Evaluation of Machine Learning Techniques for Face Detection and Recognition

E. García Amaro, M.A. Nuño-Maganda and M. Morales-Sandoval
Polytechnic University of Victoria
Information Technology Department
Scientific and Technological Park of Tamaulipas, Mexico
{egarciaa,mnunom,mmoraless}@upv.edu.mx

Abstract

Biometric identification (BI) is one of the most explored topics in recent years. One of the most important techniques for BI is face recognition. Face recognition systems (FRSs) are an important field in computer vision, because it represents a non-invasive BI technique. In this paper, a FRS is proposed. In the first step, a face detection algorithm is used for extracting faces from video frames (training videos) and generating a face database. In a second step, filtering and preprocessing are applied to face images obtained in the previous step. In a third step, a collection of machine learning algorithms are trained using as input data the faces obtained in the previous step. Finally, the classifiers are used for classify faces obtained from video frames (test videos). The obtained results shows the suitability of this approach for analyzing large collections of videos where previous face labels are not available.

1. Introduction

Biometrics is the science of establishing the identity of an individual based on the physical, chemical or behavioral attributes of the person. The relevance of biometrics in modern society has been reinforced by the need for large-scale identity management systems whose functionality relies on the accurate determination of an individual identity in the context of several different applications [1]. Facial scan is an effective biometric attribute/indicator. Different biometric indicators are suited for different kinds of identification applications due to their variations in intrusiveness, accuracy, cost, and ease of sensing [2].

Face recognition systems has received substantial attention from researchers in biometrics, pattern recognition, and computer vision communities [3, 4, 5]. There are urgent needs for face recognition in many practical applications, such as security monitoring, surveillance system and biometrics identified system. Common face detection and recognition systems consist of single route image acquisition module with one camera. A face recognition system uses a database of images and compares another image against those to find a match, if one exists. Machine Learning (ML) is a branch of computer science that is concerned with designing systems that can learn from the provided input. Usually the systems are designed to use this learned knowledge to better process similar input in the future. A ML algorithm is one that can learn from experience (observed examples) with respect to some class of tasks and a performance measure [6]. Classification which is also referred to as pattern recognition, is an important task in ML, by which machines "learn" to automatically recognize complex patterns, to distinguish between exemplars based on their different patterns, and to make intelligent decisions. Pattern recognition techniques have been applied to face recognition [7]. A typical approach assumes that exists a face database, but when this database does not exists, the construction of this database is not trivial. In fact, automatic labeling is an open issue in face recognition [8, 9, 10].

Face recognition methods for intensity images fall into two main categories: feature-based and holistic. Feature-based approaches first process the input image to identify and extract (and measure) distinctive facial features such as the eyes, mouth, nose, etc., and then compute the geometric relationships among those facial points, thus reducing the input facial image to a vector of geometric features. One of the earliest such attempts was by Kanade [11], who employed simple image processing methods to extract a vector of 16 facial parameters. More sophisticated feature extraction techniques have been explored in [12, 13, 14]. Holistic approaches attempt to identify faces using global representations, i.e., descriptions based on the entire image rather than on local features of the face. AI approaches utilize tools such as neural networks and machine learning techniques to recognize faces [15, 16, 17]. Since the performance of any classifier is more sensitive to some factors and relatively invariant to others, a recent trend has been to combine individual classifiers in order to integrate their complementary

978-1-61284-1325-5/12/$26.00 ©2012 IEEE

information and thereby create a system that is more robust than any individual classifier to variables that complicate the recognition task. Such systems have been termed as multiple classifier systems (MCSs) [18] and are a very active research area at present.

In this paper, a system that generates a face database, trains several ML algorithms and evaluates each one when they are applied to face recognition target task is proposed. This paper is organized as follows: in section 2, background of both face detection and face recognition are detailed. In section 3 the proposed methodology for face detection and recognition is described, and the modules proposed for implementing both stages are presented. In section 4, experimental results are described. Finally in section 5, the concluding remarks and further work directions are presented.

2. Background

2.1. Face Detection and Recognition

Considering an image representing a frame taken from a video stream or a graphic file selected from a database; the problem of face detection consist of finding the spatial location within the scene where human faces are located. This problem is quite challenging due numerous issues, e.g. pose, presence or absence of structural components, facial expression, occlusion, orientation, imaging conditions. Approaches used for face detection are [3]: knowledge-based methods, feature invariant methods, template matching methods and appearance-based methods.

Face recognition is the problem of identifying or verifying a specific individual from a digital image or video frame, rather than merely detecting the presence of a human face (face detection). The general term face recognition can refer to a number of different problems including, such as: Given two face images, determine where both are the same person or not,; Given a face image, decide whether it is an example of a particular individual,; Given a face image, decide which person from among a set of people represents, if any. Several methods have been proposed. One of the most common are based on ML, where a set of images of individuals and its labels are used for training a classifier, obtaining a knowledge model, and later data for a new individual can be passed among the model, obtaining an accuracy identification.

2.2. ML Algorithms

During the past decades, several ML algorithms have been proposed for classification tasks. Although the potential advantages and disadvantage of these techniques have been addressed in many published work, most of them are from the theoretical view under some assumption about data distribution, characteristics of the classification task, signal-to-noise-ratio, etc. In reality, these assumptions are often hard to be verified. Therefore, a practical solution for selecting an appropriate model for a given classification task is to experimentally compare these algorithms. In this paper, we compared five widely used machine classifiers which are implemented in Weka [19, 20]: K-Nearest Neighbor (KNN), Locally-Weighted Learning (LWL), Naive Bayes classifier (NB), Decision Table Classifier (DT) and Single Decision Tree (SDT).

K-Nearest Neighbor (KNN). KKN is a method for classifying objects based on closest training examples in the feature space. k-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The nearest neighbor class of estimators adapts the amount of smoothing to the local density of data. The degree of smoothing is controlled by k, the number of neighbors taken into account, which is much smaller than N, the simple size. Let us define a distance between a and b, for example $|a-b|$, and for each x, we define $d_1(x) \leq d_2(x) \leq ... \leq d_N(x)$ to be the distances arranges in ascending order, from x to the points in the sample: $d_1(x)$ is the distance to the nearest sample, $d_2(x)$ is the distance to the nearest, and so on. If x^t are the data points, then we define $d_1(x) = min_t\,|x - x^t|$ and if i is the index of the closest sample, namely, $i = arg\ min_t\,|x - x^t|$, then $d_2(x) = min_{j \neq i}\,|x - x^t|$, and so forth ... The KNN classifier assigns the input to the class having most examples among the k neighbors of the input. All neighbors have equal vote, and the class having the maximum number of voters among the k neighbors is choosen. Tie are broken arbitrarily or a weighted vote is taker. k is generally taken an odd number to minimize ties: Confusion is generally between two neighboring classes.

Locally-Weighted Learning (LWL). LWL is a form of lazy learning and memory-based learning focused on locally weighted linear regression. Locally weighted regression (LWR) is a memory-based method that performs a regression around a point of interest using only training data that are "local" to that point. One recent study demonstrated that LWR was suitable for real-time control by constructing an LWR-based system that learned a difficult juggling task. Given a new instance x_q, the general approach in LWR is to construct an approximator $\hat{f}$ thet fits the training examples in the neighborhood surrounding x_q. This approximator is then used to calculate the value $\hat{f}(x_q)$, which is output as the estimated target value for the query instance. Let us consider the case of LWR in which the target function f is approximated near x_q using a linear equation of the form: $\hat{f} = w_0 + w_1 a_1(x) + w_2 a_2(x) + ... + w_n a_n(x)$, where $a_i(x)$ denotes the value of the ith attribute of the instance x. There are several methods to find the coeffients $w_0 ... w_n$ to mini-

mize the error in fitting such linear functions to a given set of training examples. The most commonlyu sed error minimization is given by equation 1

$$E(x_q) = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x)) \qquad (1)$$

Where the squared error is minimized over the entire set D of training examples, while weighting the error of each training example by some decreasing function K of its distance from x_q.

Naive Bayes classifier (NB). The NB is based on conditional probabilities. It uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an $n - dimensional$ attribute vector, $X = (x_1, x_2, ..., x_n)$, depicting n measurements made on the tuple from n attributes, respectively, $A_1, A_2, ..., A_n$. Suppose that there are m classes, $C_1, C_2, ..., C_m$. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naive Bayesian classifier predicts that tuple X belongs to the class C_i if and only if $P(C_i|X) > P(C_j|X)$ for $1 \leq j \leq m, j \neq i$. Thus we maximize $P(C_i|X)$. The class C_i for which $P(C_i|X)$ is maximized is called the maximum posteriori hypothesis. By Bayes theorem:

$$P(C_i|X) = \frac{P(X|C_i)\ P(C_i)}{P(X)} \qquad (2)$$

As $P(X)$ is constant for all classes, only $P(X|C_i)P(C_i)$ need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1) = P(C_2) = ... = P(C_m)$, and we would therefore maximize $P(X|C_i)$. Otherwise, we maximize $P(X|C_i)P(Ci)$. The class prior probabilities may be estimated by $P(C_i) = |C_i, D|/|D|$, where $|C_i, D|$ is the number of training tuples of class C_i in D. In order to predict the class label of X, $P(X|C_i)P(C_i)$ is evaluated for each class C_i. The classifier predicts that the class label of tuple X is the class C_i if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j)\ for 1 \leq j \leq m, j \neq i \qquad (3)$$

In other words, the predicted class label is the class C_i for which $P(X|C_i)P(C_i)$ is the maximum.

Single Decision Tree (SDT). Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, where each internal node (nonleaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node. Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules. Most algorithms for decision tree induction follow a top-down approach, which starts with a training set of tuples and their associated class labels. The training set is recursively partitioned into smaller subsets as the tree is being built. An attribute selection measure is a heuristic for selecting the splitting criterion that "best" separates a given data partition, D, of class-labeled training tuples into individual classes. If we were to split D into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be pure. The specific algorithm for generating the decision tree is called C4.5 algorithm.

Decision Table (DT). A DTM is a predictive modeling tool that performs classification. The Decision Table Model (DTM) does not assume that the attributes are independent. The DT induces a model that presents correlations between pairs of attributes in a layered hierarchy. Decisions are made by the inducer in the same way as the SDT, but attributes are evaluated across the entire level of the tree rather than on a specific sub-tree. The result is then presented as a hierarchical table rather than a tree.

3. Proposed System

The main blocks of the proposed system are described in this section. In figure 1, an schematic of the main block of the proposed system is shown.

3.1. Face Detection

For this step, a version of the Viola-Jones face detector [21] implemented on OpenCV [22] was used for face detection. Faces were detected using the function *cvHaarDetectObjects*, using the provided Haar Classifier Cascade. with scale factor set to 1.2, *min_neighbors* set to 2, and the flag set to *CV_HAAR_DO_CANNY_PRUNING*. The *Semi-Aided Labeling Module (SALM)* reads the input video, and for each frame where at least one face was detected by the face detection module, it asks to the user for labeling each face. Once the label has been read, an image is created using as prefix the label assigned to the face contained into the frame. This process is repeated until each video ends. The SALM is executed for each one of the input videos located in the current directory.

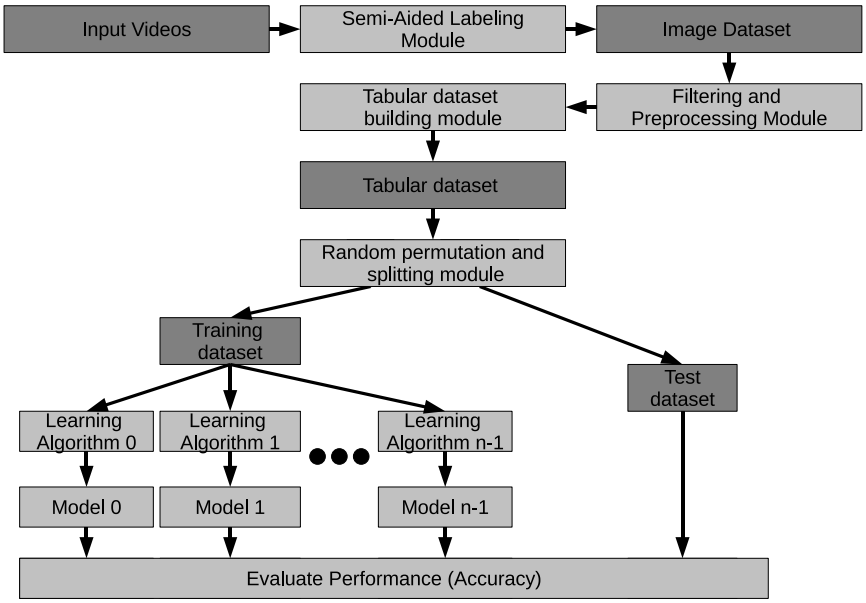

Figure 1. Block diagram of the proposed face detection and recognition system

3.2. Filtering and Preprocessing

This step is performed by the **Filtering and Preprocessing Module (FPM)**. This module performs the following transformations:

- RGB to Gray scale Transformation. For reducing the amount of data to be processed, a 24-bit per pixel RGB format is transformed into a 8-bit per pixel gray-scale format.
- Scaling. The face images are scaled to a fixed number of rows and columns. The output resolution for each face can be set by user according to the required accuracy.
- Histogram Equalization. This method increases the global contrast, so the intensity can be better distributed on the image histogram. This process is helpful for improving face recognition accuracy.

When the FPM has finished, a process called **Tabular Dataset Building Module (TDBM)** is executed. This module obtains the image pixels, and generates a tabular dataset where rows are the total number of subjects, and the columns are the image pixels. The final column represents the class attribute, the label assigned by user when the SALM was executed. The TDBM generates a Matrix of M by $N + 1$, where M is the total number of detected faces on the database, N is the total pixel count depending of the face image resolution (i. e, for a 25 x 25 image, 625 columns are required for storing the image pixels). The $(N+1)-th$ column stores the label given to that image face. Data are stored using an ARFF (Attribute-Relation File Format) format. An ARFF file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka ML software [20].

3.3. Training

For performing the training of the classification algoritms, the following steps are required:

- Permute and split dataset. This operation is performed by the **Random Permutation and Splitting Module (RPSM)**. Basically, a random permutation of the samples contained in the tabular dataset is performed, and the resulting dataset is divided into two datasets: the training dataset and the test dataset.
- Train classification algorithms. Each classification algorithm takes as input the training datset generated by the RPSM, and performs the model building for each classifier. Later, the model for each classifier is stored in disk for use it later in the classification step.

3.4. Classification

In this module, with the help of the previous trained classifiers, takes as input the faces from the test set, applies filter and prepreprocessing operators, and evaluates the test face in each model generated by the trained classifiers. After doing this comparison, face image is classified with the

Figure 2. Dataset generated by the Semi-Aided Labeling Module

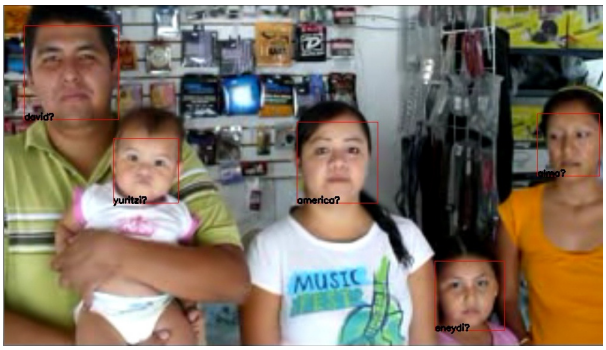

Figure 3. Screenshot for the Recognition Application

label or name predicted by each classified. The output of each classified is processed by the Performance Evaluation Module (PEM), which generates a table with a comparison among several classifiers.

3.5. Integration of the proposed modules

The main module is written in C, using OpenCV functions for image capture, display, storing and preprocessing. The OpenCV version used for the proposed system is 2.1, while the Weka version is 3.6. The modules implemented in C are: SAL, FPM, TDBM and PEM. When the TDBM finishes, then a set of Java programs (trainer programs) that performs the training for each one of the classifiers is invoked. Each program performs the training using as training set the input data set generated by the TDBM. The model generated by each classifier is stored in disk. For the classification step, other set of Java programs (test programs) are invoked. Each test program reads the model generated by each classifier, passes the face data to be classified to the classifier module, and generates the predicted label for each input face. Finally, the output frame containing the faces to be classified is shown on a window, including in each face a label with the name of the identified subject.

4. Results

For the reported results, two different videos of 10-second duration were used. A total of $10x30x2 = 600$ frames where processed. In the input video, there was 6 different individuals, representing a total of $3,600$ samples (600 for each individual). Three versions of the dataset were generated: one for a 100 x 100 pixels face resolution, one for a 50 x 50 pixels face resolution, and finally one for a 25 x 25 pixels face resolution. In figure 2, a mosaic for the frames obtained from one video is shown. The classifiers used are: KNN, LWL, Naive Bayes, Decision Tree and Decision Table (The Weka implementation of C4.5 algorithm is called J48). For each version of the dataset, the algorithms described previously were trained and the model was obtained. The accuracies obtained for the trained algorithms are shown in table 3.4. In figure 3, a screen-shot of the proposed face identification system is shown.

4.1. Discussion

From the accuracies reported in table 3.4, it can be shown that the higher face image resolutions, the best obtained accuracy results, but some issues can be highlighted. For the KNN and J48 algorithms, high accuracies results are obtained when the low resolution version of input faces are used. For methods with high accuracies obtained when using as input low resolution images, it can be possible to make a depth research about the parameters of the classifiers for improving accuracy.

5. Conclusion and Future Work

This paper evaluate the suitability of both computer vision and ML techniques for solving the problem of face detection and recognition. The use of a standard and well-known technique for face detection was applied for generating a small face database, and the use of the generated database for training of several ML techniques off-line for obtaining several models is reported. Finally, the use of the models generated by each classified are validated thought using these models for face recognition in videos where the subjects are not previously labeled.

Dimension	KNN	LWL	Naive Bayes	Decision Table	Decision Trees
25 x 25	80.0%	55.0%	66.6%	58.3%	83.3%
50 x 50	80.0%	56.6%	30.0%	51.6%	20.0%
100 x 100	98.3%	78.3%	100-0%	85.0%	100.0%

Table 1. Comparison of overall classification accuracies

As future work, four main directions must be addressed: In a first step, the integration of a more reliable face detector which copes with several problems shown by the Haar detectors must be considered. In a second step, to test with more ML algorithms for concluding which methods performs the best classification results. In a third step, to test with ensemble learning for improving recognition accuracy. Finally, in a fourth step, to implement the ML techniques in OpenCV, for compacting the current face detection and recognition system.

Acknowledgments

First author thanks to the National Council for Science and Technology (CONACyT) for the financial support granted for her postgraduate studies. The second and third authors thank to PROMEP (Programa de Mejoramiento para el Profesorado) for supporting this research work.

References

[1] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of Biometrics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[2] D. Bhattacharyya, R. Ranjan, F. Alisherov, and M. Choi, "Biometric authentication: A review," *International Journal of u- and e- Service, Science and Technology*, vol. 3, no. 2, pp. 23–27, 2009.

[3] C. Zhang and Z. Zhang, "A survey of recent advances in face detection," *Learning*, no. June, p. 17, 2010.

[4] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 34–58, January 2002.

[5] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Comput. Surv.*, vol. 35, pp. 399–458, December 2003.

[6] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[8] G. B. Huang, M. Ramesh, T. Berg, and E. L. Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," Tech. Rep. University of Massachusets Amherst, 07-49, Oct. 2007.

[9] D. S. Bolme, J. R. Beveridge, and B. A. Draper, "Facel: Facile face labeling," in *ICVS* (M. Fritz, B. Schiele, and J. H. Piater, eds.), vol. 5815 of *Lecture Notes in Computer Science*, pp. 21–32, Springer, 2009.

[10] Y. Li, G. Su, and Y. Shang, "Generating optimal face image in face recognition system," in *ICASSP (1)*, pp. 793–796, IEEE, 2007.

[11] T. Kanade, "Picture processing system by computer complex and recognition of human faces," in *doctoral dissertation, Kyoto University*, November 1973.

[12] *Feature extraction from faces using deformable templates*, 1989.

[13] C. Colombo, A. Del Bimbo, S. De Magistris, V. Cantoni, L. Lombardi, M. Mosconi, M. Savini, and A. Setti, "Human-computer interaction based on eye movement tracking," in *Proceedings of the Computer Architectures for Machine Perception*, CAMP '95, (Washington, DC, USA), pp. 258–, IEEE Computer Society, 1995.

[14] I. Craw, D. Tock, and A. Bennett, "Finding face features," in *Computer Vision ECCV'92* (G. Sandini, ed.), vol. 588 of *Lecture Notes in Computer Science*, pp. 92–96, Springer Berlin / Heidelberg, 1992.

[15] A. Eleyan and H. Demirel, "Face recognition system based on pca and feedforward neural networks," in *IWANN* (J. Cabestany, A. Prieto, and F. S. Hernández, eds.), vol. 3512 of *Lecture Notes in Computer Science*, pp. 935–942, Springer, 2005.

[16] B. Li and H. Yin, "Face recognition using rbf neural networks and wavelet transform," in *Proceedings of the Second international conference on Advances in neural networks - Volume Part II*, ISNN'05, (Berlin, Heidelberg), pp. 105–111, Springer-Verlag, 2005.

[17] U. H.-G. Kreßel, *Pairwise classification and support vector machines*, pp. 255–268. Cambridge, MA, USA: MIT Press, 1999.

[18] N. Oza, R. Polikar, J. Kittler, and F. Roli, *Multiple Classifier Systems*, vol. 3541 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.

[19] I. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. Cunningham, "Weka: Practical machine learning tools and techniques with java implementations," 1999.

[20] Weka Machine Learning Project, "Weka." URL http://www.cs.waikato.ac.nz/˜ml/weka.

[21] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, pp. 137–154, May 2004.

[22] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media Inc., 2008.