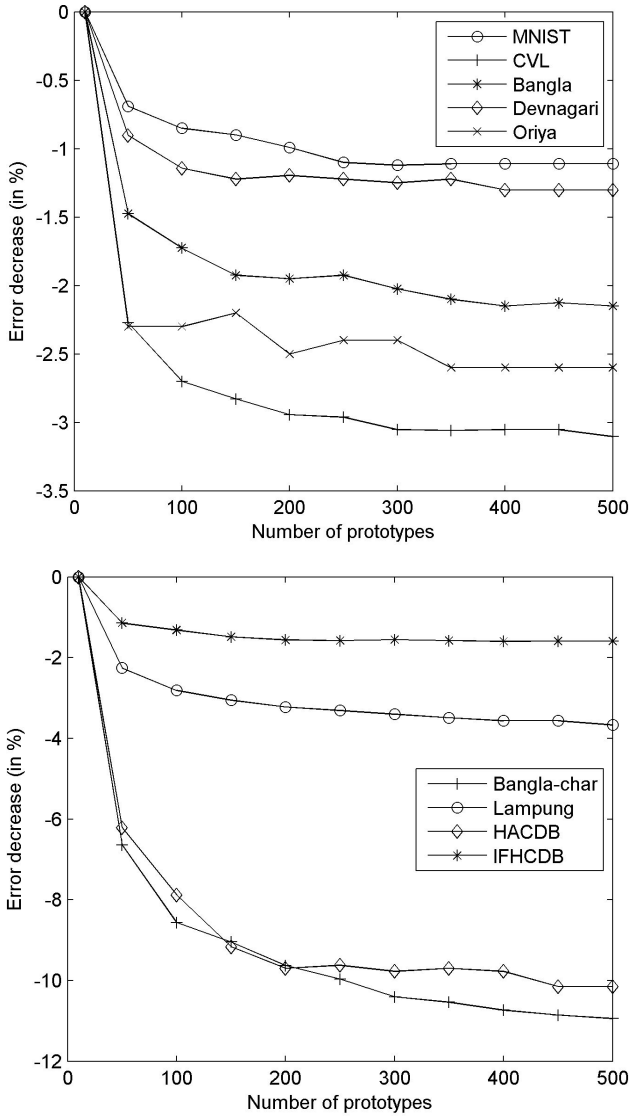
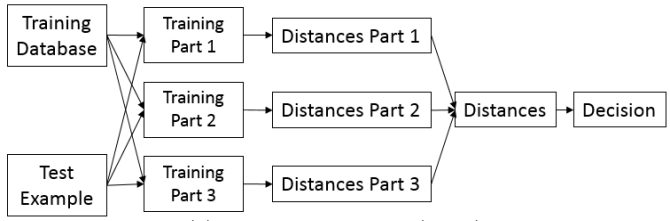
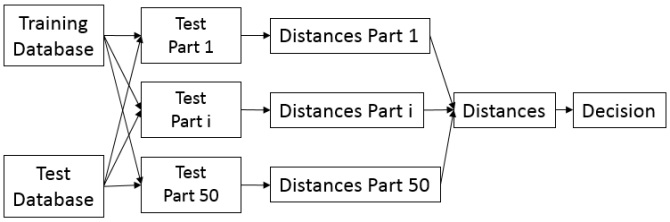


Fig. 6. Evolution of the error rate (in %) in relation to the number of prototypes based on k-nn with L_2. The baseline corresponds to the use of 10 prototypes.

dard of comparison in handwritten character recognition benchmarks, thanks to its processing time and its easy implementation. By using the same processing technique throughout nine different databases of handwritten characters or digits, we have shown that it is possible to reach state-of-the art performances with a single method, without adapting the parameters to each database. The best error rate is obtained with the IDMD (0.64%), which is close to 0.54% obtained with a large



(a) Euclidean distance (GPU)



(b) IDMD (HPC)

Fig. 7. Parallel implementation approach for k-nn with the Euclidean distance (Fig. 7(a)), and with IDMD (Fig. 7(b)).

convolutional network, with no distortion of the images for training the model. For the CVL database, the IDMD would have come in second place in the competition with an error rate of 3%. While MNIST and the CVL database are dealing with the same problem, the recognition of handwritten digits, the error rate between the two databases is significantly different, highlighting the complexity of the problem, and the amount of variation that may occur across images. For the Lampung database, IDMD outperforms the state-of-the-art method, from an error rate of 5.73% to 2.75%. With the hierarchical combination of distances, the processing time is significantly improved while keeping a low error rate (0.86%).

Whereas methods such as convolutional neural networks propose the best results in MNIST [15] and other databases of images and signals [11], it can be difficult to determine the ideal architecture of the network without prior expertise relating to this problem. It is particularly true if artificial characters are created with elastic deformations, some knowledge of the script must be assumed: the parameters of the deformations, *e.g.* intensity, parameters of the Gaussian for filtering the random fields, must be carefully chosen. With IDMD, only three main parameters have to be chosen: the size of the possible shifts (w_0), the size of the neighborhood of each pixel (w_1), and the number of pre-processed images (w_2). To some extent, the three parameters can be assimilated to three transformations, and IDMD may be seen as a "deep distance". In addition, a filtering method (*e.g.* Sobel) must be chosen to pre-process the images.

Since such a distance is computationally expensive, the technique must take advantage of parallel computing and computer clusters. We have proposed a cas-

cade of classifiers based on their computation times, which allows focusing on difficult images. Therefore, the speedup can be increased by both the number of computers/cores, and the rejection rule that limits the number of images to process with IDMD. The evolution of parallel computing enables a shift of paradigm for how machine learning and pattern recognition techniques should be employed. Image processing techniques that may only be used for natural image retrieval due to their computational costs may be used for handwritten character recognition thanks to parallel computing. The performance of the methods was assessed on nine databases, including seven different scripts. For some scripts, the introduction of images with important deformations may involve the introduction of patterns that do not correspond to the right class. The observation of the errors indicated that most of the errors, particularly for MNIST, were due to characters that were thick (*e.g.* images 948, 2463, 2131 in the test database). IDMD allows the invariance to local deformation but it remains sensitive to transformations such as dilatation, and large rotations. Since the possible deformations are dependent of the script, additional information could be added in IDMD. For instance, w_0 could be set in relation to the distance to the center of the images because points that are far from the center are more likely to be displaced at a further distance than points that are close to the center for rotated images. Finally, artificial patterns can be added in the databases to increase the size of the training database, and therefore increase the accuracy [54], but the parameters of the transformations have to be tuned in relation to the script.

In this study, the goal was to determine an efficient combination of distances for handwritten characters and digits from several scripts with a minimum number of hyper-parameters, and without prior knowledge of the script. The method is deterministic as it does not require the random initialization of some parameters. An important contribution in the present work is the benchmark of several distances for single handwritten character recognition, which are independent of the script: the parameters were set and used throughout the different data sets, ignoring the peculiar aspects of each script (*e.g.* structural features), in order to keep the same processing pipeline as the Latin digits (*e.g.* MNIST). This strategy has been proved successful as the accuracy reaches state-of-the-art levels in all the databases. The database of Indian digits and characters is noisy, and may require better denoising techniques. Moreover, a field of 28x28 can limit the performance of the method because it is probably too small to capture all the features of the Indian and Arabic characters. Furthermore, the gravity center may not be the ideal normalization technique for Indian characters, as it may be more judicious to exploit other structural features such as the presence of a headline.

7. Conclusion

This paper presents a benchmark of different distances on different scripts, and a comprehensive hierarchical combination of distances between gray level images

to obtain a relatively fast estimation of the performance for single handwritten character recognition thanks to the use of both GPUs and a High Performance Cluster. While classifiers based on non-approximate k-nn have poor performance for handwritten characters due to the L_2 distance, we have shown that the method is relevant for the recognition of handwritten digits, and characters in different scripts, with appropriate distances. By combining the advantages of parallel computing such as clusters and GPUs, and rejection rules, it is possible to significantly increase the speed of the approach, allowing the use of distances that are computationally expensive, such as the Image Deformation Model Distance. Further works will be carried out to optimize the parameters of the deformations in relation to a predefined script.

Acknowledgment

The databases used in the paper were obtained free of cost, the author would like to thank Prof. Ujjwal Bhattacharya for sharing the databases of Bangla, Devnagari, and Oriya [a], the group of Prof. Ahmed Bouridane for the HACDB database, the CVL group for their single digit database [b], the group of Prof. Karim Faez for the IFHCDB database [c], Dr. Szilard Vajda for the Lampung database, and Prof. Yann LeCun for sharing MNIST on his website [d].

References

1. J. Almazán, A. Fornés and E. Velvety, Deformable HOG-based shape descriptor, in *Proc. of ICDAR* (2013) pp. 1022–1026.
2. N. Arica and F. T. Yarman-Vural, An overview of character recognition focused on off-line handwriting, *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews* **31** (May 2001) 216–233.
3. A. Back and A. Tsoi, A time series modeling methodology using FIR and IIR synapses, in *Proc. of the Workshop on Neural Networks for Statistical and Economic Data* (1990) p. 187194.
4. S. Belongie, J. Malik and J. Puzicha, Shape matching and object recognition using shape contexts, *IEEE Trans. Pattern Analysis and Machine Intelligence* **24** (Apr. 2002) 509–522.
5. U. Bhattacharya and B. Chaudhuri, Databases for research on recognition of handwritten characters of indian scripts, in *Proc. of the 8th Int. Conf. on Document Analysis and Recognition (ICDAR'05)* (2005) pp. 789–793.
6. U. Bhattacharya, M. Shridhar and S. Parui, On recognition of handwritten bangla characters, in *Proc. of the 5th Indian Conf. on Computer Vision, Graphics and Image Processing (ICVGIP)* (2006) pp. 817–828.
7. U. Bhattacharya, M. Shridhar, S. Parui, P. Sen and B. Chaudhuri, Offline recognition of handwritten Bangla characters: an efficient two-stage approach, *Pattern Analysis and Applications* **15**(4) (2012) 445–458.

[a] `http://www.isical.ac.in/~ujjwal/download/database.html`
[b] `http://caa.tuwien.ac.at/cvl/research/icdar2013-hdrc/`
[c] `http://ele.aut.ac.ir/~imageproc/`
[d] `http://yann.lecun.com/exdb/mnist/`

8. T. Bhowmick, S. Parui, U. Bhattacharya and B. Shaw, An HMM based recognition scheme for handwritten oriya numerals, in *Proc. of the 9th Int. Conf. on Information Technology (ICIT 2006)* (2006) pp. 105–110.
9. J. Canny, A computational approach to edge detection, *IEEE Trans. Pattern Analysis and Machine Intelligence* **8** (Nov. 1986) 679–698.
10. R. Caruana and A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in *Proc. of the 23rd Int. Conf. on Machine learning* (2006) pp. 161–168.
11. H. Cecotti, M. P. Eckstein and B. Giesbrecht, Single-trial classification of event-related potentials in rapid serial visual presentation tasks using supervised spatial filtering, *IEEE Trans. Neural Networks and Learning Systems* **15** (Nov. 2014) 2030–42.
12. H. Cecotti and A. Belaid, Hybrid OCR combination approach complemented by a specialized ICR applied on ancient documents, in *Proc. of the 8th Int. Conf. on Document Analysis and Recognition (ICDAR)*, Vol. 2 (2005) pp. 1045–1049.
13. B. B. Chaudhuri and U. Pal, A complete printed Bangla OCR system, *Pattern Recognition* **31** (1998) 531–549.
14. B. Chaudhuri, A complete handwritten numeral database of Bangla - a major Indic script, in *Proc. of the 10th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR'10)* (2006) pp. 1–6.
15. D. Cireşan, U. Meier and J. Schmidhuber, Multi-column deep neural networks for image classification, in *Computer Vision and Pattern Recognition (CVPR)* (2012) pp. 3642–3649.
16. D. Ciresan, U. Meier, L. M. Gambardella and J. Schmidhuber, Deep, big, simple neural nets for handwritten digit recognition, *Neural Computation* **12** (2010) 3207–3220.
17. T. Cover and P. Hart, Nearest neighbor pattern classification, *IEEE Trans. Information Theory* **13**(1) (1967) 21–27.
18. J. Dean and S. Ghemawat, Mapreduce: simplified data processing on large clusters, *Communications of the ACM* **51** (Jan. 2008) 107–113.
19. D. DeCoste and B. Schölkopf, Training invariant support vector machines, *Machine Learning* **46**(1-3) (2002) 161–190.
20. J. Demsãr, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research* **7** (2006) 1–30.
21. M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber and R. Sablatnig, ICDAR 2013 competition on handwritten digit recognition (HDRC 2013), in *Proc. of the 12th Int. Conf. on Document Analysis and Recognition (ICDAR)* (2013) pp. 1454–1459.
22. D. Ghosh, T. Dube and A. Shivaprasad, Script recognition: A review, *IEEE Trans. Pattern Analysis and Machine Intelligence* **32** (Dec. 2010) 2142–2161.
23. S. Gold, A. Rangarajan, C. P. Lu, S. Pappu and E. Mjolsness, New algorithms for 2-d and 3-d point matching: pose estimation and correspondence, *Pattern Recognition* **31**(8) (1998) 1019–1031.
24. T. K. Ho, J. J. Hull and S. N. Srihari, Decision combination in multiple classifier systems, *IEEE Trans. Pattern Analysis and Machine Intelligence* **16** (Jan. 1994) 66–75.
25. R.-X. Hu, W. Jia, H. Ling, Y. Zhao and J. Gui, Angular pattern and binary angular pattern for shape retrieval, *IEEE Trans. Image Processing* **23** (Mar. 2014) 1118–1127.
26. K. Jarrett, K. Kavukcuoglu, M. Ranzato and Y. LeCun, What is the best multi-stage architecture for object recognition?, in *Proc. of the 12th Int. Conf. on Computer Vision (ICCV'09)* (2009) pp. 2146–2153.
27. A. Junaidi, S. Vajda and G. Fink, Lampung - a new handwritten character benchmark: Database, labeling and recognition, in *Proc. of the Int. Workshop on Multilingual OCR. ACM* (2011) pp. 105–112.

28. D. Keysers, T. Deselaers, C. Gollan and H. Ney, Deformation models for image recognition, *IEEE Trans. Pattern Analysis and Machine Intelligence* **29** (Aug. 2007) 1422–1435.
29. F. Kimura, K. Takashina, S. Tsuruoka and Y. Miyake, Modified quadratic discriminant functions and the application to Chinese character recognition, *IEEE Trans. Pattern Analysis and Machine Intelligence* **9** (1987) 149–153.
30. B. L., Multidimensional binary search trees used for associative searching, *Commun. ACM* **18**(9) (1975) 509–517.
31. L. J. Latecki, R. Lakämper and U. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in *Proc of the IEEE Conf. on Computer Vision and Pattern Recognition*, Vol. 1 (June 2000) pp. 424–429.
32. F. Lauer, C. Y. Suen and G. Bloch, A trainable feature extractor for handwritten digit recognition, *Pattern Recognition* **40**(6) (2007) 1816–1824.
33. A. Lawgali, M. Angelova and A. Bouridane, Hacdb: Handwritten arabic characters database for automatic character recognition., in *European Workshop on Visual Information Processing (EUVIP)* (2013) pp. 255–259.
34. Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, Handwritten digit recognition with a back-propagation network, in *Advances in Neural Information Processing Systems (NIPS) 2* (1990) pp. 396–404.
35. Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE* **86** (Nov. 1998) 2278–2324.
36. C.-L. Liu and H. Sako, Class-specific feature polynomial classifier for pattern classification and its application to handwritten numeral recognition, *Pattern Recognition* **39**(4) (2006) 669–681.
37. C.-L. Liu and C. Y. Suen, A new benchmark on the recognition of handwritten bangla and farsi numeral characters, *Pattern Recognition* **42** (2009) 3287–3295.
38. C. Liu, K. Nakashima, H. Sako and H. Fujisawa, Handwritten digit recognition: Benchmarking of state-of-the-art techniques, *Pattern Recognition* **36** (Oct. 2003) 2271–2285.
39. W. Lu, Y. Shen, S. Chen and B. C. Ooi, Efficient processing of k nearest neighbor joins using MapReduce, in *Proc. of the VLDB Endowment VLDB Endowment Hompage archive*, Vol. 5 (June 2012) pp. 1016–1027.
40. S. Mandal, S. Sur, A. Dan and P. Bhowmick, Handwritten Bangla character recognition in machine-printed forms using gradient information and Haar wavelet, in *Proc. of the 2011 Int. Conf. on Image Information Processing (ICIIP)* (2011) pp. 1–6.
41. M. Manuel Fernández-Delgado, E. Cernadas, S. Barro and D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, *Journal of Machine Learning Research* **15** (2014) 3133–3181.
42. S. Mozaffari, K. Faez and H. Kanan, Recognition of isolated handwritten farsi/arabic alphanumeric using fractal codes, in *IEEE Proc. of Southwest Symposium on Image Analysis and Interpretation (SSIAI)* (2004) pp. 104–108.
43. S. Mozaffari, K. Faez, F. Faradji, M. Ziaratban and S. Golzan, A comprehensive isolated farsi/arabic character database for handwritten OCR research, in *Proc of the Int. Workshop on Frontiers in Handwritten Recognition (IWFHR)* (2006) pp. 385–389.
44. M. Muja and D. G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, *IEEE Trans. Pattern Analysis and Machine Intelligence* **36** (Nov. 2014) 2227–2240.
45. J. Nagi, F. Ducatelle, G. A. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber and L. M. Gambardella, Max-pooling convolutional neural networks for vision-based hand gesture recognition, in *Proc. of the IEEE Int. Conf. on Signal and Image Processing Applications* (2011) pp. 342–347.

46. N. Nagy, Twenty years of document image analysis in PAMI, *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**(1) (2000) 38–62.
47. X.-X. Niu and C. Y. Suen, A novel hybrid cnn-svm classifier for recognizing handwritten digits, *Pattern Recognition* **45**(6) (2012) 1318–1325.
48. N. Otsu, A threshold selection method from gray-level histograms, *IEEE Trans. Sys. Man. Cyber.* **9**(1) (1979) 62–66.
49. U. Pal and B. B. Chaudhuri, Indian script character recognition: a survey, *Pattern Recognition* **37** (Sept. 2004) 1887–1899.
50. R. Plamondon and N. S. Srihari, On-line and off-line handwritten recognition: a comprehensive survey, *IEEE Trans. Pattern Analysis and Machine Intelligence* **22** (Jan. 2000) 63–84.
51. A. Rahman and M. Fairhurst, Multiple classifier decision combination strategies for character recognition: A review, *Int. J. of Document Analysis and Recognition (IJDAR)* **5**(4) (2003) 166–194.
52. K. C. Santosh and L. Wending, Character recognition based on non-linear multiprojection profiles measure, *Frontiers of Computer Science* **9**(5) (2015) 678–690.
53. I. K. Sethi and B. Chatterjee, Machine recognition of constrained hand printed devanagari, *Pattern Recognition* **9** (July 1977) 69–75.
54. P. Simard, D. Steinkraus and J. Platt, Best practices for convolutional neural networks applied to visual document analysis, in *Proc. of the 7th Int. Conf. Document Analysis and Recognition (ICDAR)* (Aug. 2003) pp. 958–962.
55. S. Uchida and H. Sakoe, Survey of elastic matching techniques for handwritten character recognition, *IEICE Trans. Information and Systems* **88**(8) (2005) 1781–1790.
56. S. Vajda, A. Junaidi and G. A. Fink, A semi-supervised ensemble learning approach for character labeling with minimal human effort, in *Proc. of the 11th Int. Conf. on Document Analysis and Recognition (ICDAR)* (2011) pp. 259–263.
57. S. Vajda, K. Roy, U. Pal, B. Chaudhuri and A. Belaïd, Automation of Indian postal documents written in Bangla and English, *Int. Journal of Pattern Recognition and Artificial Intelligence* **23** (Dec. 2009) 1599–1632.