Evaluation of Face Recognition Techniques for Application to Facebook

Brian C. Becker
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
briancbecker@cmu.edu
http://www.briancbecker.com

Enrique G. Ortiz
University of Central Florida
4000 Central Florida Blvd.
Orlando, FL 32816
eortiz@cs.ucf.edu
http://www.enriquegortiz.com

Abstract

This paper evaluates face recognition applied to the real-world application of Facebook. Because papers usually present results in terms of accuracy on constrained face datasets, it is difficult to assess how they would work on natural data in a real-world application. We present a method to automatically gather and extract face images from Facebook, resulting in over 60,000 faces representing over 500 users. From these natural face datasets, we evaluate a variety of well-known face recognition algorithms (PCA, LDA, ICA, SVMs) against holistic performance metrics of accuracy, speed, memory usage, and storage size. SVMs perform best with ~65% accuracy, but lower accuracy algorithms such as IPCA are orders of magnitude more efficient in memory consumption and speed, yielding a more feasible system.

1. Introduction

Face recognition is a popular research topic, maturing as researchers develop sophisticated algorithms to achieve more accurate results on increasingly difficult face datasets [1]. Since the original "eigenface" approach [2], other techniques such as Fisherfaces [3], Independent Component Analysis [4], and Support Vector Machines [5] have been proposed. More difficult face databases, such as the FERET [6] database, introduce variations in pose, illumination, expression, and time lapses, leading to recent advances such as 3D modeling techniques [7].

With such advances in face recognition and algorithms claiming accuracies of greater than 90%, one wonders how these algorithms would fare on a real system with real data. Consider Facebook, a popular social networking website with over 80 million people [8]. Users upload a staggering 14 million photos per day, representing an opportunity to evaluate face recognition systems. This source of real-world faces is nearly untapped by the face recognition field. Labeled Faces in the Wild (LFW) [9], a face database created from Internet photos, is similar but focuses on pair matching rather than recognition.

Often the field of face recognition pursues the highest possible accuracy at the cost of additional computation or memory usage. However, accuracy represents only one facet of system performance. Consider again Facebook, where users "tag" the identity of people in shared photos. This manual "tagging" is tedious, prompting us to ask if face recognition can successfully automate the process. To adequately service such a large website, an auto-tagging system must not only exhibit high accuracy, but high speed, low memory consumption, and scalability. Ruiz-del-Solar and Navarrette in [10] provide an excellent review of face recognition algorithms, but only evaluate them for accuracy.

Our contribution is utilizing a new source of real-world data to evaluate several face recognition algorithms' overall performance for application to Facebook. Section 2 describes our method for automatically constructing face databases. Section 3 briefly overviews a variety of well-known algorithms that section 4 evaluates using a holistic metric of accuracy, speed, memory usage, and storage size. Section 5 reports performance characteristics and suitability to an application like Facebook. Section 6 concludes and discusses future work.

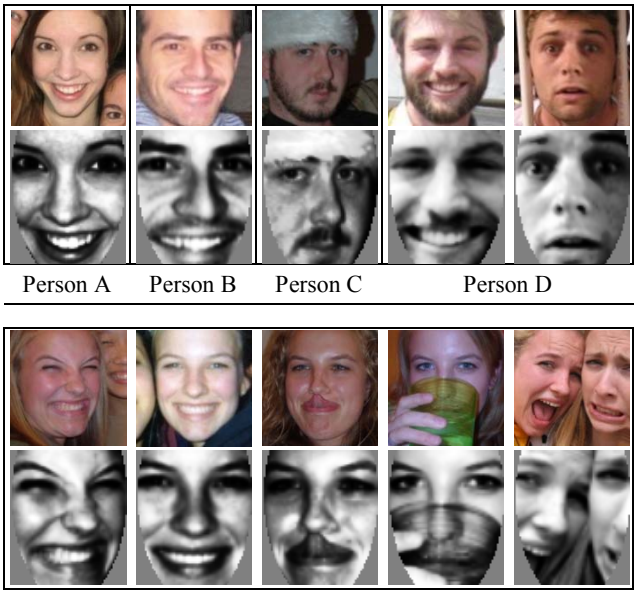

Figure 1: Typical faces automatically extracted from Facebook. Top rows: raw color faces. Bottom rows: Corresponding preprocessed face image. Bottom right photo pair represents a false positive. Permission was obtained to publish these photos.

978-1-4244-2154-1/08/$25.00 ©2008 IE

2. Facebook Data Acquisition

Each Facebook user can upload photos and place location tags to identify friends in the picture. Uploaded images are visible to the user's friends, but more importantly, Facebook allows third-party applications to access pictures and tags through an Application Programmer's Interface (API). Using this feature, we automatically downloaded tagged photos from seven different Facebook accounts and their friends.

2.1. Face Extraction Approach

To construct a dataset of faces for each Facebook account, it was necessary to detect, identify, and extract faces from thousands of raw tagged photos. For simplicity, we focused on frontal faces detected using the Viola-Jones face detector [11] implemented in Intel's OpenCV library. Because of the large number of false positives encountered, eye detection [12] was employed to augment the face extraction. Detected faces were only accepted if they met certain geometric criteria that relate the presence and location of both eyes $e(l), e(r)$ to the face f:

$$C\big(f, e(l), e(r)\big) = S(l) + S(r) + S_b \tag{1}$$

$$S(i) = \left(\frac{2f_{size}}{e(i)_{size}} - \alpha\right)^2 + \left(\frac{|f_x - e(i)_x|}{0.01 f_{radius}} - \beta_x\right)^2 + \left(\frac{|f_y - e(i)_y|}{0.01 f_{radius}} - \beta_y\right)^2 \tag{2}$$

$$S_b = \left(\frac{f_{size}}{e(l)_{size}} - \frac{f_{size}}{e(r)_{size}}\right)^2 + \left(\frac{d\big(e(l), e(r)\big)}{0.01 f_{radius}} - \gamma\right)^2 \tag{3}$$

The cost function C calculates the score for each combination of face f and left/right eyes $e(l)/e(r)$. The function d measures Euclidean distances. Both faces and eyes are modeled as circles; f_x, f_y refer to the center of the circle representing the face. Through experimental observation using available Facebook images, appropriate values of $\alpha = 20,\ \beta_x = \beta_y = 25,\ \gamma = 75$ were chosen, which correspond to the eyes residing in the center of the upper quadrants of the face. The system only accepts the face if the cost is below a threshold, usually a value between 2000-5000 depending on the desired strictness.

Once accepted, a face is identified if a unique tag of a friend is located within the circle representing the face. The face is subsequently rotated by aligning the eyes horizontally. The system extracts the face and normalizes it with the procedure developed by [13], namely by a grayscale conversion, elliptical mask crop, histogram equalization, and pixel value normalization. The final image is resized to 56x64 pixels. The entire processing of a face from a tagged image takes 1-3 seconds on average, which could easily be run client-side as the image is uploaded to Facebook.

2.2. Real-World Non-idealities

By evaluating the permutations of face and eye pairs, the above method automatically constructs the datasets. Unlike artificial datasets that introduce realism with scarves, hats, etc., our datasets exhibit a realism almost approaching surrealism. Face poses are largely frontal but contain up to ±30° variations in all directions. Indoor and outdoor conditions result in volatile illumination, further complicated by the usage of many different cameras. Additionally, differences over time such as the growth of beards (i.e. Person D in Figure 1), stylization of hair, and digital alterations add to the complexity of the dataset. To gauge the difficulty of the datasets, we manually analyzed the small BCB dataset. Out of 914 pictures, we found 17 images that were out of alignment, partially occluded, digitally altered, or improperly tagged.

2.3. Resulting Facebook Datasets

We gathered pictures and tags from seven different Facebook accounts and applied the techniques described to automatically construct seven different face datasets. The statistics for each are described in Table 1 and compared to the AT&T Database of Faces [14]. See Figure 1 for sample faces from various datasets.

Table 1: Statistics for seven Facebook datasets, including the number of friends each user has, the number of people in the database (friends who had 10 or more extracted face images), the number of photos uploaded amongst all friends, and the number of faces detected, identified, and extracted for the dataset.

Dataset	Friends	People	Total Pictures	Total Faces
ATT	N/A	40	N/A	400
ACR	81	10	1,974	282
BCB	51	22	4,880	839
JBK	123	65	20,109	4,009
EGO	355	148	34,117	7,950
RGA	407	215	44,403	12,394
LLP	396	264	85,450	17,602
SSB	335	222	60,553	18,599

3. Face Recognition Approaches

To achieve our goal of applying face recognition to the real-world problem of Facebook, we evaluate well-known, classical algorithms: PCA, ICA, LDA, and SVMs.

3.1. Principal Component Analysis (PCA)

One of the oldest, most reputed face recognition algorithms is Eigenfaces [2], which is based on Principle Component Analysis (PCA). PCA reduces the dimensionality of each face image by exploiting similarities between all face images. Thus, PCA seeks to extract a set of images (eigenfaces) that combine linearly

to describe all face images. Given M face images arranged as column vectors $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$, the average face $\Psi = \frac{1}{M}\sum_{n=1}^{M} \Gamma_n$ is subtracted from each image $\Phi_i = \Gamma_i - \Psi$. Joining the face images into a single matrix $A = [\Phi_1\ \Phi_2 \ldots \Phi_M]$, PCA finds a set of orthonormal vectors that best represents the data. These vectors are the eigenvectors u_k of the covariance matrix $C = AA^T$. In Eigenspace terminology, each face image is projected by the top M' significant eigenvectors u_k to obtain weights $w_k = u_k^T(\Gamma - \Psi)$ that best linearly weight the eigenfaces into a representation of the original image. Knowing the weights of the training images and a new test face image, a nearest neighbor approach determines the identity of the face. Eigenfaces has the advantage of being simple and fast at the cost of low accuracy when pose, expression, and illumination vary significantly.

3.1.1 Incremental PCA (IPCA)

As the Facebook scenario is an incremental training process where users continuously add new tagged photos, we chose to also explore Covariance-Free Incremental Principle Component Analysis (CCIPCA) [15]. By processing one image at a time, CCIPCA incrementally estimates the eigenvectors of the covariance matrix that would normally be calculated from all images. IPCA requires far less memory and is often faster with a slight degradation in accuracy.

3.1.2 Individual IPCA

As IPCA incrementally updates the eigenfaces, the weights for previously trained images become invalid because the eigenspace in which they reside has been changed. Consequently, the previously trained face images must be kept and reprojected into the updated eigenspace. Liu et al. in [16] describes a fully incremental method that does not require trained face images for updates. Instead of creating a single eigenspace to represent all faces, a smaller set of 5-10 eigenfaces is created for each individual person. Recognition is not done by nearest neighbor, but by projecting the unknown face into each user's eigenspace and finding the one that best represents the face (determined by the minimum residual error). Training times are very fast and fully incremental, but accuracy is poorer and recognition is slower.

3.2. Independent Component Analysis (ICA)

Similar to PCA, Independent Component Analysis (ICA) seeks a set of vectors that reduces the dimensionality of input images [4]. However, ICA does not require the orthonormalization of vectors, which allows higher-order dependencies in image pixels to be exploited. As the mean (first-order statistic) is removed from the images in PCA, ICA removes the first and second-order statistics by "sphering" the data. Each image (with the mean subtracted) is stored as a row vector in X, which is multiplied by the whitening matrix $W_z = 2cov(X)^{-1/2}$. ICA finds statistically independent images, represented by the rows in matrix U, that are mixed together with matrix W such that $U = WX$. In comparison to PCA, the rows of U are analogous to eigenfaces and the columns of W^{-1} are the weights of each image. ICA can account for more variations in the input images, but suffers from slower performance.

3.3. Linear Discriminant Analysis (LDA)

One of the failings in PCA and ICA is that the distances between weights from faces of the same person are greater than face weights from different people. To correct this, a method called Fisherfaces [3], based on Linear Discriminant Analysis (LDA), attempts to find vectors that not only describe the data, but also best discriminate between classes of data. Given c classes (people) with the mean of class j denoted by μ_j and the ith image in class j denoted by x_i^j, a "within-class" scatter matrix S_w and a "between-class" scatter matrix S_b is calculated.

$$S_w = \sum_{j=1}^{c}\sum_{i=1}^{N_j}(x_i^j - \mu_j)(x_i^j - \mu_j)^T \quad (4)$$

$$S_b = \sum_{j=1}^{c}(\mu_j - \mu)(\mu_j - \mu)^T \quad (5)$$

LDA creates a set of projection vectors by using these scatter matrices to maximize the between-class measure while simultaneously minimizing the within-class measure. Literature shows that LDA is often superior to PCA for well distributed classes in small datasets [3]. Since LDA requires significantly more computation than PCA for large datasets, we investigated the use of incremental LDA (ILDA) [17], which is also efficient in batch mode.

3.4. Support Vector Machines (SVMs)

Recently, Support Vector Machines (SVMs) have received much attention for their applicability in solving pattern recognition problems. SVMs were first proposed as a binary classifier. SVMs compute the support vectors through determining a hyperplane that maximizes the margin, or distance, between the hyperplane and the closest points. As described in [18], the problem begins with a set of N points $\boldsymbol{x_i} \in \mathbb{R}^n$, $i = 1, 2, \ldots, N$. Each point is labeled as one of two classes $\boldsymbol{y_i} \in \{-1, 1\}$. The best, or optimal separating hyperplane (OHS), is defined as $f(x) = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x_i} \cdot \boldsymbol{x} + b$, where the sign of $f(x)$ determines the class of the data. For the non-separable case and solving for the coefficients α_i and b refer to [5].

We chose one-vs-one recognition strategy as initial tests yielded better performance compared to one-vs-all. The one-vs-one technique uses binary tree classification to expand to a multi-class scenario, where at each level two

classes are compared and the top class is selected with:

$$d(x) = \frac{\sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x_i} \cdot \boldsymbol{x} + b}{\|\alpha_i y_i \boldsymbol{x_i} \cdot \boldsymbol{x}\|} \quad (6)$$

where the sign of d is the class. Linear, second-order polynomial, and radial basis function (RBF) kernels were explored for application with the Facebook data. A smaller dataset, BCB, was used to choose the best kernel and determine the parameters that maximized accuracy using the RBF and second-order kernels. The second-order kernel obtained very poor accuracy, while the RBF and linear kernels achieved comparable results. A linear kernel is assumed for the rest of this paper.

4. Results

When possible, we chose preexisting algorithm implementations that have been tested and accepted by the community. For LDA, we used [17]'s code; for IPCA, we used code accompanying [15]. LIBSVM [19] is a popular SVM library. For ICA, we interfaced to the Architecture I code from [4]. Also, we hybridize SVM with other architectures as in [5]. All work was done in MATLAB®, except LIBSVM, which is a C library.

4.1. Experimental Setup

To ensure a useful interpretation of results, we maintain a consistent setup between experiments. We chose seven different Facebook datasets to represent a wide spread of typical Facebook users, from heavy users with hundreds of friends to casual users with only a few dozen friends. As described earlier, the extraction and preprocessing of faces were identically and automatically performed to construct each dataset. Additionally, we included the unmodified AT&T dataset [14] as a quick, baseline benchmark, since detailed analysis of selected algorithms has been done on other standard datasets such as FERET [6]. Each dataset was randomly divided into 60% training and 40% testing for validation. Tests showed different selections of random training/testing sets changed the results by less than a percent on average. To ensure sufficient training images exist, people with <10 images were discarded.

Each algorithm is evaluated with several criteria: accuracy, memory usage during the training and testing phases, time spent to train and test, and the size of the model (i.e. eigenfaces for PCA or support vectors for SVMs). While implementations differ in optimizations and efficiency, the results provide a realistic estimate. All experiments were performed on an Intel® Core 2 Duo 2.6 GHz computer with 3 GB of RAM. Running times do not include loading images as a real implementation may store them in a database, on a disk, or over a network. Running times are normalized by the size of the dataset and thus listed in milliseconds (ms) per image. Overall memory (RAM) consumption is sampled at approximately 10 Hz and averaged over the algorithm phase (train or test). Unless the algorithm processes faces incrementally, memory consumed by the batch training images is included in the training phase.

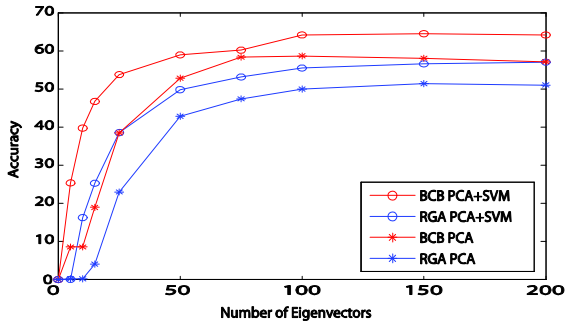

Figure 2: Varying the number of eigenvectors for a small (BCB) and large (RGA) dataset to determine a satisfactory threshold.

To choose the number of eigenvectors to use for subspace algorithms, we varied eigenvectors for a small and large dataset, BCB and RGA respectively. Since accuracy levels out around 75-150 in both cases, PCA/ICA use 100 basis vectors. To reduce the effect of illumination in PCA/ICA, the first 10 eigenfaces are discarded based on findings in [1] and several Facebook test runs. Individual IPCA methods discard the first eigenface for better accuracy as well.

4.2. Results

For each algorithm, we present a tabular form of the system characteristics for each dataset. Tables 2-7 are arranged by evaluation criteria so an interested user can scan and directly compare datasets and approaches.

4.2.1 Analysis and Evaluation

As expected, the results show that the system performance varies significantly and accuracy on real data is lower than reported on most face datasets. In a typical face recognition paper focused on accuracy, we would conclude that a SVM approach is best. However, for Facebook users with many friends, SVMs require half a gigabyte of memory, over half an hour to train or classify, and a half a gigabyte for storage. Without a very large cluster, deploying SVMs to service any significant part of the Facebook population is simply not feasible.

By sacrificing accuracy, other approaches are much more feasible. PCA, the traditional "eigenface" algorithm, fares well on the ATT & ACR dataset but begins performing very poorly on the larger Facebook datasets. As the first 10 eigenfaces encode illumination changes, discarding them significantly improved the accuracy, as shown in the Table 2 for datasets BCB through SSB. While training time and storage are more reasonable than SVMs, the memory requirements for training are high

Table 2: Accuracy across approaches and datasets (percent)

	ATT	ACR	BCB	JBK	EGO	RGA	LLP	SSB
PCA	93.1	69.7	57.2	56.5	52.8	50.0	47.1	56.2
IPCA	94.4	65.1	56.0	56.2	51.6	49.9	46.9	55.6
Ind. IPCA*	93.1	69.7	59.3	50.9	48.3	43.4	38.9	44.1
ICA	91.3	72.5	56.9	51.4	50.1	45.5	43.0	48.9
ILDA	96.3	70.6	66.7	59.4	55.6	52.4	49.3	57.6
PCA/SVM	97.5	72.5	71.3	64.0	61.7	58.1	55.5	62.4
ILDA/SVM	96.3	72.5	63.6	62.3	64.3	60.3	57.4	64.5
SVM	96.9	70.6	70.9	66.6	64.3	60.2	58.2	66.0

Table 3: Training time across approaches and datasets (ms/img)

	ATT	ACR	BCB	JBK	EGO	RGA	LLP	SSB
PCA	1.3	1.1	3.6	53.1	54.4	35.7	26.6	25.3
IPCA	23.7	16.7	22.4	24.4	22.5	24.4	17.5	24.4
Ind. IPCA*	1.7	3.8	3.5	3.3	3.5	3.5	3.6	3.7
ICA	504	735	261	132	439	231	160	146
ILDA	2.8	1.4	4.5	66.7	75.8	51.8	37.9	35.7
PCA/SVM	1.6	1.2	3.9	53.7	55.8	39.0	30.3	29.0
ILDA/SVM	1.9	1.3	4.6	67.0	77.3	51.6	40.3	37.4
SVM	5.9	4.8	8.4	29.2	53.1	77.2	109	108

Table 4: Testing time across approaches and datasets (ms/img)

	ATT	ACR	BCB	JBK	EGO	RGA	LLP	SSB
PCA	0.1	0.1	0.1	0.2	0.6	1.1	1.5	1.6
IPCA	0.3	0.1	0.1	0.2	0.5	1.1	1.5	1.6
Ind. IPCA*	3.2	0.6	2.0	4.6	12.3	14.2	17	17.3
ICA	0.2	0.1	0.1	0.2	0.6	1.0	1.4	1.5
ILDA	0.0	0.0	0.1	0.2	0.6	1.1	1.5	1.5
PCA/SVM	0.2	0.1	0.1	1.4	4.3	8.2	12.5	11.4
ILDA/SVM	0.0	0.0	0.1	0.8	4.3	8.0	12.9	12.1
SVM	3.3	1.9	6.3	32.1	64.5	105	146	156

Table 5: Training memory across approaches and datasets (MB)

	ATT	ACR	BCB	JBK	EGO	RGA	LLP	SSB
PCA	7.7	5.0	25.5	241	457	602	772	804
IPCA	8.3	6.0	16.0	68.4	133	206	291	308
Ind. IPCA*	0.4	0.5	1.4	1.1	2.8	19.2	24.8	24.9
ICA	21.6	16.0	39.6	281	593	902	1158	1206
ILDA	8.7	5.2	27.2	274	564	687	832	859
PCA/SVM	8.0	5.1	25.3	239	450	580	720	746
ILDA/SVM	7.8	5.0	27.4	273	558	668	787	813
SVM	8.6	5.6	21.8	129	261	409	581	614

Table 6: Test memory across approaches and datasets (MB)

	ATT	ACR	BCB	JBK	EGO	RGA	LLP	SSB
PCA	2.7	2.6	2.8	4.4	7.2	10.8	15.0	15.8
IPCA	2.2	2.3	2.5	4.0	6.4	10.1	13.9	14.7
Ind. IPCA*	5.7	1.9	3.7	9.5	21.5	31.8	38.3	32.3
ICA	2.8	2.7	2.9	5.0	10.4	19.8	30.0	31.9
ILDA	1.0	0.2	0.5	1.7	3.4	5.3	7.1	7.5
PCA/SVM	3.1	2.8	3.9	12.3	35.5	69.8	106	86.5
ILDA/SVM	1.2	0.3	0.7	8.0	32.9	65.4	99.4	79.5
SVM	11.4	7.0	24.1	118	250	405	580	575

Table 7: Model size across approaches and datasets (MB)

	ATT	ACR	BCB	JBK	EGO	RGA	LLP	SSB
PCA	2.7	2.6	2.8	4.2	6.0	8.0	10.3	10.7
IPCA	2.4	2.3	2.5	3.8	5.4	7.2	9.3	9.7
Ind. IPCA*	5.7	1.9	3.7	9.4	21.4	31.1	38.2	32.1
ICA	2.8	2.7	3.0	4.4	6.2	8.2	10.5	10.9
ILDA	1.0	0.2	0.5	1.6	2.5	2.5	2.5	2.5
PCA/SVM	3.1	2.8	3.5	8.6	18.5	32.4	48.1	43.6
ILDA/SVM	1.2	0.3	0.7	4.7	15.6	28.0	41.9	36.5
SVM	10.8	7.1	22.7	113	231	367	525	535

(>500MB). Using the incremental version of PCA [15] in batch mode reduces the training memory requirements by over half while still achieving comparable accuracy. IPCA is not fully incremental in that the mean face of the training images must be available a-priori; incrementally calculating the mean causes a large drop in accuracy.

As indicated by the asterisk (*), Individual IPCA is unique because it is the only algorithm that is completely incremental. An eigenspace is built for each person, with the mean face calculated incrementally as each image is processed. A new face can be trained on by creating or updating the eigenspace using IPCA. Furthermore, training is extremely fast. However, the accuracy is 10-20% less than SVMs and recognition is slower because of the projections into multiple eigenspaces.

ICA improves the accuracy of straight PCA by significantly increasing the computation times and memory requirements. However, when discarding the top 10 eigenfaces for each algorithm, PCA performs better, perhaps because the top PCA eigenfaces encode illumination variations. ICA's learning logistic function [4] executes a fixed number of iterations, leading to a bottleneck and the counter-intuitive result that larger datasets exhibit a lower training time per image.

LDA [13] was a top performer on the small datasets, but scaled so poorly we did not run it on the three largest datasets. Since LDA is infeasible on a large system, we instead list the batch-mode ILDA [17] results. ILDA's accuracy is slightly greater than PCA and the run times are more favorable than ICA. Additionally the model size is

smaller because of the decreased eigenspace dimensionality of LDA.

Perhaps the best compromise is the combination of PCA or ILDA and SVMs. Instead of the expensive computation of training SVMs on the full images and generating very large model files, we can use the eigenspace representation of the faces to greatly reduce the complexity while only sacrificing 2-4% accuracy. However even in this scenario, it takes roughly five minutes and a gigabyte of memory for a normal user.

5. Conclusion and Future Work

In conclusion, we have utilized a new, real-world source of images to test a variety of algorithms for holistic performance with respect to the potential application of Facebook. The results from PCA, LDA, ICA, and SVMs show that no single or hybrid method tried is ideally suited to a widespread application for use by millions of Facebook users. SVM and ILDA methods yield fair ~65% accuracy at the cost of high computation and memory requirements. Further, they must be completely retrained with each new image. Likewise, the Individual IPCA approach is ideally suited to a real-world implementation, but yields a low accuracy intolerable to most users. However if the scope was scaled back from full autonomy, Individual IPCA could aid in tagging by automatically detecting faces and suggesting most likely identities.

Future work includes the exploration of iterative SVMs and its parameter space to yield more optimal results in memory consumption. In addition, more recent approaches such as 3D face reconstruction may correct pose problems inherent to the algorithms presented.

The detection and extraction of faces is tightly constrained by Viola-Jones and other parameters, which further reduces the accuracy of recognizing faces because they are rejected by the face extraction stage. In addition, we inadvertently accept non-faces like the last photo of Person E in Figure 1, as well as miss-tagged users. Further work should add face detection methods of increased accuracy and a filtering step to remove outliers.

More resources are available on the authors' websites.

Acknowledgements

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Special thanks to our Facebook volunteers.

References

[1] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face Recognition: A Literature Survey," *ACM Computing Surveys,* vol. 35, pp. 399-458, 2003.

[2] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," in *IEEE CVPR*, 1991, pp. 586-591.

[3] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," in *IEEE TPAMI*. vol. 19, 1997, pp. 711-720.

[4] M. S. Bartlett, J. R. Movellan, and T. J. Sejnowski, "Face Recognition by Independent Component Analysis," *IEEE Transactions on Neural Networks,* vol. 13, pp. 1450-1464, 2002.

[5] G. Guo, S. Z. Li, and K. Chan, "Face Recognition By Support Vector Machines," *Image and Vision Computing,* vol. 19, pp. 631-638, 2001.

[6] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET Evaluation Methodology for Face-Recognition Algorithms," *IEEE TPAMI,* vol. 22, pp. 1090-1104, 2000.

[7] Y. Hu, D. Jiang, S. Yan, and L. Zhang, "Automatic 3D Reconstruction for Face Recognition," in *IEEE FG*, 2004, pp. 843-848.

[8] Facebook, "Facebook | Statistics," 2008. http://www.facebook.com/press/info.php?statistics.

[9] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," University of Massachusetts, Amherst 2007.

[10] J. Ruiz-del-Solar and P. Navarrete, "Eigenspace-Based Face Recognition: A Comparative Study of Different Approaches," *IEEE Trans. on Sys., Man. & Cyb. C.,* vol. 35, p. 315, 2005.

[11] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," in *IEEE CVPR*, 2001, pp. 511–518.

[12] M. Castrillón-Santana, O. Déniz-Suárez, L. Antón-Canalís, and J. Lorenzo-Navarro, "Face and Facial Feature Detection Evaluation," in *VISAPP*, 2008.

[13] D. S. Bolme, J. R. Beveridge, M. Teixeira, and B. A. Draper, "The CSU Face Identification Evaluation System: Its Purpose, Features, and Structure," in *ICVS* Graz, Austria, 2003.

[14] A. T. L. Cambridge, "The Database of Faces," http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[15] J. Weng, Y. Zhang, and W. S. Hwang, "Candid Covariance-Free Incremental Principal Component Analysis," *IEEE TPAMI,* vol. 25, pp. 1034-1040, 2003.

[16] X. Liu, T. Chen, and S. M. Thornton, "Eigenspace Updating for Non-Stationary Process and Its Application to Face Recognition," *Pattern Recognition,* vol. 36, pp. 1945-1959, 2003.

[17] T. K. Kim, S. F. Wong, B. Stenger, J. Kittler, and R. Cipolla, "Incremental Linear Discriminant Analysis Using Sufficient Spanning Set Approximations," in *IEEE CVPR*, 2007, pp. 1-8.

[18] B. Heisele, P. Ho, and T. Poggio, "Face Recognition with Support Vector Machines: Global versus Component-Based Approach," in *ICCV*. vol. 2 Vancouver, Canada, 2001, pp. 688–694.

[19] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.