Handwritten word preprocessing for database adaptation

Cristina Oprean, Laurence Likforman-Sulem, Chafic Mokbel

Handwritten word preprocessing for database adaptation

Cristina Oprean[a], Laurence Likforman-Sulem[a], Chafic Mokbel[b]
[a]Institut Mines-Telecom/Telecom ParisTech and CNRS LTCI, 46 rue Barrault, 75013 Paris (France)
[b]University of Balamand, Faculty of Engineering, P.O. Box 100 Tripoli (Liban)

ABSTRACT

Handwriting recognition systems are typically trained using publicly available databases, where data have been collected in controlled conditions (image resolution, paper background, noise level, ...). Since this is not often the case in real-world scenarios, classification performance can be affected when novel data is presented to the word recognition system. To overcome this problem, we present in this paper a new approach called *database adaptation*. It consists of processing one set (training or test) in order to adapt it to the other set (test or training, respectively). Specifically, two kinds of preprocessing, namely stroke thickness normalization and pixel intensity normalization are considered. The advantage of such approach is that we can re-use the existing recognition system trained on controlled data. We conduct several experiments with the Rimes 2011 word database and with a real-world database. We adapt either the test set or the training set. Results show that training set adaptation achieves better results than test set adaptation, at the cost of a second training stage on the adapted data. Accuracy of data set adaptation is increased by 2% to 3% in absolute value over no adaptation.

Keywords: Handwritten word recognition, database adaptation, word preprocessing

1. INTRODUCTION

Handwriting recognition systems can read strings of characters, words and even text lines, and are of fundamental importance to automatize repetitive tasks which were traditionally reserved to humans, such as bank check and envelope reading,[1] or postal mail sorting and analysis.[2] These recognition systems are generally trained using databases of handwritten documents, words and text blocks.[3–5] These datasets are typically collected from volunteer writers by simulating different application scenarios — such as mailing, invoice writing, etc. — using a controlled acquisition protocol. As a result, word images collected in these datasets are *regular*, in the sense that they share similar characteristics in terms of image quality, noise level, intensity and thickness of letter strokes.

Conversely, real-world images are collected in a variety of acquisition conditions, e.g., they may come from different sources (scanned text or fax-simile images), with different resolution, and their background may include noise (lines, stains, rulings). As a consequence, recognition systems trained on a large amount of clean and regular data may fail to classify novel, noisy or irregular test data. Writer adaptation techniques (e.g. maximum a-posteriori)[6,7] have been proposed in order to re-use a word recognition system trained on labelled, clean data on novel, irregular data. However, these techniques are inefficient when data come from different writers.

In this paper we propose a dataset adaptation approach to cope with the variability of real-world images, by taking advantage of existing recognition systems[5] trained on regular data. There are several solutions to tackle this variability. The first one is to collect a large amount of labelled real-world data and re-train the system from scratch. This solution assumes that a large amount of labelled data is available. This is not often the case since data may be either partially/totally unlabelled or mislabelled. In both cases, data needs to be manually labelled before being usable, which could be a time-consuming task. The second solution consists in adding noisy/irregular data to the training set and re-training the system with the mixed (clean and irregular) data. The rationale behind this strategy is that a larger variability in the training set leads to higher classification performance on novel, irregular test data. However it still needs manual labeling and a strategy to balance the training set with regular and real-world data.

In this paper, we propose a novel technique, which we refer to as *dataset adaptation*. This adaptation is useful for recognizing real-world test data which differ considerably from the training data. Real-world test data are not similar to training data because they come from diverse text corpora such as: invoices, cheques, mails, financial

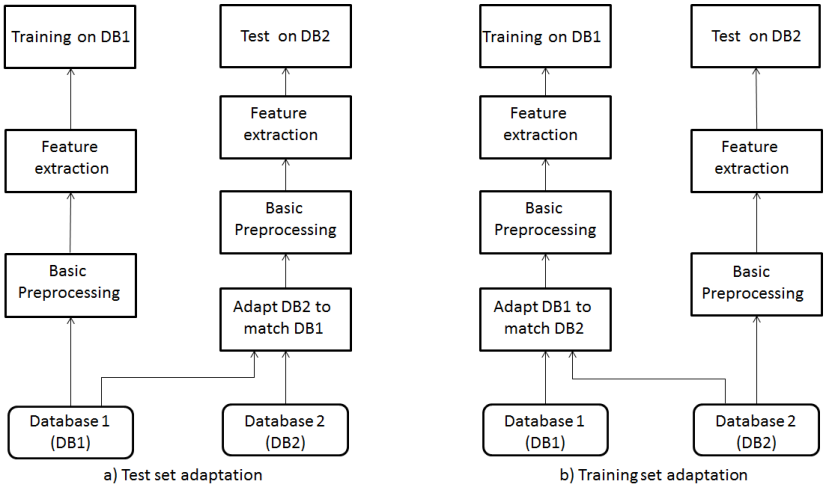

Figure 1. Database adaptation. Each set (training or test) has different characteristics and is processed differently. The resulting adapted set matches stroke thickness and intensity characteristics of the other set.

files, etc. Let DB1 denote the dataset originating from a single text source, and DB2 the dataset originating from different text sources. The first approach we propose is *test set adaptation* and aims at reducing the variability of the test corpus with respect to the training material. The test set (which is a subset of DB2) is pre-processed in order to have similar characteristics — e.g., stroke thickness, grayscale intensity — as the training set (which is a subset of DB1). This pre-processing is carried out *before* the standard pre-processing (de-slanting, etc.) applied to both DB1 and DB2,[8] as shown in Figure 1. The second approach, which we call *training set adaptation*, aims to render the training material more similar to the dataset used for testing. We achieve this by modifying the images of DB1 in order to match characteristics of DB2. In other words, we process regular training images so that they look like real-world ones. The pre-processing for adaptation applied in this case is not necessarily the same as in the test set adaptation. Both approaches have the advantage to employ only regular images to train the system. Notice that only in training set adaptation, the recognition system has to be re-trained.

In order to demonstrate the effectiveness of the two approaches, we tested them on two datasets coming from different sources. One set (DB1) includes gray-level images, while the second one (DB2) contains binary images with irregular writing. The pre-processing we propose for dataset adaptation includes stroke thickness and stroke intensity normalizations, whose parameters depend on whether test or training adaptation is performed. This approach can be generalized to other datasets whose characteristics such as resolution or intensity differ.

The article is organised as follows: Section 2 presents in detail the preprocessing methods for dataset adaptation. Section 3 describes the databases, the recognition system and the results of our experiments. Conclusions and future work are discussed in Section 4.

2. DATABASE ADAPTATION

This section presents in detail the two novel approaches for *dataset adaptation* mentioned in Section 1 to cope with the discrepancy between datasets. We consider two datasets DB1 and DB2 which contain gray-scale and binary images, respectively. Images in DB1 have thicker strokes, but non-uniform intensity of pixels, while DB2 images have a lower resolution and thinner strokes. An example of word images from the two datasets is given in Figure 4. We adopt the recognition system described in,[9] which has been designed for gray-scale images like those in DB1. Specifically, the feature extraction in[9] includes features that are computed on gray-scale pixel values. In order to make DB1 and DB2 more similar to each other, we can perform two independent kinds of dataset adaptation:

- *Test set adaptation*: we increase and normalize the thickness of DB2 to be similar to DB1.

- *Training set adaptation*: Since DB2 images are binary, they first need to be converted to gray-scale in order to use the system in.[9] After gray-scale conversion, the stroke intensity of DB2 images is uniform. This is a source of variability with respect to images in DB1, whose gray-scale intensity is not in general uniform. Therefore, we pre-process DB1 so that the strokes in its images have the same uniformity in intensity as DB2's after gray-scale conversion.

2.1 Test set adaptation: Stroke thickness normalization

Stroke thickness normalization allows us to adapt the test set to the training set as shown in Figure 1(a). In addition to our adaptation goal, stroke thickness normalization also reduces the variability of writing, due to pen pressure on the paper and pen size, and thus has a positive impact on feature extraction and recognition performance. We determine the stroke thickness and normalize all images in DB2 to the average stroke thickness of DB1 using the following two-steps procedure. First, the average stroke thickness for the training set is computed as follows: since the training set contains gray-level images, each of them is binarized with a global threshold. For the obtained binary image, the foreground pixels are replaced with the Euclidean distance from the background. We refer to the obtained image as *distance transform* image. The values in the center of the strokes in the distance transform image represent half of the distance to the closest background pixel. We are interested only on these pixel values. In order to obtain them we calculate the skeleton of the distance transform image. We approximate the average stroke thickness as twice the average distance value on the skeleton. The average stroke thickness for the entire train dataset is the average of the stroke thickness values obtained for each image. We denote this value by $avgAllTrain$. The algorithm for word gray-scale image stroke thickness is outlined in the pseudo-code of Algorithm 1.

Algorithm 1 Pseudo-code for stroke thickness determination for a gray-scale image

```
Input:  im = gray-scale image
        bim = binary_transform(global_threshold, im)
        dim = euclidean_distance_transform(bim)
        sim = skeletonization(dim)
        dist_transform_value_list = dim(sim)
Output: avgTrainImage = 2*average(dist_transform_value_list)
```

In the second step, we compute the average stroke thickness for each word image from the test dataset DB2. In order to compute the average stroke thickness for a binary test word image, we apply Algorithm 1 with the only difference that no binarization has to be carried out. Let *avgTest* be the average value of stroke thickness for a binary test word image. Depending of this value, morphological operations are applied (dilations, erosions) until the value *avgAllTrain* is reached. At the end, morphological closure is applied. The tuning of parameters for morphological filters is described in section 3.3.2. The algorithm for stroke thickness normalization is given in Algorithm 2.

Algorithm 2 Pseudo-code for stroke thickness normalization

```
For each train image
        Calculate the average stroke thickness avgTrainImage
Calculate Average stroke thickness avgAllTrain of all train images
For each test image
        Calculate the average stroke thickness avgTestImage
        While |avgTestImage-avgAllTrain|>certain value
                If avgTestImage>avgAllTrain then
                        Apply morphological erosion on test image
                else
                        Apply morphological dilation on test image
                Apply morphological closure
```

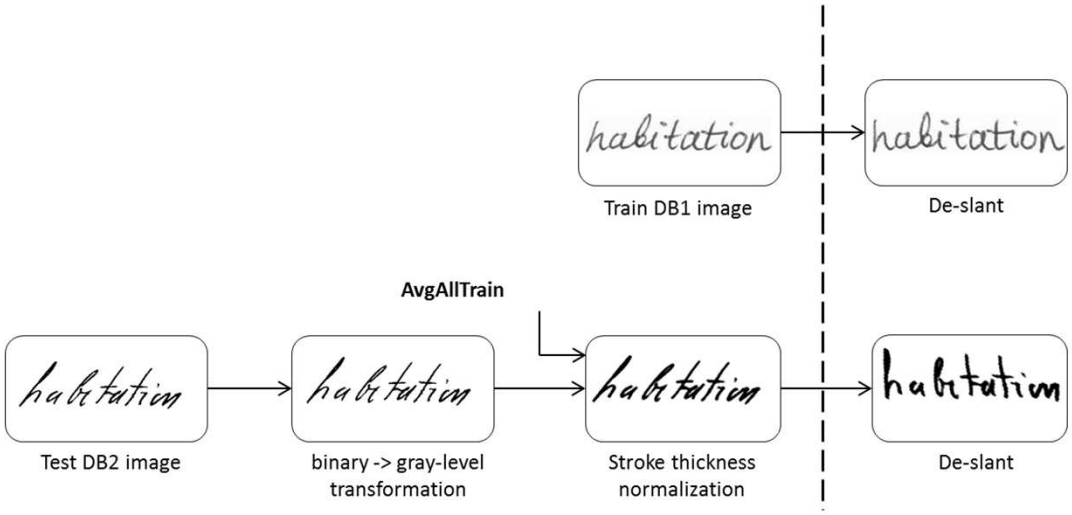

Figure 2. Test database adaptation through stroke thickness normalization

The main advantage of this solution is represented by the automatic reduction of variability of data introduced by the real-world datasets, while using only regular (labelled data we dispose, written in controlled conditions). We report the performance improvement obtained through test set adaptation in Section 3.

We provide a visual example of test set adaptation in Figure 2.

2.2 Training set adaptation: Stroke intensity normalisation

In training set adaptation we modify the stroke intensity of training images to increase its uniformity, in order to match the characteristics of test word images, as shown in Figure 1(b). The proposed stroke intensity normalization algorithm is composed of three steps: i) the training images are binarized with a local threshold; ii) morphological closure is applied; and iii) the binary image is transformed back to gray-scale using Gaussian smoothing on the binary image. Sauvola algorithm is applied to binarize the training images.[10] This algorithm is applied in order to determine the local threshold for each pixel at position (x, y), which is obtained as:

$$T(x,y) = m(x,y)\left[1 + k\left(\frac{s(x,y)}{R} - 1\right)\right], \tag{1}$$

where m and s are the local mean and standard deviation in a square neighbourhood of the pixel, R is a constant, and the coefficient k is a real positive number. Since Sauvola's algorithm is applied locally, it preserves better stroke continuity than a global thresholding method. The dimensions of the window used for calculating the average and the standard deviation in the pixel neighbourhood depend on the average stroke thickness of the word image. If the window size is smaller than the stroke average thickness of the word image, after binarization the stroke will be damaged. So the window size is chosen greater than the average stroke thickness of training dataset: $avgTrainImage$.

After binarization, there could still be some discontinuities inside the strokes. For this reason, in the second step of the algorithm we carry out a morphological closure to further reduce the artefacts left by binarization. This operation offers an additional advantage by increasing the variability of the strokes in the training set, which would better fit the variability in the real dataset. After this step, the strokes in the word image are restored almost completely. Finally, the third step consists in applying a Gaussian smoothing in order to transform the binary image back to a gray-scale one. The algorithm for stroke intensity normalization is given in Algorithm 3.

Algorithm 3 Pseudo-code for stroke intensity normalization

For each train image
Calculate average stroke thickness of the word image
Choose Sauvola parameters corresponding to the average stroke thickness
Binarize with Sauvola algorithm
Apply morphological closure
Apply Gaussian smoothing

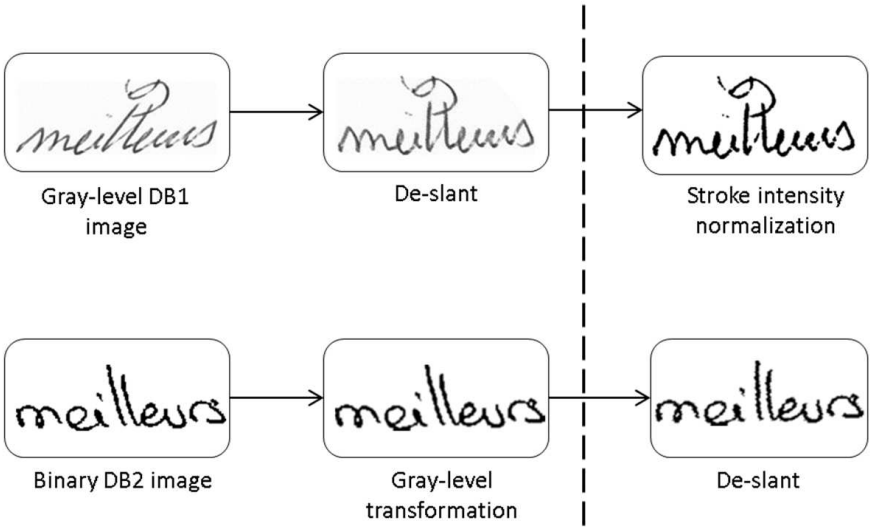

Figure 3. Training database adaptation through stroke intensity normalization

The advantage of this solution is that we adapt automatically to the variability (in pixel intensity) of the real-world word images, while using regular data (labelled data we dispose, written in controlled conditions). We compare both approaches (training set adaptation and test set adaptation) in Section 3. We provide a visual example of training set adaptation in Figure 3.

3. EXPERIMENTAL RESULTS

In this section we demonstrate the effectiveness of our dataset adaptation technique using a regular dataset used in competitions and a real-world dataset.

3.1 Databases

The database DB1, which contains grey-level word images generated in controlled conditions is the publicly available Rimes 2011 database (Reconnaissance et Indexation de données Manuscrites et de fax-similés).[11] It is a French database created by asking volunteers to write fictive and unconstrained letters on a white paper using black ink. It contains 12,723 pages, corresponding to 5,605 mails. In Figure 4, on the left side, words from Rimes database can be seen. DB1 is divided in three parts: a training database which has 51,739 word images, a test database containing 7,776 images and a validation database with 7,464 images. Dictionaries for training, test and validation sets are made of 4,279, 1,588 and 1,612 words, respectively.

DB2 comes from a company focused on capturing incoming handwritten forms. This word database is more irregular than the Rimes database used for competitions. DB2 consists of binary word images since they were constrained by the available storage capacity. Some samples of this dataset can be seen on the right side of Figure 4.

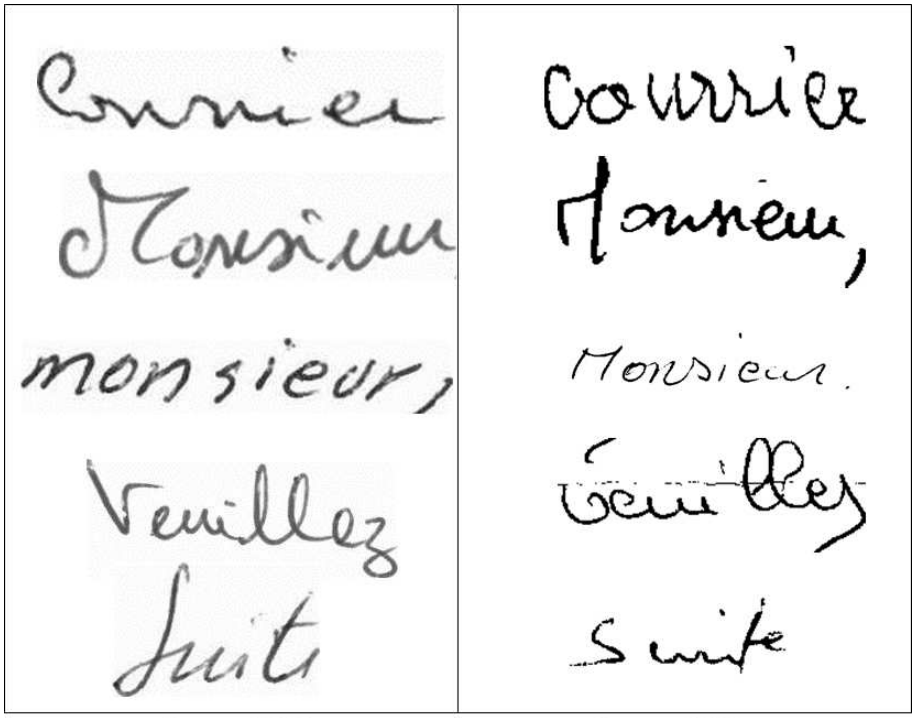

Figure 4. Example of 2 equivalent word lists from DB1 and DB2. Word distribution is the same in each list.

3.2 Recognition system

The recognition system used for the experiments is a context-independent HMM-based system as described in.[5] It uses the HTK tool for Markovian modelling ("Hidden Markov Models Toolkit"), developed at Cambridge University Engineering Department.[12] The system is segmentation-free, based on the sliding-window approach. A word is represented by a concatenation of character models.

We use the existing system developed in El-Hajj[4] and Bianne-Bernard[5] for grey-level images. The input of the system is a sequence of windows (frames), divided into a fixed number of subsets of pixels (cells). A set of 37 features is extracted for each frame. We extract geometrical, statistical, and directional features. Specifically, our feature set includes:

- 2 features related to foreground/background transitions
- 12 features related to concavity configurations
- 3 features related to the position of the gravity center: the first one gives the position related to the baselines (above the upper baseline, below the lower baseline, between the two baselines), the second one gives the position in number of pixels related to the lower baseline, and the last one the difference between the gravity centers of two neighbouring windows.
- $w = 9$ features corresponding to the density of pixels in each column (w is the width of the window)
- 3 features related to the density of pixels: global density in the sliding window, above and below the baselines.
- 8 directional features corresponding to histogram of gradient for the 8 orientations 0, $\pi/4$, $\pi/2$, $3\pi/4$, .., 2π.

We add dynamic information to the feature vector by including their first-order derivatives. We use the same parameter set up for the Rimes 2011: the overlap between two consecutive frames is $\delta = 3$; number of cells = 20; number of states = 12 for each letter and 4 for symbols such as (, -, /)). The likelihood of each state is modeled using a Gaussian Mixture Model, with either 20 or 50 components as suggested in.[5]

Overall, a number of 79 models for all the characters of the French alphabet were considered: 26 models for lowercase letters, 26 models for uppercase letters, 13 models for accentuated letters from French alphabet (à, À, ç, è, é, ê, Ê, É, ï, î, ô, û,etc.), 9 models for digits (because 0 and O share the same model), 4 for punctuation signs (-, ', /,) and one model for space. Viterbi algorithm is used for decoding.

3.3 Experimental results

We present here our experiments conducted on the two databases DB1 and DB2. Our performance metric is word accuracy, computed as:

$$Accuracy = \frac{C}{N} \tag{2}$$

where C is the number of the words correctly recognized and N is the number of words in the test list. The metric is accent sensitive and case insensitive.

3.3.1 No adaptation

In this experiment, we perform no database adaptation. We consider 10,000 word images from DB1 (Rimes 2011 database) to train the system. We consider 2,500 test images from DB2. These 2,500 word images were randomly chosen from DB2 database, with the condition that they exist in the validation Rimes database. Word distribution is equal in validation and test sets (see Figure 4). Performance comparison is thus fair between these *equivalent* sets (wrt word distribution). Results are shown in Table 1.

First line of Table 1 shows recognition performance on a validation set extracted from DB1 (2,500 images). This provides an optimistic bound on the recognition performance, since the test is carried out on images which have the same characteristics as the ones used to train the system. The accuracy of the system is 73,92 % since the system is trained on a restricted set of 10,000 word images. When tested on novel, irregular data (2,500 images from DB2), this accuracy decreases to 53%, showing that the system as it is built cannot handle properly the variability of data in DB2.

We repeat the experiment using all DB1 data for training (51,738 word images). In this case, classification accuracy is expected to be higher since a larger number (almost 5 times) of training examples are available. On a larger training set, more Gaussians per state can be learnt. Table 1 shows that 50 Gaussians per state improves accuracy when the training set is sufficiently large (51,738 images).

In the next Section, we show instead that the two dataset adaptation techniques described in Sections 2.1 and 2.2 can increase classification performance.

3.3.2 Test set adaptation

In this experiment, we evaluate how test set adaptation can improve classification performance. We consider a training set composed of 10,000 word images from DB1, whereas the test set consists of DB2 2,500 adapted word images (stroke thickness normalization). The 2,500 word images were randomly chosen from the DB2 database with the condition that they exist in the validation Rimes database.

We set the size and the standard deviation σ of Gaussian filter in stroke thickness normalization to 400×400 and 0.5, respectively. This choice of parameters is based on preliminary tests on a small dataset.

As we can see in Table 1, the effect of normalizing the test database to the average stroke thickness of the train database improves results, as classification accuracy increases from 53.48% to 55.60% for a training set of 10,000 word images. This performance gain is not negligible considering that DB2 variability is much higher than DB1, while we are focusing only on one specific difference between the two sets, namely stroke thickness. Including other sources of variability (e.g., resolution, noise, slope, etc.) is supposed to further increase classification performance. This is left to future work.

Train DB	Test DB	Adaptation	Recognition Accuracy
DB1 (10,000 images)	DB1 (2,500 images)	no adaptation	73.92 % (20 Gaussians)
DB1 (10,000 images)	DB2 (2,500 images)	no adaptation	53.48 % (20 Gaussians)
DB1 (10,000 images)	DB2 (2,500 images)	test set adaptation	55.60 % (20 Gaussians)
DB1 (10,000 images)	DB2 (2,500 images)	training set adaptation	56.24 % (20 Gaussians)
DB1 (51,738 images)	DB2 (2,500 images)	no adaptation	61.12 % (20 Gaussians)
DB1 (51,738 images)	DB2 (2,500 images)	no adaptation	62.04 % (50 Gaussians)
DB1 (51,738 images)	DB2 (2,500 images)	test set adaptation	61.92 % (20 Gaussians)
DB1 (51,738 images)	DB2 (2,500 images)	train set adaptaton	63.28 % (20 Gaussians)
DB1 (51,738 images)	DB2 (2,500 images)	test set adaptation	63.08 % (50 Gaussians)
DB1 (51,738 images)	DB2 (2,500 images)	train set adaptation	65.16 % (50 Gaussians)

Table 1. Recognition accuracy for test set (DB2) adaptation, training set (DB1) adaptation according to the amount of training data. Comparison with recognition accuracy on validation set (DB1) and no adaptation on validation set (DB1 and DB2).

3.3.3 Training set adaptation

This experiment is conducted in order to analyse the effect of adapting the training database to the test database. For the first part of the experiment, training set consists of a subset of the same 10,000 word images as in the previous experiment, but the training samples themselves are adapted using the algorithm described in Section 2.2. The test set contains 2,500 word images from DB2, transformed into gray-scale images in order to perform feature extraction. In order to compare the results obtained with test set adaptation, we pick the same 2,500 word images from DB2 as described in Sections 3.3.1 and 3.3.2.

In the training set adaptation (stroke intensity normalization), the parameters to be set are related to the Sauvola binarization algorithm.[10] They have been set as follows: 128 for the dynamic range R; 0.1 for the sensitivity value k; 25 for the size of the neighbouring window. After binarization with Sauvola, Gaussian smoothing is applied with the same parameters as described in the previous section. The square structuring element for morphological closure is of size 3×3 . These parameters were obtained by maximizing classification performance on a validation set extracted from DB2, which has no overlap with the 2,500 image words of the test set. In Table 1 we also report the results of experiments when all DB1 data (51,738 images) are used for training.

Results show that train set adaptation performs slightly better than test set adaptation. This is mainly due to the fact that in training set adaptation, the recognition system is re-trained on adapted data, differently from test set adaptation which does not require a new training phase. However, this re-training has a cost in terms of computational time — pre-processing, feature extraction and HMM training have to be performed from scratch. Therefore in practice, test set adaptation could be a viable solution to reduce the computational cost of dataset adaptation.

4. CONCLUSIONS

A common challenge in practical off-line word recognition is to cope with the variability of real-world data. In this paper, we propose a dataset adaptation approach to increase the similarity between the available training data and the novel test data. Specifically, we proposed two methods, *training set adaptation* and *test set adaptation*, which change the training or the test data, respectively. In the case of training set adaptation, we achieve better performance, but at the cost of a second training stage on adapted data, which could be time consuming. On the other hand, test set adaptation enables us to increase classification performance without increasing considerably the computational cost of the adapted system. Although we provide only a proof of concept of the potentiality of dataset adaptation, we believe that: 1) the data set adaptation techniques can be generalized to other types of datasets than the ones used in this paper; and 2) the performance gain could be further increased by considering other type of word variability (e.g., resolution, noise, etc.). 3) As mentioned in the introduction (Section 1), another strategy would be to add real-world data to the clean/regular training set. To avoid time-consuming

manual labeling, one could use automatically labeled data as in.[13] We plan to investigate these aspects in future work.

Acknowledgements

The authors would like to thank Professor Elisa Barney Smith for kindly providing the source code of the Sauvola algorithm.[10]

REFERENCES

1. N. Gorski, V. Anisimov, E. Augustin, O. Baret, D. Price, and J. Simon, "A2ia check reader: A family of bank check recognition systems," in *Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on*, pp. 523–526, IEEE, 1999.
2. D. D'Amato, E. Kuebert, and A. Lawson, "Results from a performance evaluation of handwritten address recognition systems for the united states postal service," in *Proceedings of 7th Int. workshop on Frontiers in Handwriting Recognition, Amsterdam*, pp. 189–198, 2000.
3. E. Grosicki and H. El-Abed, "ICDAR 2011: French handwriting recognition competition," in *ICDAR*, 2011.
4. R. Al-Hajj-Mohamad, L. Likforman-Sulem, and C. Mokbel, "Arabic handwriting recognition using baseline dependent features and hidden markov modeling," in *Proceedings of the Eighth International Conference on Document Analysis and Recognition - ICDAR05*, pp. 893–897, 2005.
5. A.-L. Bianne-Bernard, F. Menasri, R. El-Hajj, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem, "Dynamic and contextual information in HMM modeling for handwritten word recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **99**(10), pp. 2066–2080, 2011.
6. C. Leggetter and P. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer speech and language* **9**(2), p. 171, 1995.
7. J. Gauvain and C. Lee, "Map estimation of continuous density hmm: theory and applications," in *Proceedings of the workshop on Speech and Natural Language*, pp. 185–190, Association for Computational Linguistics, 1992.
8. A. Vinciarelli, "A survey on off-line cursive word recognition," *Pattern recognition* **35**(7), pp. 1433–1446, 2002.
9. A.-L. Bianne-Bernard, *Reconnaissance de mots manuscrits cursifs par modèles de Markov cachés en contexte: application au français, à l'anglais et à l'arabe*. PhD thesis, Télécom ParisTech, 2011.
10. J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition* **33**(2), pp. 225–236, 2000.
11. E. Grosicki and H. El-Abed, "Icdar 2011-french handwriting recognition competition," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp. 1459–1463, IEEE, 2011.
12. S. Young, "The htk hidden markov model toolkit: Design and philosophy," *Department of Engineering, Cambridge University, UK, Tech. Rep. TR* **153**, 1993.
13. V. Frinken and H. Bunke, "Self-training strategies for handwriting word recognition," in *ICDM*, pp. 291–300, 2009.