[[1]](#footnote-2) Comparing Arabic and Latin numerical recognition

|  |
| --- |
| Sherif Abdelazeem†  *[shazeem@aucegypt.edu](mailto:shazeem@aucegypt.edu)* |
| *Electronics Engineering Dept., The American University in Cairo* |

Abstract

*The objective of this study is to compare the performances of different classification techniques on both Arabic and Latin (English) digit recognition problems. This comparison gives insights into the difference between the natures of the two problems.*

# **INTRODUCTION**

Handwritten digit recognition problem can be seen as a sub-task of the more general Optical Character Recognition (OCR) problem. However, there are some applications (e.g., postal code and bank checks reading) that are restricted to recognizing digits but require very high accuracy and speed. In addition, handwritten digit recognition problem is usually used as a benchmark for comparing different classification techniques [1].

While recognition of handwritten Latin digits has been extensively investigated using various techniques [1-8], too little work has been done on Arabic digits. This is most probably because there is no large free online database for Arabic digits like that of Latin (e.g. MNIST).

In a previous paper [9], we have taken a step to encourage doing more research on Arabic digits by building a large dataset: the Arabic Digits dataBase (ADBase) which is composed of 60,000 digits for training and 10,000 for testing written by 700 different writers.

This paper will compare between performances of different classification techniques upon both Arabic and Latin digit recognition problems (on MADBase and MNIST, respectively). It is well known that there is no universally powerful or universally weak classifier [10]. The performance of a certain classification technique depends on how it matches the problem at hand. Realizing this, comprehensive comparative studies [11] compare the performances of classifiers upon many different problems. However, since researchers doing such comparative studies do not have a direct experience with the problems they use, they do not give reasons why a certain classification technique performs poorly on a certain problem while having a good performance on another. However, in our case, we have an experience with the two problems used in the comparative study: the Arabic and Latin digit recognition. This gives us the chance to interpret the comparative study results and gain more insight into the natures of the two problems.

The remaining of the paper is organized as follows. Section 2 is a survey of previous research on Arabic handwritten digit recognition problem. The classifiers and databases used in the comparison are described in section 3. Section 4 reports and analysis the performances of the classification techniques on both the Latin digits (on MNIST) and Arabic digits (on MADbase). In section 5 we conclude.

# **Previous Work**

TABLE 1  
Arabic Printed and Handwritten Digits

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Latin Equivalent | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Printed |  |  |  |  |  |  |  |  |  |  |
| Typical Handwritten |  |  |  |  |  |  |  |  |  |  |
| Other Writing Style | -- | -- | -- |  | -- | -- | -- | -- | -- | -- |

Naming conventions for different numeral systems may be confusing. Digits used in Europe and several other countries sometimes are called “Arabic Numbers”; and digits used in Arab world are sometimes called “Hindi Numbers”. A different naming convention is used in this paper. Digits used in Europe will be referred to as “Latin Digits” and that used in Arab world as “Arabic Digits”.

It is worthwhile to mention here that Arabic and Persian handwritten digits (digits used in Iran) are similar but not identical. However, there are some writing styles for Persian digits that are very similar to Arabic. Tables 1 and 2 show Arabic and Persian handwritten digits with different writing styles as well as their printed versions.

TABLE 2  
Persian Printed and Handwritten Digits

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Latin Equivalent | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Printed |  |  |  |  |  |  |  |  |  |  |
| Typical Handwritten |  |  |  |  |  |  |  |  |  |  |
| Other Writing Style |  | -- |  |  |  |  |  | -- | -- | -- |

Mozaffari et al. [12] extract structural information form Persian digits, which in turn are used to extract statistical information describing the digits. They used then a nearest neighbor 4 classifier to get an accuracy of 94.44%. Soltanzadeh et al. [13] calculate number of features in multiple orientations of the digit images. They presented these features to an SVM to achieve an accuracy of 99.57%. Al-Omari et al. [14] proposed a system for recognizing Arabic digits from ‘1’ to ‘9’. They used a scale-, translation-, rotation-invariant feature vector to train a probabilistic neural network (PNN). Their database was composed of 720 digits for training and 480 digits for testing written by 120 persons. They achieved 99.75% accuracy. Said et al. [15] used pixel values of the 16×20 size-normalized digit images as features. They fed these values to an Artificial Neural Network (ANN) where number of its hidden units is determined dynamically. They used a training set of 2400 digits and a testing set of 200 digits written by 20 persons to achieve 94% accuracy. Note that results of different works can not be compared because the used databases do not have the same sizes or formats.

In a previous paper [9], we devised a two-stage system for recognizing Arabic digits. The first stage is an ANN fed with a short but powerful feature vector to handle easy-to-classify digits. Ambiguous digits are rejected to the more powerful second stage which is an SVM fed with a long feature vector. The system had a good timing performance and achieved 99.15% accuracy on the ADBase.

As mentioned before, one of the reasons why there is little research in Arabic digit recognition is that there is no free online Arabic digits database similar to that available for Latin digits. Digits databases collected in some other works are of small sizes and limited accessibility [16,17]. In the following section, the databases and classifiers are going to be introduced.

# **3. Benchmark Results**

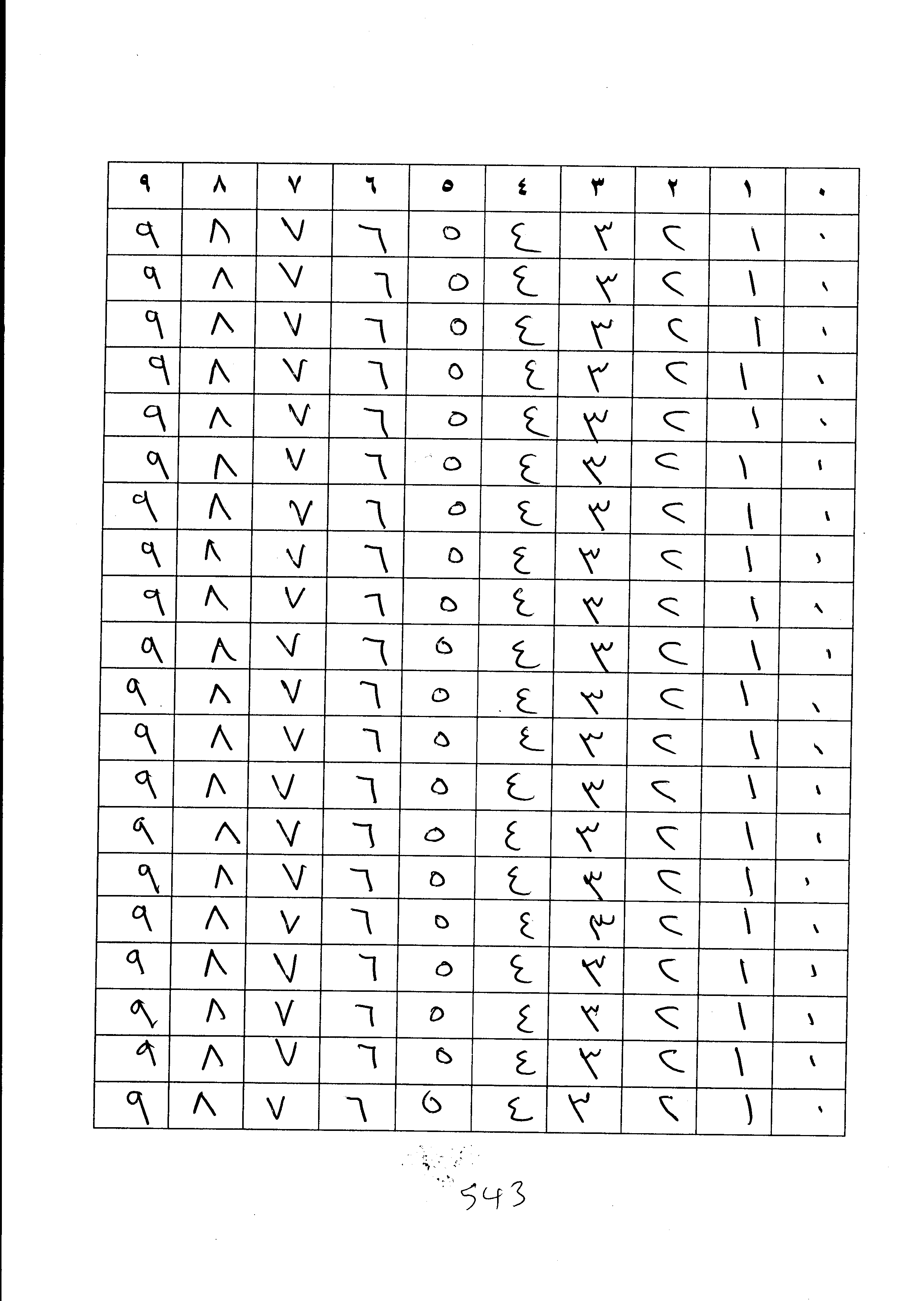
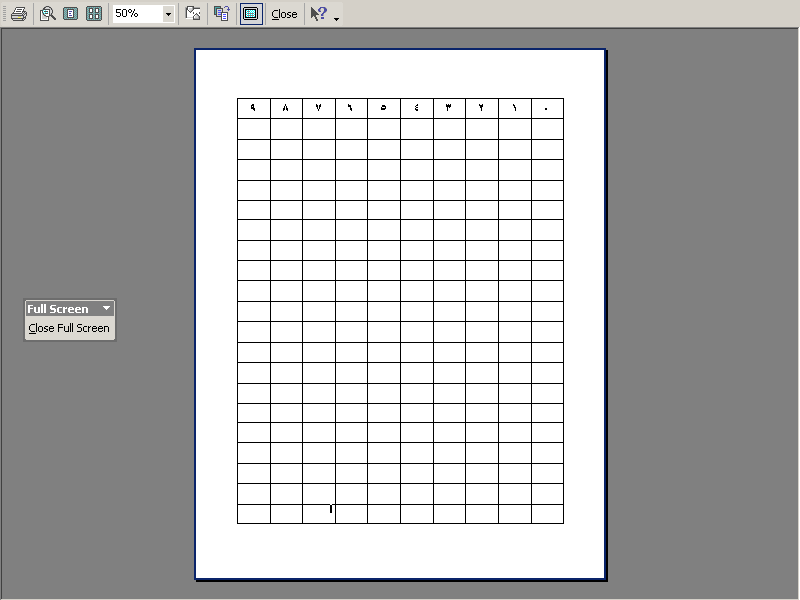
In this section, we are going to report the performances of 10 different classification techniques on MNIST and MADBase to use in comparison between both databases. Each of these 10 classifiers will be fed directly with gray-scale pixel values of digit images without feature extraction step. Some of these classifiers have parameters which need to be specified (e.g., the number of hidden units of ANN). Such parameters are selected so as to give the best performance on a validation set; which is a set of 10,000 digits selected randomly from the training set.

## **The Database**

The objective of this paper is to compare the performances of different classification techniques on Arabic digits with their performances on Latin. To make such a comparison valid, the two databases of Arabic and Latin digits should be of the same format. Since we have chosen the MNIST to be the used Latin digits database, a Modified version of the ADBase (MADBase) that has the same size and format of MNIST has been created. The MADBase is created from ADBase as follows. For each digit of ADBase, its height (h) and width (w) are calculated, and then the digit is size-normalized to have a new height (hnew) and new width (wnew). The assigned values of hnew and wnew depend on whether h or w is greater than the other. If h>w, then hnew is set to 20, and w to floor(20×w/h). If w>h, then wnew is set to 20 and hnew to floor(20×h/w). This procedure ensures that each digit of MADBase is confined in a 20×20 box while its aspect ration is preserved. Note here that the resulting size-normalized digits have gray levels as a result of the anti-aliasing filter used in size-normalization procedure. Then each digit is placed in a 28×28 white background such that its center of gravity coincides with the center of the white background.

The MADBase now has the same size and format of MNIST. Actually, MNIST is a modified version of the digits database NIST as MADBase is modified from ADBase. More about how MNIST is modified from NIST can be found in [1]. Samples of 10 digits from MNIST and their corresponding digits from MADBase are shown in Fig. 2(a) and 2(b), respectively.

Both the MADBase is available for free at <http://datacenter.aucegypt.edu/shazeem/>.



(a) (b)

Fig. 1. (a) An empty form, and (b) a filled form

# **Comparing the Performances of Different Classifiers on Arabic Digits To that of Latin**

The goal of this paper is to compare between the performances of different classifiers on Arabic digits to their performances on Latin digits. This may give us insights into the natures of the two problems. In the previous section, the MADBase and MNIST . Results are summarized in table 4. We may compare between performances of different classifiers on Arabic and Latin digits by comparing tables 3 and 4. To better visualize accuracy comparison, see Fig. 3.

TABLE 4  
Performances of different classification techniques on MNIST

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifier** | **Acc**  **(%)** | **RR**  **(%)** | **T** |
| OVA Linear | 92.35 | 41.34 | 1 |
| OVO Linear | 94.25 | 43.83 | 5 |
| OVO-lin-SVM | 94.76 | 27.61 | 5 |
| KNN, K=3 | 97.05 | 15.33 | 7100 |
| KNN, K=5 | 96.88 | 12.47 | 7100 |
| Parzen | 96.93 | 12.4 | 8150 |
| PCA+Quadratic | 97.75 | 7.72 | 8 |
| PCA+ANN | 97.88 | 5.27 | 10 |
| ANN | 97.91 | 4.8 | 46 |
| OVO-rbf-SVM | 98.5 | 2.93 | 2120 |
| LeNet | 98.9 | 1.76 | 90 |



(a)

(a)



(b)

Fig. 2. Sample digits from (a) MNIST, and (b) MADBase

In this section, we are going to report the performances of 10 different classification techniques on MADBase to serve as benchmark results for future research on Arabic digit recognition. Each of these 10 classifiers will be fed directly with gray-scale pixel values of digit images without feature extraction step. Some of these classifiers have parameters which need to be specified (e.g., the number of hidden units of ANN). Such parameters are selected so as to give the best performance on a validation set; which is a set of 10,000 digits selected randomly from the training set.The performances of many of the classifiers used in this study have been evaluated on MNIST previously in [1]. However, their performances are re-evaluated in our study using the same classifiers implementations used with MADBase. This ensures the validity of the comparison made between performances of different classifiers on MNIST and MADBase. Classification techniques used in this study are implemented in C++ using Torch machine learning library [28].

The classifiers configurations and parameter setting strategies used with Latin digits are the same as that used with Arabic digits. The *C* parameter of the linear SVM was set to 0.1. The kernel variance of the Parzen window was set to 1. The first 40 components of PCA were used for both PCA+Quadratic and PCA+ANN. The number of hidden units for ANN was set to 450, and to 240 for ANN+PCA. We used *C* = 10 and *γ* = 0.02 for the OVO-rbf-SVM.

Note that the order of arranging the classifiers in Fig. 3 is not arbitrary. They are arranged in a descending order form top to bottom according to their classification errors on MNIST. Hence, we may say that classifiers in Fig. 3 are ordered according to their powerfulness. We notice from Fig. 3 that classification errors of different classifiers on MADBase follow the same pattern of decay as the case of MNIST.

The most interesting observation that may be noticed from Fig. 3 is that although classification errors of the most powerful classifiers in case of MNIST are close to their scores in case of MADBase, there is a huge gap between classification errors of the weakest classifiers in case of MNIST and their classification error for MADBase.

An explanation for this may be as follows. The Arabic digit recognition problem is in general easier than that of Latin digits. This is why simple classifiers (like linear and OVO linear) score high on MADBase. However, the percentage of the poorly written digits are close for MADBase and MNIST (both around 1%) and they are equally hard to classify for the weak and the powerful classifiers. See Fig. 4 for the patterns misclassified by LeNet for Arabic digits and Fig. 5 for Latin digits.

But why Arabic digit recognition problem is simpler than that of Latin? We believe that there are two reasons. The first one is that the interclass distance between each pair of typical (i.e., not poorly written) Latin digits tend to be smaller than that between pairs of Arabic digits. When interclass distance between two digits is small, any variation in writing one digit will lead this digit to be easily confused with the other.

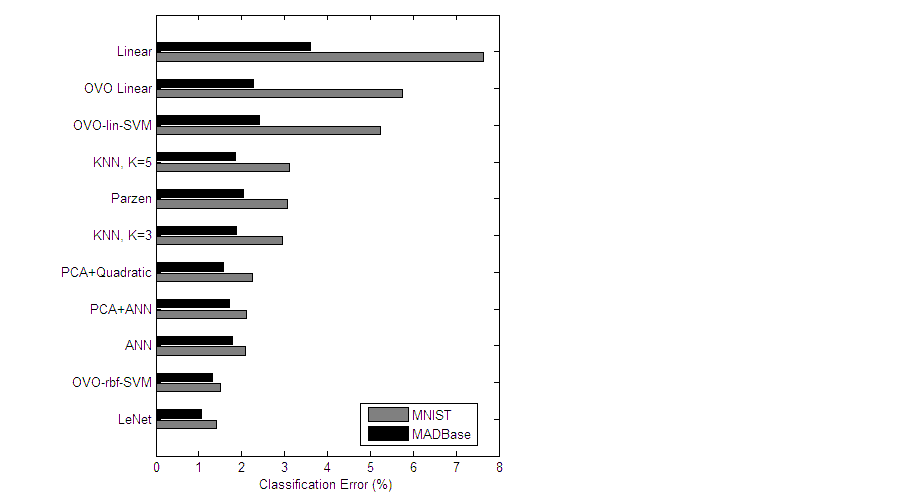


Fig. 3. Comparison of 10 classification techniques on MNIST and MADBase.

The second reason why Latin digits are harder to classify than Arabic digits is that the variance in writing each of Latin digits is in general higher than that in case of Arabic. In the following two subsections, each of these two proposed reasons is discussed in details.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

Fig. 6. Image at row *i* and column *j* is the result of subtractin digit *j* from digit *i* for (a) Latin digits, and for (b) Arabic digits

## **Interclass distances**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4→2 | 2→7 | 2→1 | 4→9 | 9→8 |
| 3→5 | 4→8 | 8→2 | 1→8 | 2→6 |
| 5→3 | 8→9 | 6→5 | 7→1 | 6→4 |
| 3→5 | 4→6 | 4→0 | 7→2 | 9→4 |
| 4→9 | 7→1 | 8→3 | 7→2 | 5→3 |

Studying Fig. 2, we may notice that Latin digits have so many common strokes, and, for many cases, any addition or deletion of a stroke from a Latin digit may lead it to appear as another one. For example, adding a stroke to upper right of the Latin digit ‘5’ will lead it to appear like ‘9’; and to appear like ‘6’, if we add a stroke at the left bottom of it. Also adding a stroke at left bottom of ‘9’ leads it to appear like ‘8’. Digit ‘4’ appears already like ‘9’, and removing the upper stroke of ‘9’ leads it to appear like another writing style of ‘4’. The fact that Latin digits have many common strokes helps Latin digits to be represented using only 7 strokes in a 7- segment display with natural appearance. But now if we look at Arabic digits in Fig. 2, we notice that there are not so many common strokes between them, and any attempt to represent them using 7-segment will lead to very artificial appearance (a dot matrix representation is usually used for them).

Fig. 5. Samples of hard-to-classify Latin digits (misclassified by LeNet).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4→9 | 4→2 | 0→8 | 5→9 | 9→6 |
| 1→0 | 5→0 | 6→4 | 5→2 | 9→4 |
| 4→0 | 0→5 | 3→2 | 0→2 | 5→2 |
| 3→6 | 9→8 | 3→2 | 9→2 | 3→2 |
| 9→6 | 4→2 | 5→2 | 5→0 | 3→2 |

Fig. 4. Samples of hard-to-classify Arabic digits (misclassified by LeNet).

A visual illustration of this idea is depicted in Fig. 6 which shows the resulting images from subtracting each digit mean from each of other digits means for both Latin and Arabic digits. The image at row i and column j is the result of subtracting the mean of digit j from that of digit i; where foreground (white) is represented by ‘1’ and background (black) is represented by ‘0’. Each pixel resulting from this subtraction is then added to 1, and then the result is divided by 2. Hence, if the result of subtraction for some pixel location is ‘-1’, it appears as black; if ‘0’, it appears as gray; and if ‘1’, it appears as white. Now, if any two digits have a common stroke, the average of their subtraction will be ‘0’ at the location of this stroke; and hence, will appear gray. Non-common strokes appear black or white depending on which digit is subtracted from the other. It is clear form Fig. 6 that Latin digits have many common strokes and hence the gray pixels dominate to the degree that most of the digit pairs are not distinguishable from each other. On the other hand, for Arabic digits, most of digit pairs are quite distinguishable.



Fig. 7. Some Latin digits and their different typical writing styles.

The above is rather informal evidence. In the following, more formal numerical evidence is going to be introduced. A numerical measure for the interclass distance between a pair of classes (i,j), i.e. between digit i and digit j, may be the Euclidian distance dij between their means,



where, *i,j* = 0,1,…,9; *x,y* = 1,2,…,28; and *μk*(*x,y*) is the mean value of the pixel at location (*x,y*) of the images of digit *k*,



where *Ikn*(x,y) is the pixel value at location (*x,y*) of the nth image of the training set of digit *k*, and *N* is the number of training samples of digit *k*.

A numerical measure for the overall separation between all class pairs (*doverall*) may be the summation of *dij* for all *i* and *j*, i.e., .

The value of *doverall* was found for Arabic digits to be larger than that of Latin by 11.9%. This means that interclass separations between Arabic digits are larger than that in case of Latin. To check the significance of this result, the hypothesis of having the *doverall* equal for both cases Latin and Arabic is checked. To do so, the training set is partitioned into 6 separate subsets, each containing 10,000 samples. The *doverall* measurement is calculated for each subset resulting in 6 different measurements. This is done for both cases Latin and Arabic. A *t*-test [29] is then performed to check the hypothesis that Latin and Arabic have equal mean value of *doverall*. The hypothesis was found to have a *p*-value of 4.3×10-8, which means that it can be rejected with very small significance level.

## **Variances**

The second reason why Latin digit recognition problem is harder than that of Arabic is that variances in writing Latin digits are larger than that of Arabic. One measure of variances (scattering) *Sk* of some pattern k around its mean is the trace of its covariance matrix [10],

,

where Σk is the covariance matrix of pattern *k*,  is the variance of the *ith* dimension of pattern *k*, and *d* is the dimensionality of the pattern. For our problem, the scattering of some digit *k* around its mean is then given by,



A numerical measure for the overall scattering is. It is found that *Soverall* for Latin digits is 27.9% larger than that of Arabic. This means that large deviation from the mean is more probable for Latin digits than that for Arabic. The reason behind this might be the fact that Latin digits have many writing styles as shown in figure 7.

To check the significance of the *Soverall* comparison between Arabic and Latin digits, the training set is partitioned into 6 separate subsets, each containing 10,000 samples. The *Soverall* measurement is calculated for each subset resulting in 6 different measurements. This is done for both cases Latin and Arabic. A *t*-test is then performed to check the hypothesis that Latin and Arabic have equal mean value of *Soverall*. The hypothesis was found to have a *p*-value of 1.1×10-11. This means that the hypothesis can be rejected with very small significance level.

# **Conclusion**

In this paper, we have taken a step to encourage doing more research on Arabic digit recognition problem. A comparison between performances of 10 classifiers has been made on both Arabic and Latin digit recognition problems. An interesting result of such comparison is that while the performance of the most powerful classifier on Arabic and on Latin digits are close, the weakest classifier gives very good performance on Arabic digits and much worse performance on Latin digits. The reason behind this has been suggested to be that Arabic digit recognition problem is in general easier than that of Latin. However, the percentage of poorly written digits are so close in both cases (Arabic and Latin digits), and they are actually hard to classify for both powerful and weak classifiers. An analysis has been given for why Arabic digits are in general easier to classify than Latin and it has been found that this is because interclass distances between Arabic digits are larger than that of Latin and variances in writing Arabic digits are smaller.

**Acknowledgment**

This research is supported by a grant form the American University in Cairo (AUC). We are also thankful to all who assisted us in building the database: Hossam Hassan, Reem Ater, Nagwa Ibrahim, Mohammed Ismail, Mohammed Khairy, Karim Ater, Mohammed Omran, and Ahmad Ra’fat.

**References**

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition”, Proceedings of the IEEE, vol. 86 no. 11, pp. 2278-2324, 1998.

[2] J. Dong, “[Comparison of algorithms for handwritten numeral recognition](http://www.cenparmi.concordia.ca/~jdong/DjxReport99.ps)”, Technical report, CENPARMI, Concordia University, 1999.

[3] C. Liu, K. Nakashima, H. Sako, H. Fujisawa, “Handwritten digit recognition: benchmarking of state-of-the-art techniques”, Pattern Recognition, vol. 36, pp. 2271- 2285, 2003.

[4] P. Zhang, T. Bui, and C. Suen, “Hybrid feature extraction and feature selection for improving recognition accuracy of handwritten numerals,” ICDAR, 2005.

[5] L. Teow, and [K. Loe](http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/l/Loe:Kia=Fock.html), "Robust vision-based features and classification schemes for off-line handwritten digit recognition," Pattern Recognition, vol. 40, no. 6, pp. 1816-1824, 2007.

[6] S. Belongie, J. Malik, J. Puzicha, "Shape matching and object recognition using shape contexts," PAMI, vol. 24, no. 24, 2002.

[7] F. [Lauer](http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V14-4MFCVV9-2&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=25d447ea833fbfca242534f1f9a20335#biog1#biog1)[a](http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V14-4MFCVV9-2&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=25d447ea833fbfca242534f1f9a20335#aff1#aff1), C. [Suen](http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V14-4MFCVV9-2&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=25d447ea833fbfca242534f1f9a20335#biog2#biog2), and G. [Bloch](http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V14-4MFCVV9-2&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=25d447ea833fbfca242534f1f9a20335" \l "biog3#biog3)[a](http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V14-4MFCVV9-2&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=25d447ea833fbfca242534f1f9a20335#aff1#aff1)," A trainable feature extractor for handwritten digit recognition," Pattern Recognition, vol. 36, pp. 2271- 2285, 2003.

[8] D. Gorgevik, D. Cakmakov, "An efficient three-stage classifier for handwritten digit recognition", ICPR'04, pp. 1051-4651, 2004.

[9] E. El-Sherif, S. Abdelazeem, “A Two-Stage System for Arabic Handwritten Digit Recognition Tested on a New Large Database”, International Conference on Artificial Intelligence and Pattern Recognition (AIPR-07), Orlando, FL, USA, July 2007.

[10] R. Duda, P. Hart, D. Stork, *Pattern Classification*, 2nd Edition, Wiley, New York, 2000.

[11] R. Caruana, A. Niculescu-Mizil, “An empirical comparison of supervised learnin g algorithms”, ICML, pp. 161-168, 2006.

[12] S. Mozaffari, K. Faez, M. Ziaratban, “Structural Decomposition and Statistical Description of Farsi/Arabic Handwritten Numeric Characters”, ICDAR,pp. 237-241, 2005.

[13] H. Soltanzadeh, M. Rahmati, “Recognition of Persian handwritten digits using image profiles of multiple orientations”, Pattern Recognition Letters, vol. 25, no. 14, 1569-1576, 2004.

[14] F. Al-Omari, O. Al-Jarrah, “Handwritten Indian numerals recognition system using probabilistic neural networks”, Advanced Engineering Informatics, vol. 18, no. 1, pp. 9-16, 2004.

[15] F. Said, R. Yacoub, C. Suen, “Recognition of English and Arabic Numerals using a Dynamic Number of Hidden Neurons”, ICDAR, pp. 237-240, 1999.

[16] Y. Al-Ohali, M. Cheriet, and C. Suen, “Database for recognition of handwritten Arabic cheques”, Pattern Recognition, vol. 36, pp. 111–121, 2003.

[17] N. Kharma, M. Ahmed, and R.Ward, “A new comprehensive database of handwritten words, numbers and signatures used for OCR testing”, In Proc of IEEE Canandian conference on electrical and computer engineering, pp. 766–768, 1999.

[18] N. Giusti, F. Masulli, A. Sperduti, “Theoretical and Experimental Analysis of a Two-Stage System for Classification”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 893 – 904, 2002.

[19] M. Hellman, "The nearest neighbor classification rule with a reject option," IEEE Transactions on Systems Science and Cybernetics, vol. 6, no. 3, pp.179–185, 1970.

[20] J. Friedman, “Another Approach to Polychotomous Classification”, Technical Report, Stanford University, 1996.

[21] K. Bennett, O. Mangasarian, “Multicategory Discrimination via Linear Programming”, Optimization Methods and Software, vol. 3, pp. 27-39, 1994.

[22] D. Price, S. Knerr, L. Personnaz, and G. Dreyfus, “Pairwise neural network classifiers with probabilistic outputs”, Advances in Neural Information Processing Systems 7 (NIPS-94),pp. 1109–1116, 1995.

[23] T. Hastie and R. Tibshirani, “Classification by pairwise coupling”, Advances in Neural Information Processing Systems 10 (NIPS-97), pp. 507–513, 1998.

[24] C. Cortes, V. Vapnik, “Support vector networks”, Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.

1. [↑](#footnote-ref-2)