Classification of Handwritten Digits

Mario Coutino
MSc. Electrical Engineering
Delft University of Technology
Std. Num: **4410475**
M.A.Coutinomingue@student.tudelft.nl

Kris Shrishak S
MSc. Electrical Engineering
Delft University of Technology
Std. Num: **4411811**
K.S.Sridaran@student.tudelft.nl

***Abstract*—Handwritten digit recognition is a widely encountered pattern recognition problem. In this report two classifiers for this purpose are proposed when a small dataset and a large dataset are given for training. HOG features for both scenarios are used and the performance of this representation is then compared with pixel and dissimilarity based approaches to show their lower error rate. It is also shown that considering a large number of features is not always an advantage and hence feature reduction techniques like PCA need to be used. A live test has been performed using the selected classifiers and the performance has been analyzed.**

I. Introduction

Handwriting digits recognition is a well known problem in both pattern recognition and image processing [9]. As handwriting is a personal ability, huge differences can be found among a sample of digits taken from several people [13]. This issue becomes of interest in pattern recognition when a way to generalize the description of a digit out of a given training set is searched.

In this report, the results for two different scenarios proposed by a fictional client are shown. The first scenario addresses the necessity of a system to be able to deliver an error rate below 25% under the assumption of at most 10 samples per class, which implies a high probability of overfitting when a complex classifier or a high number of features are used. The second scenario requires an error rate below 5% when a training set with at least 200 samples per class is provided. In this latter case a more complex classifier and/or representation can lead to better performance rates.

The current report continues as follow: In Section II an outline of our approach will be elucidated. First, the data set that will be used as well as the pre-processing suggested will be discussed. Then, three different representations will be addressed (pixels, features and dissimilarity), pointing out their basic characteristics and methodologies for their usage. In Section III a fair comparison between selected classifiers in each of the representation is performed. Section IV discussed the outcomes of the comparison and the classification results over the frozen data held by the client. In addition, Section V shows the results of a live test performed with a real sample of digits using the selected classifiers. Section VI outlines some recommendations with respect to handwriting recognition. Finally, in Section VII we conclude and discuss future improvements.

II. Approach Overview

Our goal in this project is to identify the best classifier in each of the representations and then select the best one (in terms of classification results and, training and evaluation performance).
In order to do so, a fair comparison between the three data representations in both scenarios is needed. A design pipeline is proposed to build a coherent comparison between classifiers and reduce the complexity of the initial problem by restraining it to a set of solutions that can achieve the desired results but also our personal designing criterion.

The use of raw images is avoided by performing an initial preprocessing over the images. The new images are transformed into one of the proper representations (pixels, features and dissimilarity). Some variants over the representations are tried and are discussed further in this report (this includes representation reduction and features/dissimilarity measures). Finally, a set of 13 classifiers are tested in each of the representation in order to find the best candidate of each representation.

All these experiments are evaluated for both scenarios using 10-fold cross validation [14] and by testing over a frozen set of NIST data. In addition, for Scenario I (training set by batches) an average over random training sets is computed in order to obtain a better estimate of the classification performance in unseen batches. Finally, all the candidates for each representation and the best of each of the scenarios is selected, taking into account not only its error rate but also training complexity and evaluation time. Fig. 1 offers a complete overview of our proposed design pipeline.

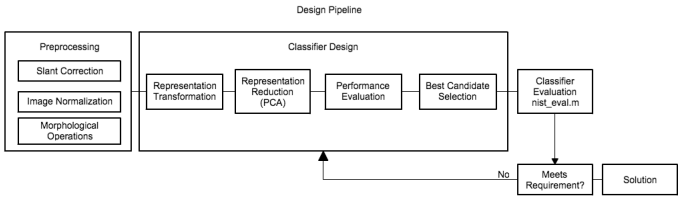

Fig. 1: Classifier Design Pipeline

In the following parts the approach will be discussed in further detail. The report is meant to be self-contained. The reader mainly interested in results can proceed directly to Section III.

A. Preprocessing

Preprocessing is an essential step in any recognition system. It prepares the raw data for the subsequent stages of processing, removing noise or enhancing the original properties of the information [15]. In addition, preprocessing of the raw data will also reduce the complexity in it (slant, quantization, bad sampling, etc), removing some of the problems which decrease the rate of recognition.

In the current work two preprocessing steps are performed over the data before its usage in the classification task: *slant correction* and *image normalization*.

The correction of slant reduces the personal influence on the data, while image normalization standardizes the image in order to use a *fair* representation for each sample of each class.

Slant Correction

As slant is the most basic characteristic of a person's handwriting [13], the correction of it can enhance the classifier's performance (reducing particularities over the writing, looking for general descriptors, not personal tendencies in the samples). Any line whose direction is not perpendicular or horizontal is a slant. Slant can be uniform and non-uniform. The slant angle is the same in the case of uniform slant while it is varying in the case of non-uniform slant.

In our application, we consider non-uniform slant as each individual person has a different slant angle and each number written by the same person might also have different slant angles. In order to correct this problem, the application of an *affine* transformation over the the images is used.
First, the direction θ of the higher inclination's variance in relation to the vertical axis (Eq. 2) is found using central moments (Eq. 1) and the information from the pixel distribution.

$$\begin{aligned}\mu_{j,k} &= E[(X - E[X])^j (Y - E[Y])^k] \\ &= \int_{-\infty}^{+\infty} (x - \mu_X)^j (y - \mu_Y)^k f(x,y) dx dy \end{aligned} \tag{1}$$

$$\theta = \arctan \frac{2\mu_Y}{\mu_X - \mu_{XY}} \tag{2}$$

It is important to notice that our definition of orientation (Eq. 2) differs from the usual one which considers the direction over the largest eigenvalue. The reason is that we look for the angle of projection of maximum spread along Y, in order to correct it.

Then, an affine transformation is used to rotate the image in order to position the orientation of the maximum variance parallel to one of the axis. The rotation matrix used is described in Eq.3.

$$S_\theta = \begin{pmatrix} 1 & 0 & 0 \\ \sin(0.5\pi - \theta) & \cos(0.5\pi - \theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

Even though the images are $2D$, the routine used in MATLAB requires a $3D$ matrix in order to perform the transformation. As the last column and row only contain a 1, no modification is performed over Z, which keeps the transformed image (with slant correction) in the original $2D$ space.

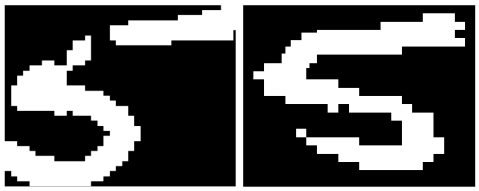

Fig. 2: Results of Preprocessing over digit image. Original (Left), Slant Correction + Morphological Operations + Normalization (Right).

Image Normalization

One of the problems in object classification, is the objects description. Along with slant, other characteristic that can hinder the classification rate is the object dimensions [15]. As each person writes digits in different sizes, normalization has to be performed over the digits.

Scaling complicates the recognition system, as the image proportions can be misleading for the classifier.
In order to simplify the recognition of digits, the image size is normalized to a single size ($[20 \times 20]$ pixels) and, finally a black bounding box is added to avoid *edges* effects (an addition of two rows and two columns).

The complete preprocessing pipeline leads to a cleaned, slant corrected and normalized image with a final size of $[22 \times 22]$ pixels, which will be used later for the input of different representations and the classification task. An example of the preprocessing can be seen in Fig. 2. The output image also incorporates morphological operations not discussed in the report, further information about this kind of operations can be reviewed extensively in [16].

B. Data Representation

All learning processes share a common process: data gathering. Meaningful information has to be retrieved in order to be able to represent, somehow, reality. The way in which this information is stored, manipulated and, ultimately in our case, learned is what it is called *data representation* [11].

In general, common representations of data are feature spaces, power spectrum, graphs, (dis)similarities measures [11]. Based on different representations, objects can be compared (how similar or how different they are from each other) and as a result, the recognition task is reduced to the problem of finding a *classifier* that use this representation to find a *generalized* description of each of the classes (or objects).

In the current work three major representations (pixels, features and dissimilarity) used for image processing are discussed. In the following part main characteristics of each representation and how their issues are addressed is discussed in detail.

Pixels

Digital images are nothing more that light intensity variation at a particular location in a grid (Fig. 3). Each element

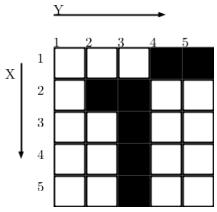

Fig. 3: Pixel Representation of an Image

of these matrices are called *pixels* or picture element. As pixels represent light intensity distributed in space, they can be thought as the most fundamental digital representation of an image.

Pixels capture, in raw manner, all the information of an image. The main drawbacks of this representation are the high variability in values that can exist between sensor and the high amount of information, which most of the time is redundant [19].

Designing a classifier based on pixel representation could be asymptotically perfect as more and more data is available (in the limit, in theory, we will have all possible images of our objects). However, this approach is far from optimal. Huge amount of memory and computational power will be necessary in order to use this approach and still more efficient solutions can be implemented.

In order to deal with these issues several techniques can be used. Pixel selection can reduce the amount of data to be used looking for the most representative pixels and, ultimately, a change of representation by feature extraction. This last solution will obtain a different representations of the image with, hopefully, lower dimensionality.

Features

A feature is an individual measurable heuristic property of a phenomenon being observed (i.e temperature, diameter, taste, etc). Choosing discriminating and independent features is key to any pattern recognition algorithm being successful in classification [4].
The set of features f_1, f_2, ..., f_m of a given data instance $\mathcal{I}$ is often grouped into a feature vector $\mathbf{f}$ which represents the object $\mathcal{O}$ and allows the comparison of it against the other in a vector space $\mathbb{R}^m$, which facilitates, most of the times, the discrimination process.

$$Object: \ \mathcal{O}_k = [f_1, f_2, ..., f_m] \tag{4}$$

Feature Extraction

Feature extraction is a process that extracts salient features from observed variables [1]. It performs two tasks: transforms a input parameter vector $\mathbf{x}$ into a feature vector $\mathbf{f}'$ and modifies its dimensionality $T : \mathcal{X}^m \rightarrow \mathcal{Y}^l$, most of the times we expect a reduction in order to obtain a *smaller* representation of our object (condensed).

A well-defined feature extraction algorithm makes the classification process more effective [2].

In our application, for feature representation, we only focus in a single feature set: Histogram of Gradients (HOG). Other feature sets (moments, salients, etc.) were considered during the initial stages of the project, but after a literature review in handwriting digits recognition [5] [9] only HOG was selected for further analysis.

Histogram of Gradients (HOG)

Histogram of Oriented Gradient (HOG) was first proposed by Dalal and Triggs [3] for human detection but has since been used in many pattern recognition applications.

In order to provide an overview of this feature set, the key steps and parameters of HOG are discussed. For details in the method, the interested reader should consult the article by Dalal and Triggs in the references.

- *Gradient Computation*
  The first step in HOG is the gradient computation. A $1D$ centered, point discrete derivative mask is applied on both horizontal and vertical directions.

  $$[-1\ 0\ 1]\ and\ [-1\ 0\ 1]^T$$

  Other complex masks can be used to performed the computation, but has been shown by Dalal and Triggs (2005) that no gain is acquired, they even shown that this can be detrimental for the detection performance.

- *Cell Size*
  The next is to decide on the cell size in order to construct the histograms. We tried different cell sizes and compared the classification error of the overall algorithm, as a first estimate of performance over the feature characteristic.
  A cell size of 8×8 gave a small feature vector but the classification result was unsatisfactory (too much information is lost in the averaging process). A cell size of 4×4 gave good results but the feature vector was too large. As result we had to performed a trade off between cell resolution (feature size) and performance. We decided to choose a cell size of 6×6 as the results obtained meet the design conditions and the feature has a manageable dimension because we expect to reduce this dimension in later stages of design. In Fig 4 a comparison between different cell sizes is shown.

- *Orientation Binning*
  We use a block size of 2×2 by default with 50% overlap. In each block, the gradient orientation is quantized into 9 orientation bins. The histogram bins are evenly spread over 0 to 360 degrees.

- *Block description*
  The pixels whose gradient alignment falls in a particular orientation are summed up in each block. This is the strength of edge gradient for each orientation bin for each block. In order to account for illumination and contrast changes, the gradients are locally normalized for each block.

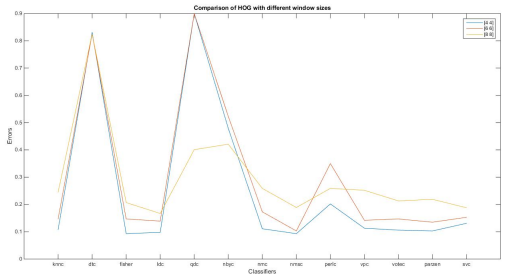

Fig. 4: Comparison between different cell sizes for HOG over the proposed classifiers.

Dissimilarity

Beyond pixel and feature representation it is possible to describe the data by means of how different the object are in relation with each others. Measuring the *difference* between objects gives an alternative way of comparing objects in a vector space. In addition, the dissimilarity space can be created over an existing feature representation or the raw data, giving enough flexibility to discover underlying information not visible in the original representation.

The issue that arises now is how to measure the difference fairly. As a measurement is required to gauge the dissimilarity between the objects, we are forced to come up with a fair distance to describe the relationships between the objects. These situations lead us to the main advantage of dissimilarity: expert knowledge can be now naturally introduced in the system. However, the problem of constructing a meaningful *measure* is not trivial [12].

As stated in [7] [12], the selection of a proper dissimilarity measure is a complex task. As one of the biggest advantages of dissimilarity representation is the possibility to represent non-metric distances (non-Euclidean), the selection of the measure has to be mainly based in prior knowledge obtained from the problem.

In this project, discussion on pseudo euclidean embedding [12] is avoided for simplicity. In order to make a smooth first approach to dissimilarity representation only well studied distance measures for binary vectors [8] is being used.

From all the dissimilarities measures discussed in [6] and [7] we selected the two measures with the best performance as described in [6] and included a proximity map (*proxm* in prtools) using the euclidean distance.

Dissimilarities Measures

- Rogers-Tanmoto

$$\mathcal{D}_{X,Y} = \frac{2S_{10} + 2S_{01}}{S_{11} + S_{00} + 2S_{10} + 2S_{01}} \tag{5}$$

- Sokal-Michener

$$\mathcal{D}_{X,Y} = \frac{n - S_{11} - S_{00}}{N} \tag{6}$$

- Euclidean Distance

$$\mathcal{D}_{X,Y} = \|x - y\|_2^2 \tag{7}$$

where $S_{ij}(i, j \in \{0, 1\})$ is defined as the number of occurrences of matches with i in the first pattern and j in the second pattern at the corresponding positions.
In the designing phase these three binary distances are applied to the raw data (image pixels) of the training set in order to build the dissimilarity matrix $\mathcal{D}$. Fig.5 shows an example using Sokal-Michener over the training set.

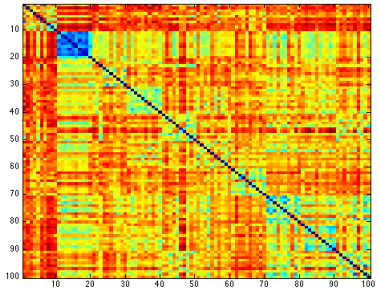

Fig. 5: $\mathcal{D}$ from the training dataset using Sokal-Michenel dissimilarity

This matrix is then used to train the proposed classifiers using a dimension-reduced representation of $\mathcal{D}$, which is performed over feature selection using PCA. An example of feature selection using PCA for dissimilarity over all proposed classifiers is shown in Fig.6.

Representation Selection

In many cases there are several reasons to perform feature selection [18]. In addition, even when individual features carry high amount of information, when they are used together, not much gain is obtained due high mutual correlation. Finally, the main goal of the classifiers is to describe, in the most general way possible, the structure of each of the classes, which can can be hindered by a high number of features.

In order to obtained a good ratio between free parameters and training patterns, Principal Component Analysis (PCA) was selected as our technique for representation/features reduction. PCA allows us to reduce the mutual correlation of the features and to construct an orthogonal basis capable of explaining the most of the variance in the data [17].

Feature curves were implemented using PCA in order to obtain the best number of features for each of the classifier in the different representations. An example of the procedure can be seen in Fig. 6.

Principal Component Analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated features into a set of values of linearly uncorrelated features called principal components [17]. The number of principal components can be

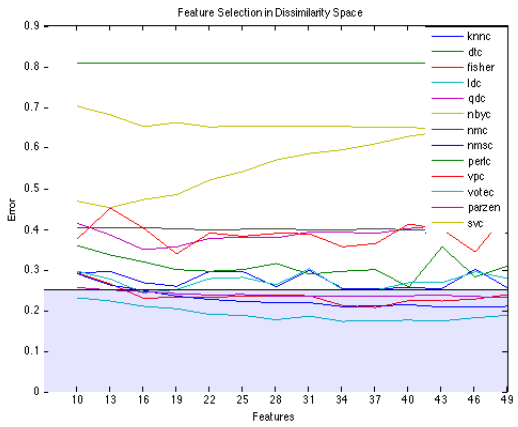

Fig. 6: Feature Selection over Sokal-Michener Dissimilarity Measure using PCA. Blue box enclose all classifiers which meet the performance requirements. (Small training set $\epsilon < 25\%$)

equal to the number of original features or less than the number of original features in the case of feature reduction. The central idea behind feature reduction is to reduce redundancy. Redundant cases can be identified by using a covariance matrix.
Let $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \ldots \mathbf{x}_n]^T$ be a matrix which $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ are the row vectors representing each features.
Now

$$\mathbf{C_x} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$$

contains in its diagonal elements the variance of each of the variables. In addition the non-diagonal elements provides an insight of the existence of redundancy of the data.

Optimally, all off-diagonal elements should be zero. In general, we consider a covariance matrix $\mathbf{C}_Y$ where $\mathbf{Y} = \mathbf{PX}$ and $\mathbf{P}$ is the orthonormal matrix. $\mathbf{P}$ is made of the basis vectors $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$. The next step is to diagonalize $\mathbf{C}_Y$. This can be done by the following algorithm:

1) Select a normalized direction in m-dimensional space along which the variance in $\mathbf{X}$ is maximized. Save this vector as $\mathbf{p}_1$.
2) Find another direction along which variance is maximized, however, because of the orthonormality condition, restrict the search to all directions orthogonal to all previous selected directions. Save this vector as $\mathbf{p}_i$.
3) Repeat this procedure until m vectors are selected. Now $\mathbf{p}_i$ is rank ordered and the ordered set of $\mathbf{p}$ś are called the principal components. The orthonormality assumption allows us to obtain a simple analytical solution to the problem.

The process of diagonalyzing the covariance matrix $\mathbf{C}_x$ is performed by Singular value decomposition (SVD) and the principal components obtained are the eigenvectors of $\mathbf{X}$.

III. Classifier Performance

The processed image $[22 \times 22]$ pixels is considered as the input image to each of the representations. Later, each representation is used to train 13 classifiers in order to compare their performance. The choice of 13 classifiers was made keeping in mind the computational complexity and the simplicity of the data in hand (i.e neural networks were excluded in the comparison as we consider their complexity unnecessary for the task in hand).

In addition, as discussed in Section II, feature selection is performed using PCA for each classifier and each representation in order to find the best set or combination of features for each case, searching for a considerable improvement in the performance of the classifiers.

The error rates of the best classifiers from each of the representations for both scenarios are shown in the tables below. In each of them the classifier with the smallest error rate is pointed out.

A. Scenario 1

Classifier	Full Info	PCA 60D	PCA 38D
nmsc	0.9	**0.087**	0.111
fisher	0.602	0.108	0.139

TABLE I: HOG144 [6x6] – Error rates of best classifiers (10 objects/class)

Classifier	Full Pixels	PCA 80D	PCA 40D
ldc	0.9	0.679	**0.215**
svc	0.231	0.234	0.24

TABLE II: Pixel – Error rates of best classifiers (10 objects/class)

Classifier	Full Info	PCA 34D	PCA 88D
Sokal Michener (SM)			
ldc	0.19	0.173	**0.169**
nmsc	0.378	0.209	0.204
Euclidean (E)			
ldc	0.251	**0.183**	0.185
nmsc	0.385	0.215	0.22
RogerTanMoto (RM)			
ldc	0.19	0.175	**0.172**
nmsc	0.378	0.211	0.21

TABLE III: Dissimilarity – Error rates of best classifiers (10 objects/class)

B. Scenario 2

Classifier	Full Info	PCA 38D	PCA 60D
knnc	0.02	**0.02**	0.04
vpc	0.05	**0.02**	0.03

TABLE IV: HOG144 [6x6] – Error rates of best classifiers (200 Objects/class)

Classifier	Full Pixels	PCA 40D	PCA 80D
qdc	0.9	**0.064**	0.759
knnc	0.108	**0.078**	0.083

TABLE V: Pixel – Error rates of best classifiers (200 Objects/class)

Classifier	Full Info	PCA 34D	PCA 88D
Sokal Michener (SM)			
qdc	0.9	**0.066**	0.099
parzen	0.9	0.091	0.091
Euclidean (E)			
qdc	0.9	**0.066**	0.078
parzen	0.9	0.089	0.09
RogerTanMoto (RM)			
qdc	0.9	**0.066**	0.099
parzen	0.9	0.091	0.091

TABLE VI: Dissimilarity – Error rates of best classifiers (200 objects/class)

IV. Results

A. Scenario 1

On training the 13 classifiers using the three representations (pixels, features and dissimilarity), the classifiers showing the best performance were found. Using HOG144 and PCA 60D, Nearest Mean Scaled Classifier (*nmsc*) gave the best performance with 8.7% error (Table I). In the case of pixel representation and dissimilarity, LDC gave the best performance. As can be seen from table I, II and III, the performance of **nmsc** with **HOG144** and **PCA 60D** gives the best performance in this case and hence is our **choice** for this scenario. NMSC is similar to a normal nearest mean classifier with the assumption of normal distributions, which leads to feature scaling and shows sensitivity to priors. The second characteristic can become a problem, however the usage of balanced training sets overcomes this issue, allowing us to take advantage of the scaled distributions which lead to avoid overfitting (in this particular case).

Table VII reinforces our selection by showing the results after the validation process using the client routine for the set of best classifiers.

Classifier	nist_Eval()
SM*34D*ldc	0.215
SM*34D*nmsc	0.215
RM*34D*ldc	0.219
RM*34D*nmsc	0.217
E*34D*ldc	0.227
E*34D*nmsc	0.211
HOG144*60D*nmsc	**0.086**
HOG144*60D*fisherc	0.117
Pix*40D*ldc	0.231
Pix*40D*svc	0.252

TABLE VII: Evaluation of best classifier using nist_eval()

B. Scenario 2

In this scenario, the performance obtained by all 13 classifiers using pixels and dissimilarity representation were unsatisfactory as can be seen in Table V and VI. Using HOG144 and PCA 38D, knnc and voted perceptron classifiers gave a good performance of 2% error rate (Table IV). It can be observed in Table IV that the curse of dimensionality is imminent when a large data set is used. Using PCA 60D gives a much larger error compared to PCA 38D. We decided to choose the **HOG144 voted perceptron classifier** with **PCA 38D**. This choice is justified by Table VIII where the performance of Voted perceptron is better than knnc using after performing an averaging over several runs in the client validation routine.

Finally, another reason for the selection of **vpc** over **knnc** is because vpc is *training-expensive* rather than *evaluation-expensive*. This means vpc takes time to be trained, but after that vpc is just a linear classifier contrary to knnc which needs no training but needs considerable time to perform evaluation. For big amount of data, as in Scenario II, vpc seems to us a better option: only time is invested once.

Classifier	nist_eval()
SM*34D*qdc	0.076
SM*34D*parzen	0.082
RM*34D*qdc	0.0933
RM*34D*parzen	0.0963
E*34D*qdc	0.102
E*34D*parzen	0.114
HOG144*38D*vpc	**0.032**
HOG144*38D*knnc	0.05
Pix*40D*qdc	0.056
Pix*40D*knnc	0.064

TABLE VIII: Evaluation of best classifiers using nist_eval()

V. Live Test

So far only objects from NIST has been used to test the classifiers performance. In the current section a live test is performed using our own handwritten digits. In Fig. 7 a sample taken from the authors is shown. It should be noticed that some of the digits highly differ in shape from the one in NIST (i.e number seven).

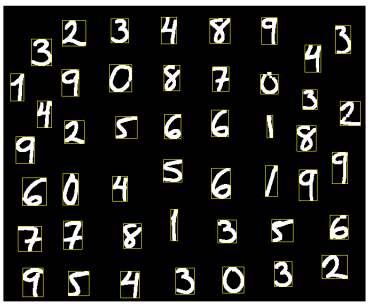

Fig. 7: Segmentation output from Live Test (Authors handwriting)

Automatic segmentation of the numbers is performed over the image. Preprocessing is applied over the extracted digits in order to be compatible with our pre-trained classifiers. A comparison of the performance using the selected classifiers trained with the NIST data and the live test data is shown in Table IX.

In addition, from the confusion matrix in Table X it is possible to observe that most of the error of our classifier are

Classifier	Error Rate
HOG144+ **vpc38D** + NIST (200 Objects/class)	0.21739
HOG144 + **nmsc38D** + NIST (10 Objects/class)	0.26087
HOG144 + **vpc38D** + LiveData (Training Set Error)	0.0001
HOG144 + **nmsc38D** + LiveData (Training Set Error)	0.021739

TABLE IX: Classifier Performance Comparison in Live Test

when the numbers seven and six are evaluated. The errors are due the particular handwriting of the authors. With both proposed classifiers the error rates are above 5% and 25% for scenarios I and II respectively. When the training set error is computed, it can be seen that with almost no error the classifiers can separate all digits.

True Label / Est. Label	0	1	2	3	4	5	6	7	8	9	Total
0	4	0	0	0	0	0	0	0	0	0	4
1	1	2	0	0	0	0	0	1	0	0	4
2	0	0	4	0	0	0	0	0	0	0	4
3	0	0	0	7	0	0	0	0	0	0	7
4	0	0	0	0	4	0	0	0	0	1	5
5	0	0	0	0	0	4	0	0	0	0	4
6	0	0	0	0	0	**2**	3	0	0	0	5
7	0	0	0	**2**	0	0	0	1	0	0	3
8	0	0	0	0	0	0	0	1	3	0	4
9	0	0	0	0	0	0	0	2	0	4	6
Totals	5	2	4	9	4	6	3	5	3	5	46

TABLE X: Voted Perceptron 38D trained with 200 Objects per class using NIST

VI. Recommendations

By observing both scenarios it is possible to see evidence of a relation between the amount of data available for training and the error rate. As more data is at our disposable, smaller the error rate becomes. It is important to point out that even when the data can possibly be linear separable, the personal component in handwritten digits makes impossible a 0% error rate.

In addition to the inherent personal component, bad samples are always going to exist. If the classifiers are trained using an *uncontrolled* data set, this mean that any handwritten symbol is considered without first checking if it is a number or not, the performance will be affected. In order to deal with this issue a hierarchical-scheme can be adopted. Before training the classifiers, the data should be evaluated to reject any non-digit symbol. This procedure can be seen as an outlier detector where, possibly, dissimilarity space-based classifiers can help (by choosing a proper representation set, the rejection can be carried out naturally).

Another important question to tackle in this set up is: what else will help? First, new features. Even though it was showed that increasing the dimension of the HOG vector not necessary improve the performance, adding *unseen* information may help. The hard part of adding features is the selection of meaningful information as the addition random features can even hinder the classifier performance. In our particular case, geometric-based features can be a good match with HOG features as a hierarchical classification can be implemented.

Second, more power classifiers. From the very beginning neural networks (NN) were not considered by the preferences of the authors due to its complexity, but it is well known [5] that NN can achieve rates above 99% easily when huge amount of data is fed to them.

Using a single classifier was shown plausible for handwritten digits but only if its usage is within its design guidelines. For example, if a travel-cheque scanner is needed, nmsc will be able to coupe with geographical differences as it will be trained on the spot, while a pre-trained terminal or software trained with tons of data will be perfect for a bank. No multiple classifiers are needed in order to solve any of the two scenarios.

Finally, time and implementation are also important when a classifier should be chosen over another. During the design and proposals selection this was taking into account. NMSC just scale the data and then computes its means, later 10 distances are computed for each query. This approach is fast in every sense. VPC needs some time to be trained, not much but still requires time, and delivers a linear classifier which is simple to evaluate. If no time constraints are set NN or KNNC classifiers become also an option. These two are in opposite sides. NN is training-expensive and KNNC evaluation-expensive. If no further modifications are going to be made to the system, NN can be an option with just one train session. On the other hand, if a fast platform is at disposal (i.e FPGA, embedded solution, etc) KNNC is easy to design, no train is needed and it can be run in parallel.

VII. Conclusion

A comparison between different data representations in order to select the best classifier for each scenario was performed. The proposed classifiers trained using HOG features were found to outperform their pixel and dissimilarity based counterparts. In addition, the designed classifier fulfill all the client requirements, showing a novel-data classification error within the desired range. Scaled nearest mean classifier obtained the best performance when the training is performed using small data batches. Voted perceptron classifier and k-nearest neighbors classifier showed similar error rates, but voted perceptron was chosen as the best candidate due its linear nature. Its semi-intensive training time is compensated by its small evaluation time in contrast with k-nearest which has to compute all the distances for each new query. In both scenarios, mainly in Scenario 2, curse of dimensionality was observed. In order to get the best performance out the classifiers (and reduce their complexity), the dimension of the features vector considered for classification was reduced using PCA and cross-validation. Finally a live test was performed and the performance of the selected classifiers was analyzed.

References

[1] Yung-Keun Kwon, Byung-Ro Moon, *Nonlinear feature extraction using a neuro genetic hybrid*, Genetic and Evolutionary Computation Conference - GECCO , 2005, pp. 2089-2096

[2] X. Wang and K. Paliwal, *Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition,* PR- Pattern Recognition, 2003, vol. 36, no. 10, pp. 2429-2439

[3] N. Dalal and B. Triggs, *Histograms of Oriented Gradients for Human Detection*, Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 886893

[4] Yu, F. et al, *Three-Dimensional Model Analysis and Processing*, Advanced Topics in Science and Technology in China, 2011, pp. 165

[5] K. Malik, *Handwritten digit recognition*, Machine Learning Summer School, UCSC, 2012

[6] B. Zhang and S. Srihari, *Properties of Binary Vector Dissimilarity Measures*, CVPRIP, 2013

[7] B. Zhang and S.Srihari, *Binary Vector Dissimilarity Measures for Handwriting Identification*, DDR, 2013

[8] A. Cheetham and J. Hazel, *Binary (Presence-Absence) Similarity Coefficients*, Journal of Paleontology, Vol. 43, No. 5 (Sep., 1969), pp. 1130-1136

[9] R. Plamondon and S. Srihari, *On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol 22, no.1, 2000, pp. 63 - 84

[10] A. Malhi and R. Gao, *PCA-Based Feature Selection Scheme for Machine Defect Classification*, IEEE Trans. On Instrumentation and Measurement, vol. 53, no. 6, 2004, pp. 1517 - 1525

[11] E. Pekalska and R. Duin, *"Discover Patterns and Learn from Data."*, Pattern Recognition Tools. N.p., n.d. Web. 11 Dec. 2014.

[12] R. Duin and E. Pekalska, *The dissimilarity space: Bridging structural and statistical pattern recognition* Pattern Recognition Letters 33 (2012) 826832

[13] J. Redfield, *Handwriting Variations in Individuals with MPD*, Dissociation, vol. 4, no. 1, 1991

[14] R.Rao and G.Fung. *On the dangers of Cross-Validation. An Experimental Evaluation*, Proc. SIAM international conference on Data Mining, 2008, pp.588-596.

[15] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, *"Data Preprocessing for Supervised Leaning"*, International Journal of Computer Science, 2006, Vol 1 N. 2, pp 111117.

[16] J. Serra. *Image Analysis and Mathematical Morphology*. London: Academic, 1982. Print.

[17] I. Jolliffe *Principal Component Analysis*, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28

[18] S. Theodoridis and K. Koutroumbas, *Pattern Recognition,* Academic Press,San Diego, 1999.

[19] R. Gonzalez, and R. Woods. Digital Image Processing. Upper Saddle River, NJ: Prentice Hall, 2002.

APPENDIX

This appendix contains a set of tables with the most important results from the experiments performed in the different representations.

Pixel-Based Results

Pixel				
Classifier	Full Pixels	PCA 99%	PCA 85%	Ind. Sel.
knnc	0.262	0.269	0.257	0.326
dtc	0.82	0.814	0.816	0.82
fisher	0.424	0.336	0.264	0.614
ldc	0.9	0.679	0.215	0.885
qdc	0.9	0.9	0.813	0.9
nbyc	0.261	0.707	0.61	0.307
nmc	0.437	0.267	0.269	0.311
nmsc	0.9	0.276	0.242	0.315
perlc	0.342	0.484	0.365	0.464
vpc	0.28	0.301	0.261	0.32
votec	0.273	0.322	0.25	0.323
parzen	0.261	0.262	0.257	0.318
svc	0.231	0.234	0.247	0.404

TABLE XI: Small Dataset Scenario Results

Pixel			
Classifier	Full Pixels	PCA 40D	PCA 85%
knnc	0.108	0.078	0.083
dtc	0.667	0.588	0.588
fisher	0.243	0.165	0.15
ldc	0.898	0.125	0.127
qdc	0.9	0.064	0.759
nbyc	0.196	0.177	0.237
nmc	0.265	0.195	0.195
nmsc	0.192	0.148	0.146
perlc	0.125	0.155	0.183
vpc	0.128	0.12	0.116
votec	0.107	0.122	0.121

TABLE XII: Big Dataset Scenario Results

Dissimilarity-Based Results

Sokal Michener			
Classifier	Full Info	PCA 34D	PCA 99%
knnc	0.296	0.249	0.25
dtc	0.81	0.807	0.807
fisher	0.356	0.212	0.238
ldc	0.19	0.173	0.169
qdc	0.9	0.392	0.9
nbyc	0.37	0.594	0.689
nmc	0.395	0.398	0.396
nmsc	0.378	0.209	0.204
perlc	0.362	0.296	0.317
vpc	0.423	0.356	0.322
votec	0.319	0.251	0.273
parzen	0.244	0.235	0.231
svc	0.641	0.651	0.638
Euclidean			
Classifier	Full Info	PCA 34D	PCA 99%
knnc	0.306	0.264	0.304
dtc	0.81	0.807	0.807
fisher	0.383	0.235	0.238
ldc	0.251	0.183	0.185
qdc	0.9	0.496	0.504
nbyc	0.355	0.568	0.584
nmc	0.405	0.405	0.405
nmsc	0.385	0.215	0.22
perlc	0.339	0.321	0.328
vpc	0.381	0.321	0.352
votec	0.315	0.262	0.284
parzen	0.25	0.236	0.237
svc	0.259	0.272	0.273
RogersTanMoto			
Classifier	Full Info	PCA 34D	PCA 99%
knnc	0.296	0.254	0.253
dtc	0.81	0.807	0.807
fisher	0.356	0.207	0.212
ldc	0.19	0.175	0.172
qdc	0.9	0.389	0.372
nbyc	0.37	0.609	0.597
nmc	0.395	0.4	0.4
nmsc	0.378	0.211	0.21
perlc	0.307	0.336	0.321
vpc	0.37	0.325	0.356
votec	0.319	0.25	0.25
parzen	0.244	0.234	0.235
svc	0.641	0.65	0.651

TABLE XIII: Small Dataset Scenario Results

Sokal Michener			
Classifier	Full Info	PCA 34D	PCA 99%
knnc	0.1	0.099	0.093
dtc	0.654	0.547	0.547
fisher	0.117	0.151	0.136
ldc	0.9	0.121	0.119
qdc	0.9	0.066	0.099
nbyc	0.8	0.176	0.21
nmc	0.9	0.33	0.33
nmsc	0.768	0.139	0.135
perlc	0.15	0.205	0.173
vpc	0.175	0.187	0.188
votec	0.148	0.121	0.112
parzen	0.9	0.091	0.091
svc	0.15	0.15	0.152
Euclidean			
Classifier	Full Info	PCA 34D	PCA 99%
knnc	0.1	0.099	0.1
dtc	0.389	0.623	0.623
fisher	0.292	0.145	0.158
ldc	0.9	0.12	0.127
qdc	0.9	0.066	0.078
nbyc	0.9	0.183	0.197
nmc	0.9	0.34	0.339
nmsc	0.9	0.136	0.143
perlc	0.371	0.246	0.215
vpc	0.172	0.167	0.176
votec	0.303	0.11	0.118
parzen	0.9	0.089	0.09
svc	0.162	0.175	0.181
RogersTanMoto			
Classifier	Full Info	PCA 34D	PCA 99%
knnc	0.094	0.099	0.093
dtc	0.389	0.547	0.547
fisher	0.135	0.151	0.136
ldc	0.9	0.121	0.119
qdc	0.9	0.066	0.099
nbyc	0.801	0.176	0.21
nmc	0.9	0.33	0.33
nmsc	0.792	0.139	0.135
perlc	0.161	0.229	0.173
vpc	0.187	0.183	0.19
votec	0.16	0.121	0.112
parzen	0.9	0.091	0.091
svc	0.165	0.15	0.152

TABLE XIV: Big Dataset Scenario Results

Cross Validation Results for Dissimilarity			
Classifier	SM*34D*ldc	RM*34D*ldc	E*34D*ldc
knnc	0.1235	0.113	0.1358
dtc	0.5797	0.601	0.6682
fisher	0.163	0.1656	0.1481
ldc	0.1254	0.1214	0.1258
qdc	0.1034	0.0933	0.112
nbyc	0.1826	0.1864	0.2141
nmc	0.356	0.3546	0.37
nmsc	0.1512	0.1477	0.1417
perlc	0.2306	0.2381	0.2582
vpc	0.1925	0.1896	0.1956
votec	0.1976	0.1919	0.1559
parzen	0.1089	0.0963	0.129
svc	0.1702	0.1748	0.201

TABLE XV: 10 folds Cross-Validation for Dissimilarity Representation (Big Data Set)

Cross Validation Results for Dissimilarity			
Classifier	SM*34D*ldc	RM*34D*ldc	E*34D*ldc
knnc	0.314	0.308	0.31
dtc	0.8	0.802	0.807
fisher	0.269	0.258	0.287
ldc	0.215	0.219	0.227
qdc	0.45	0.448	0.545
nbyc	0.609	0.628	0.637
nmc	0.318	0.326	0.334
nmsc	0.215	0.217	0.211
perlc	0.373	0.38	0.405
vpc	0.306	0.323	0.318
votec	0.307	0.299	0.299
parzen	0.248	0.244	0.24
svc	0.642	0.644	0.251

TABLE XVI: 10 folds Cross-Validation for Dissimilarity Representation (Small Dataset)

Feature-Based Results (HOG)

HOG 144		
Classifier	Full Info	PCA 38D
knnc	0.02	0.02
dtc	0.54	0.43
fisher	0.04	0.06
ldc	0.04	0.04
qdc	0.1	0.05
nbyc	0.06	0.05
nmc	0.11	0.12
nmsc	0.9	0.05
perlc	0.08	0.08
vpc	0.05	0.02
votec	0.02	0.02

TABLE XVII: Big Dataset Results

HOG 144			
Classifier	Full Info	PCA 60D	PCA 38D
knnc	0.141	0.143	0.152
dtc	0.812	0.743	0.743
fisher	0.602	0.108	0.139
ldc	0.574	0.12	0.119
qdc	0.9	0.9	0.624
nbyc	0.155	0.573	0.501
nmc	0.119	0.119	0.121
nmsc	0.9	0.087	0.111
perlc	0.132	0.348	0.25
vpc	0.146	0.173	0.167
votec	0.179	0.143	0.154
parzen	0.139	0.127	0.13
svc	0.124	0.122	0.142

TABLE XVIII: Small Dataset Results