# Training Set Expansion in Handwritten Character Recognition

Javier Cano, Juan-Carlos Perez-Cortes, Joaquim Arlandis, and Rafael Llobet

Instituto Tecnologico de Informatica, Universidad Politecnica de Valencia ⋆
Camino de Vera, s/n 46071 Valencia (SPAIN)
{jcano,jcperez,jarlandi,rllobet}@iti.upv.es

**Abstract.** In this paper, a process of expansion of the training set by synthetic generation of handwritten uppercase letters via deformations of natural images is tested in combination with an approximate $k-$Nearest Neighbor ($k-$NN) classifier. It has been previously shown [11] [10] that approximate nearest neighbors search in large databases can be successfully used in an OCR task, and that significant performance improvements can be consistently obtained by simply increasing the size of the training set. In this work, extensive experiments adding distorted characters to the training set are performed, and the results are compared to directly adding new natural samples to the set of prototypes.

## 1 Introduction

The $k-$NN classifier has received renewed attention in the last years and is being successfully used in many pattern recognition tasks, in particular in Handwritten Character Recognition [11] [10]. The $k$ Nearest Neighbors Rule is a classical statistical method which offers consistently good results, ease of use and certain theoretical properties related to the expected error. Being a memory-based technique that does not build a reduced model of the data but stores every prototype to compare it to the test observations, it can benefit from very large sets of training data. In [11], for example, consistent performance improvements of a $k$-NN classifier in a character recognition task are reported when the original training-set size is increased. The practical use of large databases is being increasingly allowed by advances in hardware speed and memory capacity.

In fact, the spatial cost of storing all the training vectors is often not a problem for current computers even in the case of huge data-sets. The temporal cost, however, can be a limiting factor in many cases. This problem can be approached from two points of view: trying to reduce the number of prototypes without degrading the classification power, or using a fast nearest neighbors search algorithm.

The first approach has been widely studied in the literature. Editing [12], [3], condensing [6], and their multiple variations are well known representatives. These methods have shown good results equaling or sometimes improving the

⋆ Work partially supported by the Spanish CICYT under grant TIC2000-1703-CO3-01

classification rates of the $k-$NN rule. Their power resides in the smoother discrimination surfaces they yield by eliminating redundant and noisy prototypes. Cleaner discrimination surfaces reduce the risk of over-training. In a pure $k-$NN classifier with a large database, an adequate choice of $k$ provides the same effect. The best value of $k$ often grows with the size of the database.

The second approach has also been extensively analyzed and many techniques have been proposed to reduce the cost of the exhaustive search of the prototypes set to find the $k$ nearest neighbors to a test point. Most of them make use of $kd-$trees [4], [2], [8] or similar data structures. A $kd-$tree is a binary tree where each node represents a region in a $k-$dimensional space. Each internal node also contains a hyper-plane (a linear subspace of dimension $k-1$) dividing the region into two disjoint sub-regions, each inherited by one of its sons. Most of the trees used in the context of our problem divide the regions according to the points that lay in them. This way, the hierarchical partition of the space can either be carried out to the last consequences to obtain, in the leaves, regions with a single point in them, or can be halted in a previous level so as each leaf node holds $b$ points in its region.

In a $kd-$tree, the search of the nearest neighbor of a test point is performed starting from the root, which represents the whole space, and choosing at each node the sub-tree that represents the region of the space containing the test point. When a leaf is reached, an exhaustive search of the $b$ prototypes residing in the associated region is performed. But the process is not complete at this point, since it is possible that among the regions defined by the initial partition, the one containing the test point does not contain the nearest prototype. If this can happen, the algorithm backtracks as many times as necessary until it is sure to have checked all the regions that can hold a prototype nearer to the test point than the nearest one in the original region. The resulting procedure can be seen as a Branch-and-Bound algorithm.

In most pattern recognition applications, there is no need to guarantee that the exact nearest neighbors of a test observation are found. Fast approximate nearest neighbors search algorithms allow for efficient classification without losing significant classification performance.

In a $kd-$tree, if a guaranteed exact solution is not needed, the backtracking process can be aborted as soon as a certain criterion is met by the current best solution. In [1], the concept of $(1+\epsilon)-$approximate nearest neighbor query is introduced. A point $p$ is a $(1+\epsilon)-$approximate nearest neighbor of $q$ if the distance from $p$ to $q$ is less than $1+\epsilon$ times the distance from $p$ to its nearest neighbor. An algorithm based on these concepts have been used on conventional $kd-$trees in the experiments.

The rest of the paper is organized as follows: Section 2 describes the parameterization, classification and the database expansion procedures proposed. In Section 3, the data and the deformation patterns used are presented. In Section 4, extensive empirical results are reported, and the conclusions are presented in Section 5.

## 2 Parameterization and Character Classification

The preprocessing and feature extraction methods used are simple and can be performed easily and quickly. The character images were first sub-sampled from their original 128x128 binary pixels into 14x14 gray value representations by first computing the minimum inclusion box of each character, keeping the original aspect ratio, and then accumulating into each of the 14x14 divisions the relative area taken by black pixels, to obtain a continuous value between 0 and 1. Other grid sizes from 8x10 to 16x16 have been tested in this task with similar results. Principal Component Analysis (or Karhunen-Loeve Transform) was then performed on the image representations to reduce their dimensionality to 45, the value which showed the best results in preliminary experiments.

The $k-$NN classifier with the Euclidean Distance in the 45-dimensional space described above has been used in the experiments, employing a fast $(1+\epsilon)-$approximate search with $\epsilon = 2$ in every case. This was the largest value for $\epsilon$ empirically found to keep, in our task, almost the same error rates as an exact search. The temporal cost of the classification grows approximately with the logarithm of the size of the training set. Given this slow increase in the search times, an interesting approach to improve the accuracy, keeping at the same time high recognition speeds, is to insert new prototypes into the training set. Of course, making larger the original database is an evident way to do it, but the manual or semi-automated segmentation and labeling procedures needed to build a good, large database are very time-consuming. Therefore, a possible approach to exploit the information of a given database as much as possible is to perform controlled deformations on the characters to insert them into a new larger training set.

Improving a handwritten character recognizer using appropiately distorted samples has been proposed in [5] with good results. In [10], expanding the training set by including the distorted characters is proposed as a faster and cleaner option to include the distorted characters in the training set instead of distorting the test character in several ways and carrying out the classification of each deformed pattern. In this work, we perform extensive experimentation to validate that approach.

## 3 Databases and Transformations

Two different sets of databases have been employed to perform the experiments. One is public and has been widely used in the literature, and the other one was a locally acquired image database that was used only for one experiment where a larger training set was required.

The first image databases are the well-known NIST Special Database 3, for training, and Special Database 7 for test. Only the upper-case images have been used, with a total of 44951 images for training set and 11941 for test set.

The other database is a locally acquired set of images with slightly over 1 million upper-case letters acquired from forms written by different writers.

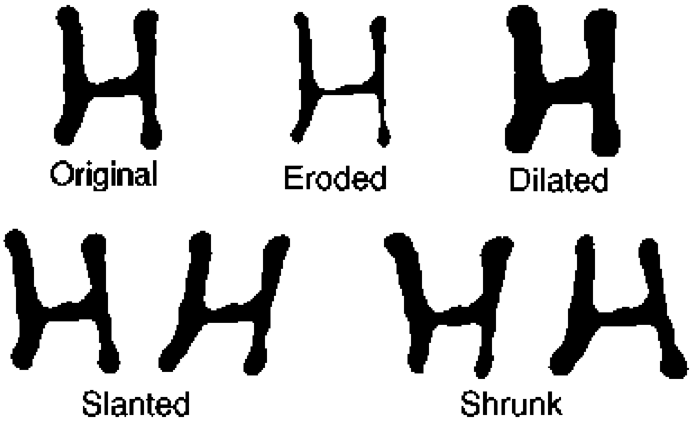

**Fig. 1.** Families of transformations tested

A database of this size was required to compare the performance of two systems trained with increasingly large sets composed by the same initial core and expanded with real and distorted images respectively.

Four simple kinds of image transformations have been tested (see Figure 1): slant and shrink, to cope with geometric distortions of the writing, and erosion and dilation to account for different writing implements, acquisition conditions, etc. [9] [7]. The transformations were first tested separately and then the one offering the best results (slant) was applied first, expanding the training set so that the rest of transformations were incrementally applied to the original plus the slanted characters. The details are presented in Section 4

All the transformations were applied to the original binary images before scaling them to 14x14 pixels. The slant transformation consisted of right- or left-shifting each row an integer number of positions. The 4 different slants performed were equivalent to rotations of a vertical segment by $-26^o$, $-9^o$, $9^o$, and $26^o$. The central row was never shifted, and the amount of shift increased linearly from there to the top and bottom rows, in opposite directions. The new pixels entering the area due to the shifts were set to white.

The shrink transformation consisted of two proportional scalings of the image, keeping the top or bottom line at its original size and then gradually scaling each line until the opposite line is reached and scaled to a fixed smaller size (50%). In the case of erosion and dilation, the classical binary morphological operations were applied by one pixel.

## 4 Experiments

The first experiment was targeted to determine the value of each of the different distortions, used alone, to improve the accuracy of a $k-$NN classifier. The results,

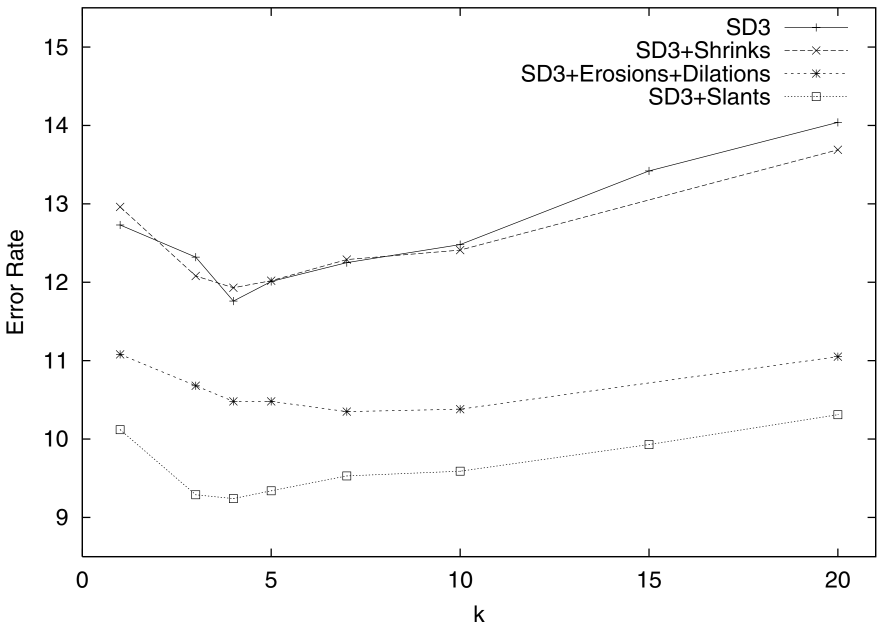

**Fig. 2.** Error rates of a $k-$NN classifier for a range of values of $k$ and different training sets composed of real and synthetic images of the NIST database

using a holdout estimation, for the 44951 upper case letters from SD3 as the training set and the 11941 images of SD7 as the test set are shown in Figure 2. The insertion of artificially slanted, eroded and dilated images to the training set produced significant improvements. However, the inclusion of shrunk images did not improve the results. Therefore, this last transformation has not been included in the subsequent experiments. The best value of $k$ was 4 in all cases but one.

In the second experiment, with the same training and test data, the goal was to expand the training set in the best possible way to improve the classification accuracy. To do that, the best transformation, slant, was first used to expand the core of original images, obtaining 224755 images $(44951 + 4 \times 44951)$. Then, the next best transformations, erosion and dilation, have been sequentially applied to the previous set obtaining training sets of 449160 $(224755 \times 2)$ and 673740 $(449160 + 224755)$ respectively. The results are shown in Figure 3.

Another effect that can be noticed in the results of these experiments is that the best value of $k$ seems to gradually increase as the database grows, which is in accordance with what could be expected.

To find out if the accuracy improvement achieved by artificially expanding a core database of images is comparable to using extra real data, another experiment was carried out. In this case, a large locally acquired real database was used, allowing increasingly large training sets up to 674265 images, equivalent to

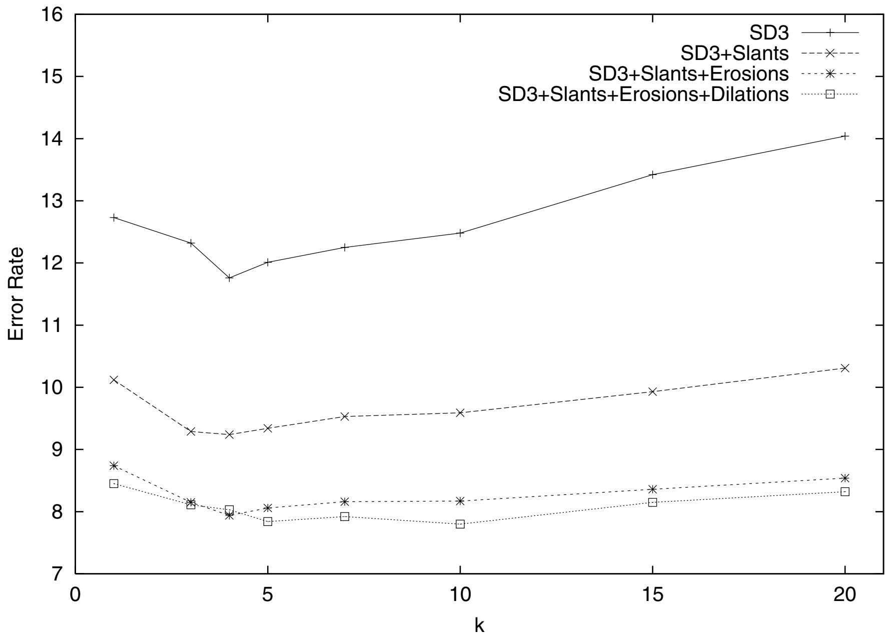

**Fig. 3.** Error rates of a $k-$NN classifier for a range of values of $k$. The original data set was incrementally expanded using deformed images of the NIST database

the size of SD3 expanded with all the synthetic images generated in the previous experiments.

A core of 44951 images (the same size of SD3) from this local database was randomly selected. On the one hand, this core set was made larger by adding new real images from the rest of the local database, and on the other hand, the proposed expansion of the core using deformations was applied. In Figure 4, the results of this experiment on the local database using a $k-$NN classifier are shown. The value of $k$ used was 4.

From this figure we can see that a reduction from 11.77% to 7.77% on the error rate has been achieved in the experiments with NIST SD3/SD7 by adding synthetic images to the training set. Similarly, an error-rate reduction from 4.08% to 2.85% has been achieved in the experiments with the local database. But an even higher reduction of the error rate, from 4.08% to 1.88%, has been found in the local database when the expansion of the training set is performed using additional real data, instead of deformed images.

These results suggest that, with the synthetic image generation methods employed, it is preferable to make use of extra real data. Of course, this is only possible if a very large database is available.

In Figure 5, the results of the largest real and synthetic local databases, both of similar size, around 675.000 images, are plotted for a range of values of $k$.

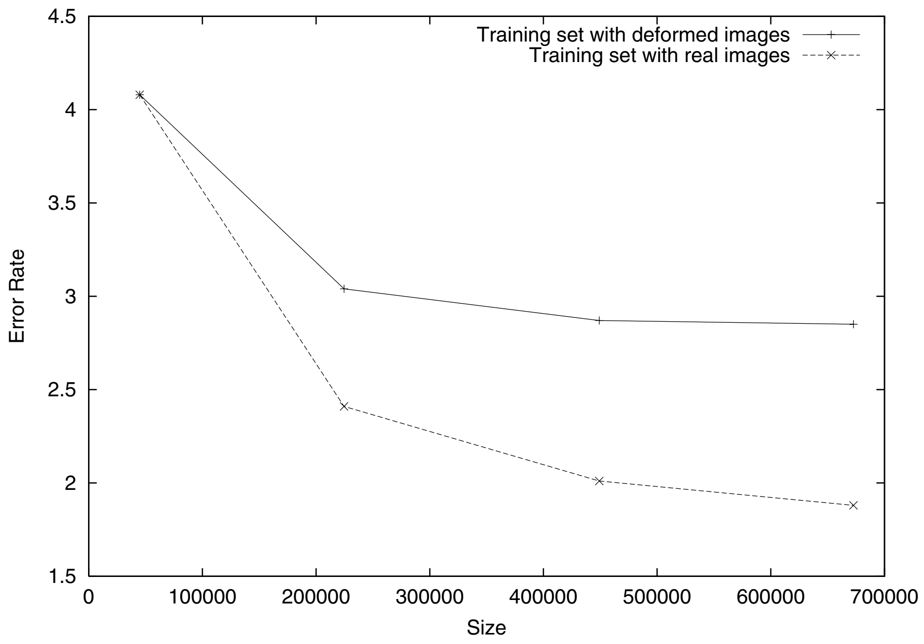

**Fig. 4.** Comparison of error rates of a $k-$NN classifier for increasingly large training sets composed of real-only and real+synthetic images from the local database

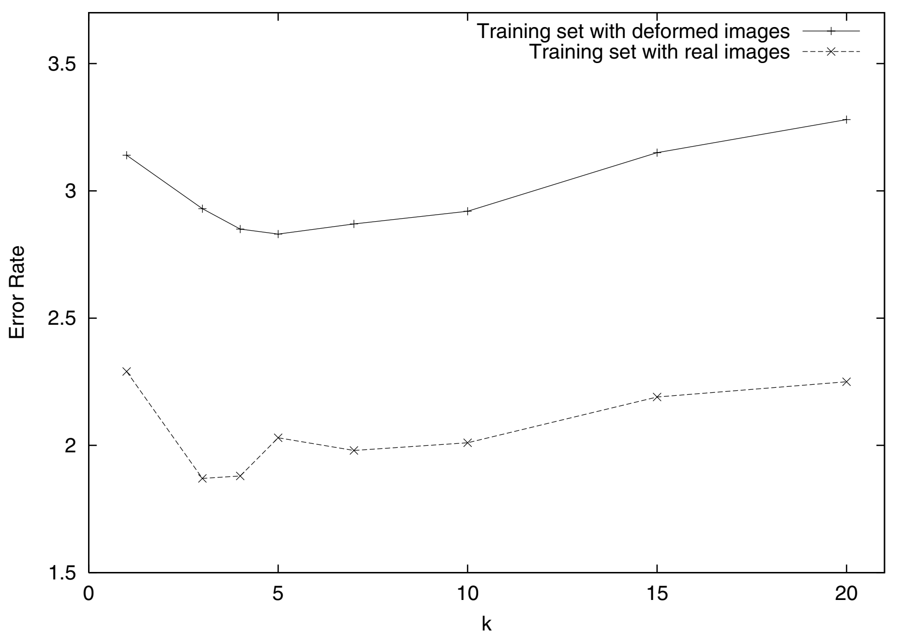

**Fig. 5.** Error rates of a $k-$NN classifier for several values of $k$. A training set comprising the full local database (real images) is compared to one composed by a core subset of real images plus those images deformed in various ways

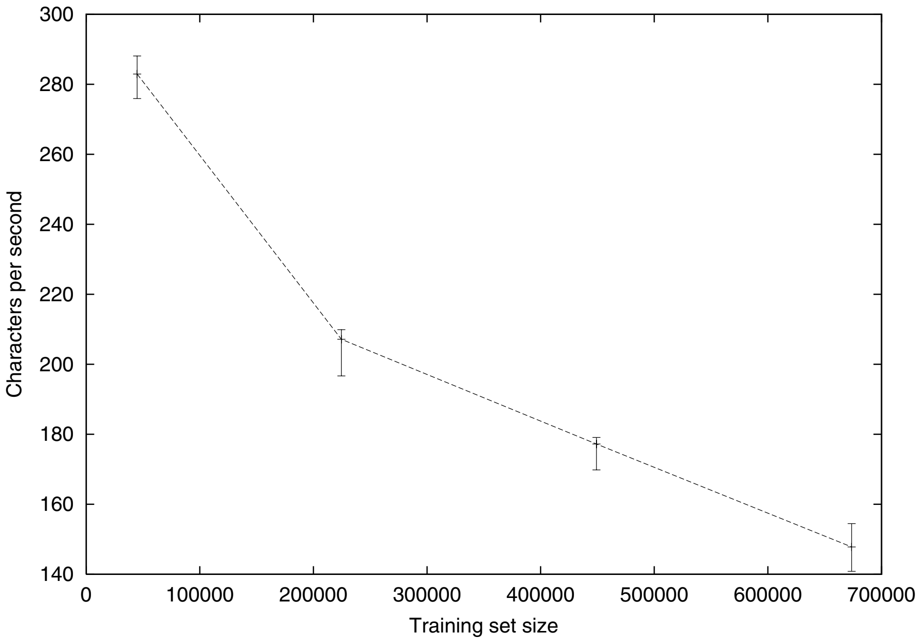

**Fig. 6.** Recognition speed against traininig set size with real data from the local database in an AMD Athlon PC at 1.2 Ghz. The average, maximum and minimum of 10 experiments are shown for each point

As can be seen from Figure 6, the improvements obtained when the database is expanded only incur a slight decrease on the average recognition speed.

## 5 Conclusions

In this work, large image databases stored in $kd-$tree structures and approximate nearest neigbour search have been used to obtain competitive error rates and recognition speed.

A recognition rate improvement of 4.0% has been achieved in the upper-case letters classification problem with the NIST-SD3/SD7 (Figure 3), and a 2.20% improvement in the local database (Figure 4) has been achieved by increasing the database size fifteen times (from 44951 original images to a final set of 674265 images).

However, this training set size increase does not seriously affect the processing time requirements of the recognition method. For instance, with the original training set, a search is performed in 3.6 ms (280 cps) with an increase of about 3 ms (to 150 cps) for the whole training set with 674265 images in a computer with an AMD Athlon processor and 1.2 Ghz clock frequency.

A comparison between artificial database expansion and real enlargement of the training set has also been performed. The results suggest that, although both approaches provide significant improvements, the latter is clearly preferable.

## References

1. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998. 549
2. J. L. Bentley, B. W. Weide, and A. C. Yao. Optimal expected time algorithms for closest point problems. *ACM Trans. on Math. Software*, 6:563–580, 1980. 549
3. P. A. Devijver and J. Kittler. On the edited nearest neighbour rule. In *Proceedings of the 5th International Conference on Pattern Recognition*, pages 72–80. IEEE Computer Society Press, Los Alamitos, CA, 1980. 548
4. J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3:209–226, 1977. 549
5. T. M. Ha and H. Bunke. Off-line, handwritten numeral recognition by perturbation method. *IEEE Trans. on PAMI*, 19(5):535–539, May 1997. 550
6. P. E. Hart. The condensed nearest neighbor rule. *IEEE Trans. on Information Theory*, 125:515–516, 1968. 548
7. A. Jain. Object matching using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:268–273, 1996. 551
8. B. S. Kim and S. B. Park. A fast $k$ nearest neighbor finding algorithm based on the ordered partition. *IEEE Trans. on PAMI*, 8:761–766, 1986. 549
9. Jianchang Mao. Improving ocr performance using character degradation models and boosting algorithm. *Pattern Recognition Letters*, 18:1415–1419, 1997. 551
10. J. C. Perez-Cortes, J. Arlandis Arlandis, and R. Llobet. Fast and accurate handwritten character recognition using approximate nearest neighbours search on large databases. In *Workshop on Statistical Pattern Recognition SPR-2000*, Alicante (Spain), 2000. 548, 550
11. S. J. Smith. Handwritten character classification using nearest neighbor in large databases. *IEEE Trans. on PAMI*, 16(9):915–919, September 1994. 548
12. D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. on Systems, Man and Cybernetics*, 2:408–420, 1972. 548