An Efficient Character Recognition Scheme Based on K-Means Clustering

Sajjad Pourmohammad, Reza Soosahabi, Anthony S. Maida
The Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504
E-mail:{sxp7579,rxs5098,maida}@cacs.louisiana.edu

***Abstract*—Hand-written character recognition has been an active area of research. However, because of the recent advancements in mobile devices with limited amount of memory and computational power, efficient and simple algorithms for both online and offline character recognition has become more appealing. In this work, an efficient character recognition systems is proposed using LDA Analysis followed by a Bayesian discriminator function based on the Mahalonobis distance. Since LDA is tailored for Gaussian distributed data and the samples dimensionality is high, a couple of preprocessing steps have been applied to reduce dimensionality and cluster the data into semi-Gaussian subclasses. In the first step affine transformations are applied to the training samples in order to make the scheme robust against distortion. Scaling and Rotation are among those popular distortions which have been considered in this work. Inactive pixels are cut off using a simple algorithm in the next step. Then, PCA and k-means clustering are applied. The results from preprocessing showed a great potential in dimensionality reduction using transformations that can preserve useful information. Numerical results on the MNIST dataset reached 3% error rate which is lower than the other linear approaches. The proposed linear techniques are discussed in a way that make it easier to have a much clearer understanding of the method and why it works compared to the other classification methods.**

I. Introduction

Handwritten character recognition has been extensively investigated in the literature (see [1] for a performance comparison). Neural networks (NN) were one of the prominent early candidates for character recognition [2]. It later turned out that fully connected neural networks are not an efficient solution for character recognition. In 2000, it was even pointed out by the organizers of the Neural Information Processing System conference that the term neural networks in the submission arises criticism [3]. Significant breakthroughs in this area occurred after introducing convolutional neural network [4]. It is a special kind of multi-layer perceptron (MLP) inspired from early studies on visual cortex of a cat that utilizes feature detection and sub-sampling to effectively cope with diversity in handwritten character samples. Hinton and et al[5] introduced an efficient learning algorithm for deep belief networks which made a significant impact. Some efforts in reducing the misclassification rate showed significant improvements with making a committee of convolutional networks [5], [6].

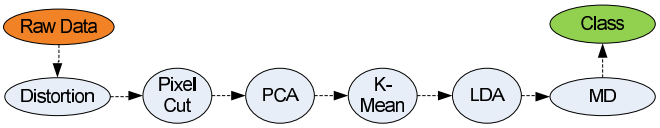

Fig. 1: General structure of the proposed method

Lowering the misclassification rate is feasible where the required computational resources are in abundance. Some of the aforementioned methods which provide very low error rates (less than 0.5%) are implemented on GPUs, using its parallel processing power to reduce the training time. Nowadays, mobile smart devices are getting very popular. Most of these devices have limited computational power and memory which hinders implementing complex algorithms. In this work a simple and efficient linear classification method proposed which offers fast computable algorithm for digit recognition. The method provides low computational complexity both in offline (training) and online (testing) stages. As opposed to neural network classifiers which have exponential time complexity [7], the SVD used in this paper has polynomial complexity [8]. Our results show 3% misclassification rate compared to error rates reported in [8], [9] (more than 4%).

Exploring the simplest alternatives for neural classifiers, LDA is selected which is followed with a Bayesian discriminator function based on Mahalonobis distance. Since LDA is tailored for Gaussian distributed data and the sample dimensionality is quite high, there are a few preprocessing stages to reduce dimensionality and cluster the data into semi-Gaussian subclasses. First a simple way is proposed to reject so many redundant pixels out of the samples. Then, the classic whitening is operated to regulate the data by transforming the data vectors into a new basis which includes the highest variance of data samples. Since the regulated samples in each class do not indicate unimodal distributions, k-means clustering is applied to each class, in order to split them into simply distributed classes. Fig. 1 demonstrates a flowchart of the proposed method.

II. Data Distortion

Adding distorted samples to the training data is a way to ensure that the developed classifier is robust to distortions.

Generating new samples can enhance the classification performance if the transformation to be applied does not change the properties to be learned [10]. There are already performance increase reports about applying distortions on the training samples for MNIST dataset [3], [11]. Simple distortions such as translations, rotations, scaling and skewing can be generated by applying affine displacement fields to images. A general form of affine transformation can be defined as below [12]:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} + B \tag{1}$$

where A is a 2×2 matrix and B is a 2×1 column vector. For example scaling and rotation transformation matrices are:

$$A_{scal} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}, \; A_{rot} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}. \tag{2}$$

One of the most important factors in applying affine displacement is dealing with the intermediate pixels. Different interpolation methods can result different qualities in the transformed images. Among the most popular interpolations applied are nearest neighbor, bilinear and bicubic interpolations. Nearest neighbor is the most basic and requires the least processing time of all the interpolation algorithms because it only considers one pixel the closest one to the interpolated point. This has the effect of simply making each pixel bigger. bilinear interpolation considers the closest 2×2 neighborhood of known pixel values surrounding the unknown pixel. It then takes a weighted average of these 4 pixels to arrive at its final interpolated value. This results in much smoother looking images than nearest neighbor. Bicubic goes one step beyond the bilinear interpolation by considering the closest 4×4 neighborhood of known pixels for a total of 16 pixels. Since these are at various distances from the unknown pixel, closer pixels are given a higher weighting in the calculation. Bicubic produces noticeably sharper images than the previous two methods, and is perhaps the ideal combination of processing time and output quality. For this reason it is a standard in many image editing programs (including Adobe Photoshop), printer drivers and in-camera interpolation [13].

Consider Fig. 2 in which 4 pixels with known intensity and 1 pixel with unknown intensity are depicted. Suppose we are going to find the pixel value at point $P = (x, y)$. If we show the pixel intensity with function $f(.)$, bilinear interpolation at P will be calculated as follows [14]:

$$\begin{aligned} f(P) \approx & \frac{(y_2 - y)(x_2 - x)}{(y_2 - y_1)(x_2 - x_1)} f(Q_{11}) + \\ & \frac{(y_2 - y)(x - x_1)}{(y_2 - y_1)(x_2 - x_1)} f(Q_{21}) + \\ & \frac{(y - y_1)(x_2 - x)}{(y_2 - y_1)(x_2 - x_1)} f(Q_{12}) + \\ & \frac{(y - y_1)(x - x_1)}{(y_2 - y_1)(x_2 - x_1)} f(Q_{22}). \end{aligned} \tag{3}$$

To better illustrate the different interpolation methods, the results after applying rotation to some samples from the MNIST dataset are depicted in Fig. 3. It can be seen that the poorest result is for Nearest neighbor interpolation. Bilinear interpolation provides a smoother result and finally the best result is from bicubic interpolation. Since, simplicity is one of the main motivations for this work; bilinear interpolation can be a good trade-off between simplicity and performance.

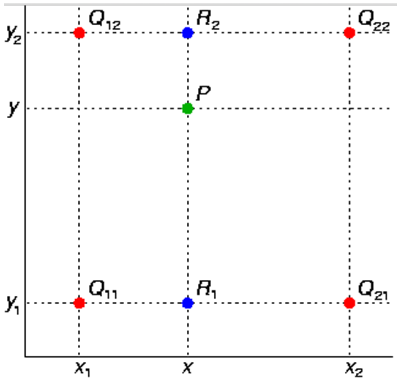

Fig. 2: Bilinear interpolation

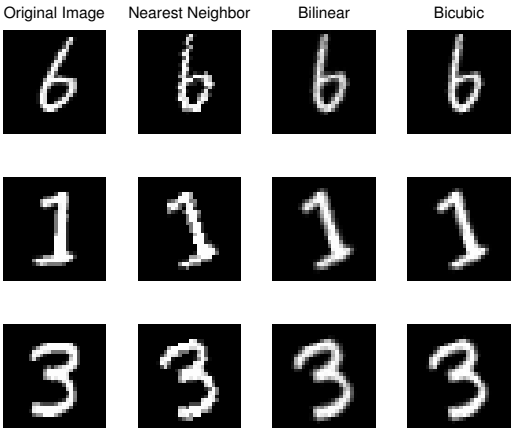

Fig. 3: Comparing different interpolation methods after applying rotation

III. Preprocessing

Computational complexity can rapidly grow with the increase in the number of training samples and network size. In this paper, the MNIST data set is used which contains 60000 samples of handwritten digits for training and 10000 samples for testing [9]. Fig. 4 shows a random selection of samples in this dataset. Referring to the above figure, it can be seen that there is a tire of pixels located around the frames which are always off in the entire sample set. Fig. 5 shows average pixel activity for the dataset.

It is clear from Fig. 5 that along with the totally off pixels in the data, there are some pixels which represent negligible activity for the entire dataset. To quantify the contribution of each pixel in the entire data set, it can be suggested to consider their value in the total average of the entire data set which is

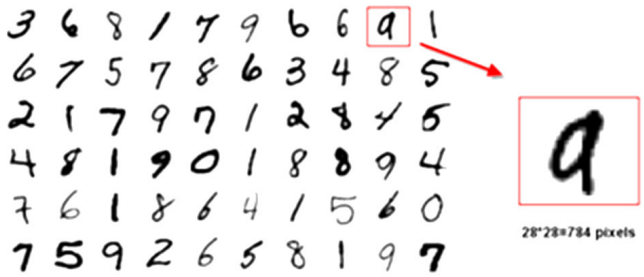

Fig. 4: Random samples from the MNIST dataset

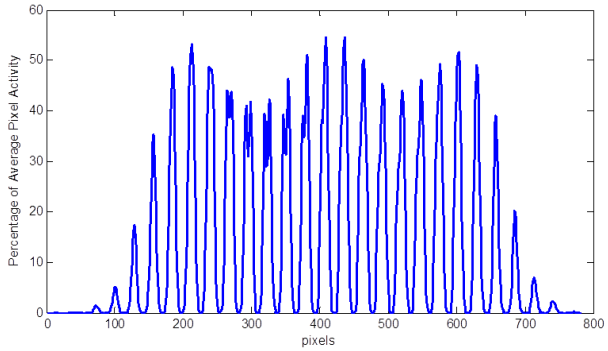

Fig. 5: Percentage of average pixel activity

computed as follows:

$$\mu_t = \frac{1}{N_t} \sum_{d=0}^{9} \sum_{k=1}^{N_d} x_k \tag{4}$$

where x_k and μ_t are sample and mean vectors, respectively. N_t and N_d are the total number of samples and samples per digit, respectively. Those dull pixels can be cut off using popular whitening methods like Principle Component Analysis (PCA). However, their presence adds up to computational complexity and leads to an ill-conditioned SVD. In this paper a threshold t_r has been considered for every pixel which is going to be included in the processing. All the pixels with activity below the threshold are eliminated resulting new samples ($\hat{x}_k$) and mean ($\hat{\mu}_t$) vectors. Using threshold around 7.8% of full pixel activity, the number of pixels reduces from 784 to 321, which is about 60% lower than the original data. It will be shown that this method does not considerably affect the misclassification rate.

A. PCA whitening

PCA analysis is one of the popular methods in the dimensionality-reduction literature [15]. PCA reduces the dimensionality of a data set while retaining as much as possible of the variation present in the data set. This is achieved by defining new set bases, the principal components, which are uncorrelated, and are ordered in a way that the first few components retain most of the variation presented in all of the original variables [16]. The entire transformation process

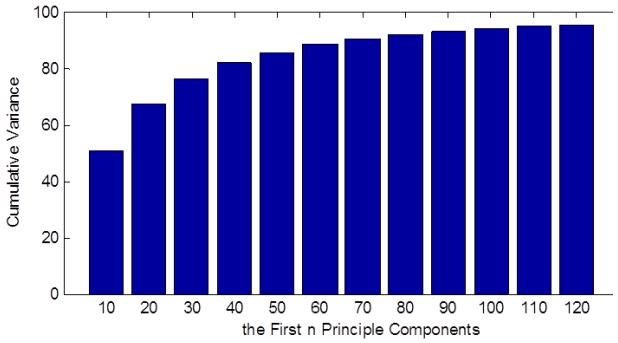

Fig. 6: Cumulative variances preserved in the n-largest principle components

can be summarized as follows [17]:

$$\begin{aligned} S_T &\triangleq \frac{1}{N_t} \sum_{d=0}^{9} \sum_{k=1}^{N_d} (\hat{x}_k - \hat{\mu}_k)(\hat{x}_k - \hat{\mu}_k)^T \\ &= U \Lambda U^T \end{aligned} \tag{5}$$

where Λ is a diagonal matrix containing the the eigenvalues of S_T and their corresponding eigenvectors compiled in U. Then we form

$$T_{pca} = (\Lambda_M)^{-\frac{1}{2}} U_M^T \tag{6}$$

$$y_k = T_{pca} \hat{x}_k \tag{7}$$

where M is the length of the corresponding PCA output y_k and T_{pca} is the PCA transfer matrix consisting of the M most-prominent eigenvectors in U (denoted by U_M) scaled with the square root of their corresponding eigenvalues (denoted by Λ_M). It is noted that all the eigenvalue in Λ are positive since S_T is a symmetric matrix. Let $\hat{N}$ denote the number of preprocessed samples. Then T_{pca} will of size $M \times \hat{N}$.

Fig 6 displays the amount of information preserved in the n-largest principle components after applying PCA transformation to the MNIST data set. There it can be seen that the variance preserved in the n-largest principle components rapidly increases with increasing n. It is noticeable that the first 10 principle components carry 51% of information existed in the whole dataset and the first 100 principle components represent more than 94% of variance information. So, a typical choice for M can be $C_t \leq M \leq 100$ in which C_t is the number of subclasses considered in K-means clustering.

B. K-Means Clustering

One of the main advantages of neural networks over linear classifiers such as LDA is their ability to trace nonlinear classification patterns even if their distribution is not unimodal. However, trying to transform training data into a unimodally distributed form can significantly reduce the need for NN and nonlinear classifiers in general.

Fig. 7 shows the average pixel activity in each class. One can see that for the MNIST dataset there is a variety of samples for each digit in terms of shape and the occupied pixels, so

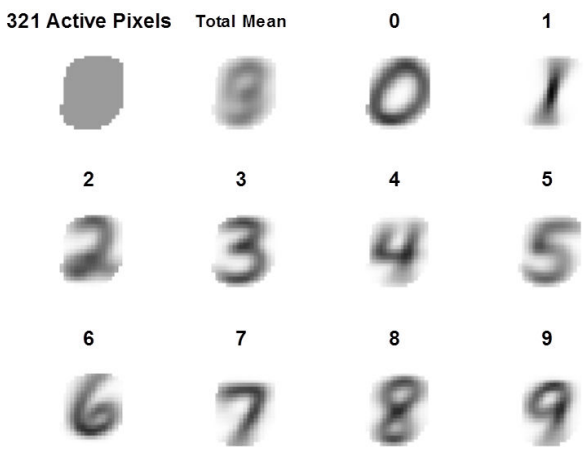

Fig. 7: Average pixel activity in each class

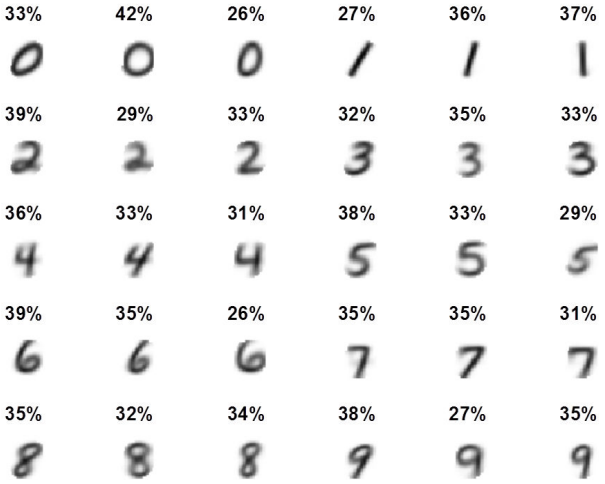

Fig. 8: K-means clustering of each class

considering a unimodal distribution for each class is far from an ideal assumption even after the dimensionality reduction. Thus we need to have a within-class clustering that is operated by k-means at a low computational complexity. It is clear that the pixels in each class do not fit into unimodal Gaussian distribution. According to Gaussian Mixture Model (GMM) method [17], we could approximate a variety of non-Gaussian distributions, especially smooth multi-modal distributions, by a linear combination of weighted Gaussian distributions. However, GMM requires a large amount of computational complexity when the data has high dimensionality. Thus, we need to find an efficient substitute for GMM which can fit to multi-modal distributed data. In this paper k-means clustering is used to divide each class to several subclasses with a unimodal Gaussian distribution. Fig. 8 shows the subclasses and their percentage after applying K-means clustering with $C_t = 30$. The subclass means look sharper compared to the original class mean. This reflects that the subclasses are from a lower variance population which can improve the classification performance.

IV. LDA Transformation

Having the preprocessing applied in the same way recommended so far, we could apply the classic LDA transform more effectively and faster. The within class scattering matrix, S_W, and the between classes scattering matrix, S_B, need to be computed. Computing S_W requires more computational effort than S_B. However, we do not need to compute S_W independently where S_B is computed and S_T is available from (5). Then, the LDA equations are applied as follows:

$$S_B \triangleq \frac{1}{N_t}\sum_{i=1}^{C_t} N_i(\hat{\mu}_i - \hat{\mu}_t)(\hat{\mu}_i - \hat{\mu}_t)^T \tag{8}$$

$$S_W \triangleq S_T - S_B \tag{9}$$

where $\hat{\mu}_i$ and N_i respectively indicate the mean and the number of samples in i-th subclass. Then we perform SVD on the following product

$$S_B^{-1} S_W = \phi \Sigma \phi^T \tag{10}$$

and use the resulted eigenvectors to form the LDA transform matrix T_{lda}:

$$T_{lda} = \phi^T \tag{11}$$

$$z_k = T_{lda} y_k = T_{lda} T_{pca} \hat{x}_k \tag{12}$$

where z_k's are fed to the discriminant function for classification. Since the Mahalanobis distances will be used in the classification, we need to estimate the first and second order statistics for z_k:

$$\mu_i^z \triangleq \frac{1}{N_i}\sum_{k=1}^{N_i} z_{ik} \tag{13}$$

$$\Sigma_i^z \triangleq \frac{1}{N_i - 1}\sum_{k=1}^{N_i}(z_{ik} - \mu_i^z)(z_{ik} - \mu_i^z)^T \tag{14}$$

where z_{ik} is the k-th transformed sample within the i-th subclass, μ_i^z and Σ_i^z are respectively the mean and the covariance matrix associated with the i-th subclass.

V. Numerical Results

To get the results, the test samples from the MNIST dataset are compared with each class of different digits. In the following the steps in preparing data and calculating the distance is presented.

A. Preparing Test Data

The same preprocessing steps are applied to the test data before using the classifier. If we denote u_k and $\hat{u}_k$ as test data before and after applying all distortion and preprocessing steps, the ultimately processed test data (v_k) which will be used to calculate mahalanobis distance with the training subclasses can be obtained by:

$$v_k = T_{lda} T_{pca} \hat{u}_k \tag{15}$$

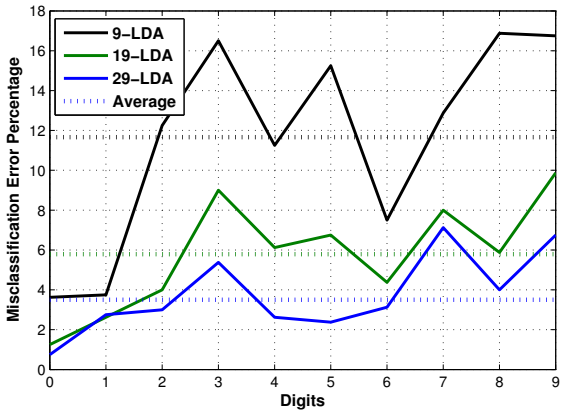

Fig. 9: Error Rate for Uniform Clustering Leading 32-PCA

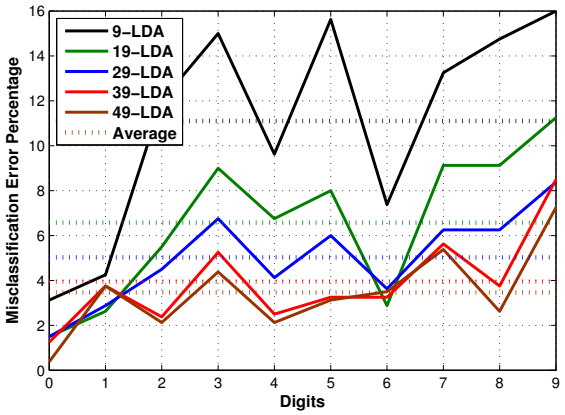

Fig. 10: Error Rate for Uniform Clustering Leading 64-PCA

B. Bayesian Classifier Using Mahalanobis Distance

For any given processed test sample, its Mahalanobis distance is computed with respect to each subclass as follows:

$$d_i(v_k) = \frac{1}{2}(v_k - \mu_i^z)^T \Sigma_i^z (v_k - \mu_i^z) + \frac{1}{2} ln(|\Sigma_i^z|) - ln(\frac{N_i}{N_t}) \quad , \quad 1 \leq i \leq C_t \tag{16}$$

v_k attributes to the subclass which produces the minimum $d_i(v_k)$. Then the final decision can be made considering the digit which that subclass belongs to. The trained classifier can be evaluated by an appropriate test sample set from the MNIST dataset. Fig. 9-11 shows the misclassification rates with different configurations of the preprocessing steps. For the best configuration error rate of 3.5% can be achieved. The number of principle components considered: $M = 32; 64; 96$, and k-means clustering is performed on each sample set with $1 \leq k \leq 7$. LDA transformation size is $C_t - 1$; in which different values are considered for C_t. It can be seen that increasing the PCA size does not lower the minimum achievable error significantly. Thus, 32-PCA with 29-LDA ($C_t = 30$) can be selected for actual tests providing the lowest computational complexity.

The error probabilities are not uniformly distributed over different digits in the test data set. Since there is no prior information about the frequency of each digit for actual tests, the ideal error configuration has to be uniform. We could see increasing K (i.e. $C_t - 1$) decreases the average error and simultaneously equalizes the error configuration. However, there are permanent error overshoots for some digits, like 3 and 9. This signals that their distribution cannot be well approximated by the sum of Gaussian distributions. To improve the error configuration, in particular for 32-PCA system with $C_t = 30$, non-uniform clustering is advised. This states that the training classes for the different digits can be split into a different number of subclasses applying k-means clustering. Then, we could allocate more subclasses to the digits which create error peaks, like 3 and 9, and less to those having less error rate.

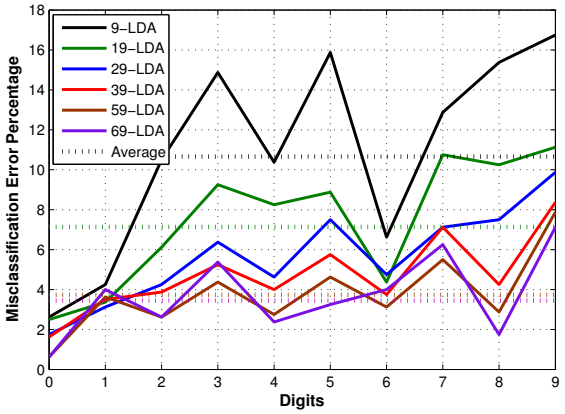

Fig. 11: Error Rate for Uniform Clustering Leading 96-PCA

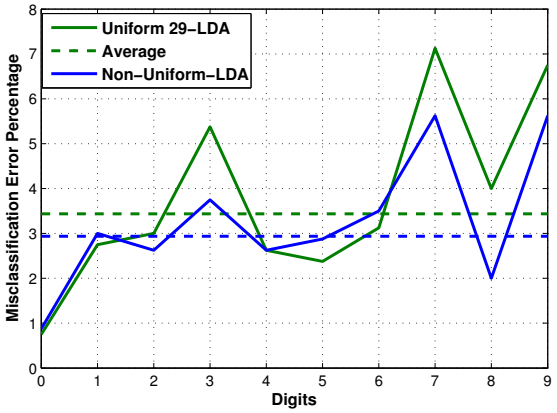

Fig. 12: Error Rate for Non-Uniform Clustering Leading 32-PCA

Fig. 12 shows the non-uniform clustering together with the best choice for uniform clustering. One can see that rational non-uniform clustering can lower the average error to about 3% and augment the error configuration.

VI. Conclusions

An efficient classification approach for character recognition has been presented. It has been shown that proper pre-processing can significantly reduce the unnecessary information in the training data. Compared to the other linear classifiers the method shows improvements in performance. The linear techniques used in the paper give us a much clearer understanding of the characteristics of the very popular MNIST dataset. The purpose and effect of each processing step and the pipeline of Fig. 1 is well understood. For example it is clear from examining figures 7, 8 what the contribution of K-means clustering is. Other figures make similar points. In contrast, even the most advanced neural network algorithms dont seem to give much insight into the structure of the dataset nor why (except in a very general probabilistic sense) the algorithm works. This paper shows the utility of carefully studying the target dataset.

References

[1] C. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques," *Pattern Recognition*, vol. 36, no. 10, pp. 2271–2285, 2003.

[2] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[3] P. Simard, D. Steinkraus, and J. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, vol. 2, 2003, pp. 958–962.

[4] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, 1995.

[5] D. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber, "Convolutional neural network committees for handwritten character classification," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 1135–1139.

[6] ——, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[7] P. Orponen, "Computational complexity of neural networks: a survey," *Nordic Journal of Computing*, vol. 1, no. 1, pp. 94–110, 1994.

[8] M. Holmes, A. Gray, and C. Isbell, "Fast svd for large-scale matrices," in *Workshop on Efficient Machine Learning at NIPS*, 2007.

[9] Y. LaCun. (2012) Mnist database @ONLINE. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[10] L. Yaeger, R. Lyon, and B. Webb, "Effective training of a neural network character classifier for word recognition," *Advances in Neural Information Processing Systems*, pp. 807–816, 1997.

[11] 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.

[12] F. Lauer, C. Suen, and G. Bloch, "A trainable feature extractor for handwritten digit recognition," *Pattern Recognition*, vol. 40, no. 6, pp. 1816–1824, 2007.

[13] cambridgeincolour. (2012, May) Digital image interpolation @ONLINE. [Online]. Available: http://www.cambridgeincolour.com/tutorials/image-interpolation.htm

[14] J. Russ, "The image processing handbook," *CRC Press*, vol. 70, no. 50, p. 30, 2011.

[15] J. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer, 2007.

[16] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[17] C. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 4, no. 4.