ViFaI: A trained video face indexing scheme

Harsh Nayyar
hnayyar@stanford.edu

Audrey Wei
awei1001@stanford.edu

Abstract

In this work, we implement face identification of captured videos by first training a set of face images using the Principal Component Analysis in addition to the Fisherfaces Linear Discriminant. This set of face images are acquired from Facebook tagged photos, stored after performing a Haar feature-based cascade classifier, which detects and only keeps good matches. Once the training phase is performed, we execute two classification algorithms: the nearest-neighbour (NN) and one-against-one multi-class Support Vector Machine (SVM). Our results demonstrate that the SVM outperforms NN by computing the accuracy of each schemes performance. Although our classification accuracy on our test images is over 70%, we do not observe such performance on our video input. In fact, the misclassification error prevents the expected system operation. We hope to address this issue in future work.

1. Introduction

With the increasing prominence of inexpensive video recording devices (e.g., digital camcorders and video recording smartphones), the average user's video collection today is increasing rapidly. With this development, there arises a natural desire to efficiently access a subset of one's collection of videos. The solution to this problem requires an effective video indexing scheme. In particular, we must be able to easily process a video to extract such indices.

Today, there also exist large sets of labeled (tagged) face images. One important example is an individual's Facebook profile. Such a set of of tagged images of one's self, family, friends, and colleagues represents an extremely valuable potential training set.

In this work, we explore how to leverage the aforementioned training set to solve the video indexing problem.

2. Problem Statement

Use a labeled (tagged) training set of face images to extract relevant indices from a collection of videos, and use these indices to answer boolean queries of the form: "videos with 'Person 1' OP_1 'Person 2' OP_2 ... OP_{N-1} 'Person N' ", where 'Person N' corresponds to a training label (tag) and OP_N is a boolean operand such as AND, OR, NOT, XOR, and so on.

3. Relevant Works

There are multiple approaches that exist in the literature for video indexing. In [5], the authors use multimodal analysis, exploring visual, auditory, and textual modality. In relevance to our research, they perform people detection in video documents by using the neural network-based system [6]. This algorithm detects 90% of all upright and frontal faces and only mistakes a minimal number of non-face areas for faces.

This source also detects people by identifying whole human bodies and applying detectors for heads, legs, and arms. Each individual detector is based on the Haar wavelet transform and utilizes a second example-based classifier. For identification, the authors compute eigenfaces and Fisherfaces, covered in section 4.2, which achieve better results when both variations in lighting and expression are present. However, one downfall is that face recognition only works well in constrained environments, such that frontal faces are captured closely to the camera.

Eigenfaces and Fisherfaces are strong algorithmic approaches for our application. With Facebook tagged images as our training set, we experience various lighting settings and facial expressions. We also exploit the Haar wavelets [4] for face detection with the calculation speed being its great advantage.

In [7], the authors use Independent Component Analysis (ICA) as a generalized Principal Component Analysis (PCA) procedure. The ICA linearly transforms data into components that are maximally independent from each other. Following this, the Discrete Wavelet Transform (DWT), due to its excellent energy compaction for highly correlated data, is used to rid high-frequency components, since high-frequency information is unneeded information for detection. Lastly, the authors perform a particular Support Vector Machine (SVM) classifier, known as SVM light, which is a fast optimization algorithm for pattern recognition. Using the DWT and SVM classifier, the authors

are able to detect with 100% accuracy a subject in a video database composed of 40 videos, 10 for each 4 subjects.

For our implementation, we executed a non-linear SVM as one form of classifiers. We chose not to deploy the neural network-based system presented in [5], since SVM is found to be less computationally expensive and performs comparably well. Additionally, we also executed the nearest-neighbours, because of its simplicity and adequate performance. These algorithms will be presented in detail in section 4.

4. Background

4.1. Face Detection

4.1.1 Haar Feature-Based Cascade Classifier

A Haar classifier is trained with hundreds of sample views of a certain object, e.g. a face, called positive examples, and views of arbitrary objects, called negative examples. It is then applied to a region of interest (ROI) of a particular input image. The classifier will output a "1" if the ROI is likely to contain the desired object, e.g. face, or "0" otherwise. This procedure is repeated, scanning every location and varying its window size, until the entire image is searched. The features used in these classifiers are as depicted in Figure 1.

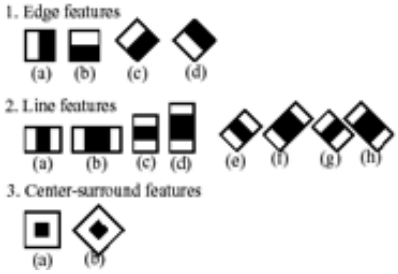

Figure 1. Haar-like features

4.2. Training

Two proposed methods of representing data for applications in the computer vision and machine learning world are the Principal Component Analysis (PCA) and the Linear Discriminant Analysis (LDA).

4.2.1 Principal Component Analysis

When applied to face images, the PCA computes a set of eigenfaces that characterizes most of the data variance; it consists of eigenvectors that correspond to the largest eigenvalues of the training datas covariance matrix. This is equivalent to a least squares solution and eliminates unnecessary existing correlations among original features, and thus significantly reduces the dimensions of the feature space. [1]

To formally present this algorithm, consider a set of N sample images, $\{x_1, x_2, ..., x_N\}$, which belongs to one of c classes, $\{X_1, X_2, ..., X_c\}$. The projection matrix W_{opt}, with a total scatter matrix S_T, is computed such that:

$$W_{opt} = \arg\max_W |W^T S_T W|$$

$$= [w_1 w_2 ... w_m]$$

$$S_T = \sum_{k=1}^{N} (x_k - \mu)(x_k - \mu)^T$$

where $\mu \epsilon \mathcal{R}^n$ represents the mean image of all samples, and W ϵ $\mathcal{R}^{nxm}$ represents the transposed linear transformation matrix with orthonormal columns. The columns of W_{opt} are the n-dimensional eigenvectors of S_T corresponding to the m largest eigenvalues. After mapping the n-dimensional image space into a reduced m-dimensional feature space, the new linearly transformed feature vectors $y_k \epsilon \mathcal{R}^m$, are computed as follows:

$$y_k = W^T x_k, \text{ with } k = 1, 2, ..., N$$

For classification, the main objective is to maximize the between-class scatter in order to categorize specific faces to its correct match. The PCA is able to obtain this goal. However, variations between images of the same face due to illumination and viewing directions are generally much higher than changes in face identity. Hence, the projection matrix W_{opt}, will contain principal components that retain the undesired variation due to lighting. Moreover, this method maximizes the within-class scatter, which increases the probability of having different classes smeared together. As a result, PCA projections do not perform well in classification problems. [1]

4.2.2 Linear Discriminant Analysis

Similarly to PCA, LDA projections map sample vectors of different classes as far apart from each other as possible in the feature space. Unlike PCA, it minimizes within-class distances, which places LDA as a much stronger classification algorithm than PCA.

Let the between-class and within-class scatter matrices be defined as:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^{c} \sum_{x_k \epsilon X_i} N_i(x_k - \mu_i)(x_k - \mu_i)^T$$

where μ_i is the mean image of class X_i, and N_i is the number of samples in class X_i. The optimal projection W_{opt} is computed such that the ratio of the determinant of the between-class scatter matrix to the determinant of the within-class scatter matrix is maximized.

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

$$= [w_1 w_2 ... w_m]$$

The columns of W_{opt} are the generalized eigenvectors of S_B and S_W corresponding to the m largest eigenvalues, λ_i, such that:

$$S_B w_i = \lambda_i S_W w_i, \text{ with } i = 1, 2, ..., m$$

Since the number of images N, is usually much smaller than the n number of pixels in each image, this algorithm faces the issue of having S_w as a singular matrix. This potentially leads to a projection matrix that causes the within-class scatter to become zero, which is undesirable.

4.2.3 Fisherfaces Linear Discriminant

The proposed solution in [1], called Fisherfaces Linear Discriminant (FLD), is to utilize PCA to reduce the dimensions of S_w to $N - c$, making it non-singular, before applying LDA, which reduces the dimensions further down to $c - 1$. The equations of these steps are as follows:

$$W_{opt} = W_{pca} W_{fld}$$

$$W_{pca} = \arg\max_W |W^T S_T W|$$

$$W_{fld} = \arg\max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

The new linearly transformed feature images $y_k \epsilon \mathcal{R}^m$, are computed as follows:

$$y_k = W_{opt}^T x'_k, \text{ with } k = 1, 2, ..., N$$

where $x'_k = x_k - \mu$ and μ is the global mean of all training images.

4.3. Classification

4.3.1 Nearest-Neighbour

Our first approach of performing classification on a single test image I, is the nearest-neighbour method. This technique is one of the simplest algorithms for predicting the class of a test sample [9]. The Euclidean distance of the newly transformed images y_k, and I', such that $I' = I - \mu$, is computed as follows:

$$d(y, I') = ||y - I'|| = \sqrt{\sum_{k=1}^{N} (y_k - I')^2}$$

After $d(y, I)$ is computed for each class c, the class which gives the minimum error $d(y, I)$ is considered the match for the test image if $d(y, I)$ falls within a threshold value of some τ.

A major disadvantage of the NN is the computational cost of $O(Nm)$ time for a single test example [9]. Thus, another classifier, known as the Support Vector Machine, is introduced in the next section.

4.3.2 Support Vector Machine

The standard SVM takes a set of data inputs and for each given input predicts which of two possible classes it falls under. Firstly, the SVM takes in a set of training samples, each indicating which of the two categories it belongs to, and builds a model such that samples of different categories are separated as far as possible. Test inputs are then mapped into the same space and predicted to belong in a category based on which region they fall in.

Assuming that each data point is a p-dimensional vector, our goal is to maximally separate the data points from different classes with a (p-1)-dimensional hyperplane. This represents a linear classifier. The training data D is defined below:

$$D = \{(x_i, y_i) | x_i \epsilon \mathcal{R}^p, y_i \epsilon \{-1, 1\}\}_{i=1}^n$$

where y_i indicates the class to which the point x_i belongs.

In our algorithm, we implement a one-against-one SVM with a non-linear classifier. The one-against-one method is constructed using training samples belonging to two classes only, e.g. classes p and q.

$$y_i = +1, \text{ if } c_i = p$$

$$y_i = -1, \text{ if } c_i = q$$

The classifier used is computed by applying kernels to maximize the margin between hyperplanes. We utilize a common kernel, known as the Gaussian radial basis function (RBF), $k(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2)$, for $\gamma = \frac{1}{2\sigma^2}$ [8].

We can predict which class the input falls under by computing the sign of $f(x)$, the value of the discriminant function, provided below:

$$f_{pq}(x) = \sum_{i=1}^{N_s} \alpha_i y_i K(s_i, x) + b$$

where N_s = # of centers, s_i = centers, α_i = weights, and b = threshold.

These parameters are automatically produced by the trained SVM model. Thus, the only unknown parameter is σ, the RBF width. Based on [8], setting $\sigma^2 = 200$ is an optimal choice, provided that it is less prone to errors and generally produces good kernels.

If $f_{pq}(x) > 0$, class p wins a vote.

otherwise, class q wins a vote

The class with the maximum number of votes is assigned to the test image. [10]

5. System Overview

In this section, we outline our proposed scheme to address the problem we postulate in section 2 and will provide further details about the system implementation in section 6.

At a high level, we subdivide the problem into two key phases: the first "off-line" phase executed once, and the second "on-line" phase instantiated upon each query.

For the purposes of this work, we define an *index* as follows: <video id, tag, frame #>.

A visual layout of the system implementation is presented in Figure 2.

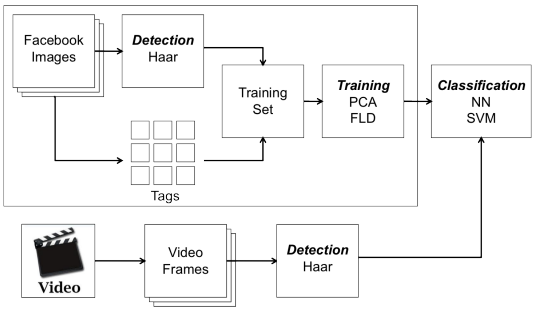

Figure 2. System Overview

5.1. The training phase

We first outline Phase 1 (the training or "off-line" phase):

1. Detect faces from acquired Facebook images using the Haar feature-based cascade classifier, presented in section 4.1.1. Store correctly detected faces in a labeled training set.

2. Use the labeled training set plus an additional set of 'other' faces to compute the Fisher Linear Discriminant (FLD) [1].

3. Project the training data onto the space defined by the eigenvectors returned by the FLD, and train a classifier (first nearest-neighbour, then SVM) using the training features.

4. Iterate through each frame of each video, detecting faces [2], classifying detected results, and adding an index if the detected face corresponds to one of the labeled classes from the previous step.

5.2. The query phase

Now, we outline Phase 2 (the query or "on-line" phase):

1. Key the indices on their video id.

2. For each video, evaluate the boolean query for the set of corresponding indices.

3. Keep videos for which the boolean query evaluates true, and discard those for which it evaluates false.

6. Implementation Details

We are implementing the project in C++, leveraging the OpenCV v2.2 framework [4]. In this section, we will highlight some of the critical implementation details of our proposed system.

6.1. Detection and Training Set Acquisition

In order to obtain the training faces, we must extract faces from tagged Facebook data. This requires parsing through Facebook's Graph API. Essentially, for our purposes we have images, each with a set of tags. We first retrieve these images (using the wget utility), and then proceed to extract the tagged faces.

At this stage, we perform some outlier removal. Specifically, we run the OpenCV Viola-Jones based face detector, specifically the Haar feature-based cascade classifier, in order to detect faces. If the detected faces correspond to tags, we accept the face. In our system, we reject regions with tags but no detected faces. We use a Haar classifier that focuses on frontal views of faces.

During the acquisition phase, we also resize all training samples to a standard size of 40px by 40px, and store the grayscale representation of the samples.

6.2. Computing the Fisher Linear Discriminant

For this stage of the training pipeline, we first perform PCA to reduce the dimensionality of our data to $N - c$ where N is the number of images in our training set, and c is the number of classes. We then perform the Fisher Linear Discriminant, minimizing the within-class scatter and maximizing the between-class scatter.

We utilize the OpenCV implementation of singular value decomposition (SVD) to perform the above tasks.

6.3. Classification

Having determined the optimal projection from the previous stage, we are left with a set of features for each training sample. We investigate two approaches for classification: nearest-neighbour (NN) and support vector machine (SVM).

First, we implement a nearest-neighbour approach. Here, we simply store the training features. For a query sample, we compute the euclidean distance to the nearest neighbour, and assign the closest class within a certain threshold, τ. This threshold will be learned through training and cross-validation.

We expect that the two nearest-neighbour approaches are unlikely to provide robust performances when testing on general test data. For this reason, we also implement a non-linear SVM-based classification scheme.

We will begin using the one-against-one scheme, and train a set of two-class SVM classifiers able to discriminate between a particular class and the rest of the training set. Based on the set of results, we infer the winning class.

6.4. Indexing

In this stage, for each supplied test video, we iterate through each frame and perform face detection to extract faces. We resize the faces to 40px by 40px, project the sample to obtain a feature vector, and this then serves as the input to our classifiers.

If our classification output determines that the query sample is one of our learned classes, we have an index and record it in the form of the index we presented above.

6.5. Query evaluation

Based on the previous progression of tasks, our problem setup makes the query evaluation relatively trivial. For a given query, we simply evaluate the boolean expression for each video. This is based on the recorded indices for that video, which are obtained according to the previous section.

7. Results and Evaluation

In this section, we evaluate the key components of our proposed system. Specifically, we evaluate our detection performance, our classification performance, and our final aggregate system performance.

7.1. Face Detection

Our first requirement in this work is a labeled (tagged) set of face images. We have obtained this using the data of five Facebook users. After parsing the Facebook Open-Graph API results, we present the following summary statistics, which are based on a user's 25 most recent tagged photos:

- Average number of tags per photo: *5.384*
- Average number of detections per photo: *1.752*

Immediately, we observe that we only actually detect approximately 33% of the potential detections.

There is a three-fold explanation for this observation. First, our detector performs poorly on large group photos. Such photos have many tags and very few detections. For reference, one such example is depicted below in Figure 3. As we can see, of the approximately 30 (tagged) individuals, only a small subset of individuals (e.g. 5) are actually detected.

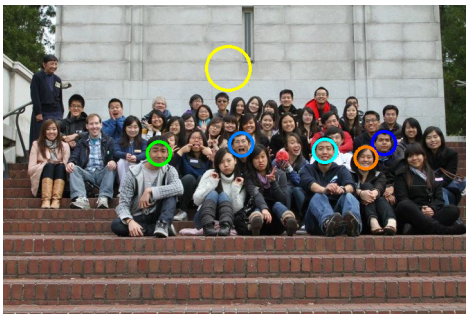

Figure 3. Poor performance for large group photos

It is important to note that this phenomena is only observed for large groups, as in this scenario. For smaller, more typical sized groups (approximately 5-8 individuals), the performance is far better. For reference, we provide a representative sample in Figure 4.

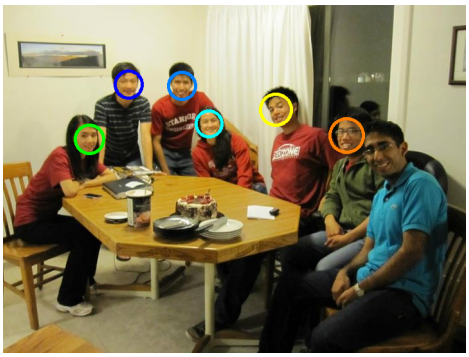

Figure 4. Improved performance for average sized group photos

The second factor contributing to the relatively low face detection rate can potentially be attributed to false tags. Our implementation only accepts a detection if and only if there is a corresponding tag at a detection hit. We have observed that on Facebook; it is often the case that users will tag non-faces as individuals. Obviously, such a scenario will yield in a missed detection. One application of such a scenario would be to ignore the clear false positives in the large group example (e.g. the wall).

The third and final factor that we believe contributes to these results is the presence of non-frontal views of faces. Although this scenario is prevalent in natural scenes, it is a known problem that side-face detection does not yet have a robust solution.

Finally, we present another important metric when evaluating our face detection stage: the false positive rate. In our system, across our five test subjects, *we do not observe a single false positive*. Specifically, what this result means is that we get a final result that is either of the following:

- not a face

- or, a mis-labeled face

Both of these results validate our implementation decision to only accept detections that correspond to a tagged location. It is important our aforementioned successive classification and indexing stages behaviours hold true.

7.2. Classification Performance

In this section, we evaluate our two candidate classification schemes: the simple NN scheme, as well as our one-against-one multi-class SVM classification scheme. We investigate the classification performance.

7.2.1 Training Set Description

In this evaluation, our training set is comprised of 5 individuals in their mid-twenties, two females and three males. We have anonymized the identities for this report as Subjects A, B, C, D, and E. Moreover, for reference, Subjects B and E are female, while A, C, and D are male.

We have selected such a group because we feel it accurately represents a typical use case. We expect our system to be used to index videos recorded on a circle of friends that are likely similar in age and roughly balanced in gender.

In this section, we reference a '5-sample' training set and frequently compare this to a '10-sample' training set. What this represents is that in the former case, each constituent class of the corresponding classifier is trained based on five samples. In the latter case, this is ten samples. Moreover, we always measure the classification accuracy based on an *independent* set of five samples for each class. That is, the test set is independent of the training set.

7.2.2 NN Performance

In this section, we present the classification performance of our nearest-neighbour classifier. We further analyze the performance improvement as we increase the size of the training set.

7.2.3 Base case: pair-wise classification

We first examine the performance of the classifier in distinguishing any pair of individuals from our training set. This is the base case.

This table is read as the classification accuracy of the row heading with respect to the pair defined by the row and column intersection. For example, the performance of A in the (A,C) classifier is 80%, while the performance of C in the (A,C) classifier is 40%.

From the Table 1 above, we conclude that the average classification accuracy is 70% when distinguishing two individuals.

	A	B	C	D	E
A	X	100%	80%	60%	80%
B	100%	X	100%	100%	80%
C	40%	60%	X	20%	20%
D	40%	80%	60%	X	40%
E	80%	80%	80%	100%	X

Table 1. NN classification accuracy for 5 training samples

	A	B	C	D	E
A	X	100%	80%	80%	80%
B	100%	X	100%	100%	80%
C	80%	100%	X	80%	40%
D	60%	80%	60%	X	60%
E	100%	80%	100%	100%	X

Table 2. NN classification accuracy for 10 training samples

In Table 2 above, we conclude that the average classification accuracy is 83% when distinguishing two individuals.

Comparing Tables 1 and 2, it becomes obvious that using a larger number of samples increases the classification accuracy. Specifically, we were able to obtain an improvement of 13%, on average in classification accuracy when we increased the number of training samples from 5 samples to 10 samples.

7.2.4 Increasing the classes to four classes

In order to quantify the generalization performance of our nearest-neighbour classifier, we increase the number of classes from 2 to 4. We evaluated the classification performance of all possible subsets of 4 subjects from our 5-subject training set.

Figure 5 depicts the classification performance. In this figure, we show the performance of each possible subset as well as the average performance over all subsets. Moreover, for each subset, we show two bars. These bars compare the classification accuracy of the five-sample training set and the ten-sample training set.

The most important conclusion we make from Figure 5 is that on average, classification performance improves as we increase the training set size. Specifically, on average, the NN classifier achieves a classification accuracy of 73%.

7.2.5 The entire training set

Finally, we investigated the generalized performance on the complete training set. Again, we have comparative results for the 5-sample training set and the 10-sample training set.

Again, we observe that the larger training set improves the classification accuracy. Moreover, for the 10-sample

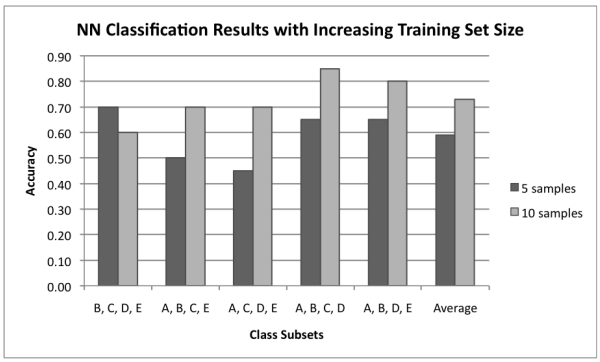

Figure 5. Performance generally increases as the size of the training set is increased.

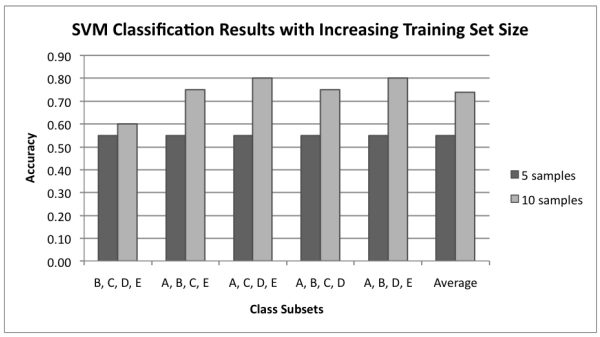

Figure 6. Performance generally increases as the size of the training set is increased.

	5-sample	*10-sample*
A	60%	40%
B	100%	80%
C	40%	60%
D	60%	100%
E	80%	80%
AVG	68%	72%

Table 3. NN classification accuracy for all 5 classes

	5-sample	*10-sample*
A	60%	60%
B	100%	80%
C	20%	80%
D	80%	60%
E	80%	100%
AVG	68%	76%

Table 4. One vs. one SVM classifier accuracy for all 5 classes

training set, we observe that the performance drops negligibly with respect to the average 4-class results we presented in the previous section.

7.2.6 One vs. One SVM Performance

Next, we analyze the performance of our one-against-one multi-class SVM classifier. One parameter we have is the number of votes we require to admit a class. We set it to 50%. This means the winning class must have greater than 50% of the votes, otherwise the classifier outputs "none".

As with the NN performance evaluation, we again evaluate the SVM performance based on the 5-sample training set as well as the 10 sample training set.

Again, as with the nearest-neighbour classification scheme, we see that as we increase the size of the training set, we observe an increase in classification accuracy as well. Moreover, in contrast to the nearest-neighbour scheme, this figure suggests that the SVM approach benefits more from a larger training set. On average, with the 10-sample training set, the SVM classifier achieves a classification accuracy of 74%.

Finally, we investigate the generalized performance on the complete training set. Again, we have comparative results for the 5-sample training set and the 10-sample training set.

We observe that increasing the training set size improves the classification performance. Moreover, we see that the 10-sample SVM outperforms the 10-sample NN classifier presented in the previous section (76% vs. 72%).

7.3. Overall System Performance

At this point, our overall system performance is unsatisfactory. We have a set of 8 training videos consisting of varied lighting conditions as well as different subsets of our five training set subjects. Although we have an integrated system, we get an overwhelming number of misclassifications that is not consistent with our classification performance results as outlined in the previous section.

Due to the extremely poor classification results on the video, we were unable to evaluate our final indexing performance. We address these shortcomings in future work.

8. Conclusion

In this work, we conduct an evaluation of two main schemes for classifying the detected faces acquired from Facebook videos: nearest-neighbours and one-against-one SVM classifiers. Our analysis shows that the one-against-one SVM classifier outperforms NN.

Although we had some poor performances in cases such as images of large groups or non-frontal face images, we were still able to capture and detect a majority of the faces accurately with an average of over 70% for both NN and SVM. Not only does this confirm that both classifiers work

reasonably well, it also shows that PCA and FLD characterize the evaluated face images well. Moreover, an important outcome of our work is that this approach works well for natural images acquired through Facebook.

Some additional reasons for the poor performances is that our training set needs to be more diverse. We constructed a training set using 5 Facebook users' photos and only kept faces that were accurately detected from the Haar feature-based cascade classifier.

Additionally, we found the following statistics for face detection on Facebook images. Based on a user's 25 most recently tagged photos, the average number of tags per photo is 5.384 and the average number of detections per photo is 1.752.

All in all, this work represents a positive start towards our ambitious goal of delivering a video indexing scheme trained on tagged Facebook images. We have resolved the entire acquisition pipeline, and need to work further to refine the classifiers and tune them for the wide variation that is a defining characteristic of our problem statement.

9. Future Work

When implementing the Haar feature-based cascade classifier, we could incorporate multiple classifiers such as one that focuses on side-profile faces. This could help us become more robust against the variations that are present in video that are simply not present in images.

Our current results suggest that we ought to also investigate acquiring a larger training set. Specifically, we need to ensure that all subjects have a variety of training samples from varying illuminations and scenes.

Finally, although our classification performance is reasonable, we might investigate the one-versus-all multi-class SVM. This could help us generalize better as we will be able to categorically rule out queries as belonging to a particular class, instead of relying on votes. Now it is possible that we get a positive result for multiple classes and in that scenario we could resort to breaking ties based on the actual margin of the classifier.

References

[1] P.N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. "Eigenfaces vs. Fisherfaces: Recognition using class specic linear projection," *IEEE Trans. Patt. Anal. Mach. Intell* 19, 711 720, 1997

[2] P. Viola and M. Jones, "Robust Real Time Object Detection," *IEEE ICCV Workshop Statistical and Computational Theories of Vision*, July 2001.

[3] Face Recognition Homepage. http://www.face-rec.org/databases/

[4] OpenCV. http://opencv.willowgarage.com/

[5] C. Snoek, M. Worring. "Multimodal Video Indexing: A Review of the State-of-the-art," *Intelligent Sensory Information Systems. Univ. of Amsterdam*.

[6] H. A. Rowley, S. Baluja, T. Kanade. "Neural Network-Based Face Detection," *IEEE Pattern Analysis and Machine Intelligence*, January 1998.

[7] M. del Pozo-Banos, C. M. Travieso, J.B. Alonso, M. A. Ferrer. "Face Identification based on TV videos," *IEEE ICCV Security Technology*, October 2009.

[8] Y. Chen, X. Zhou, T. Huang. "One-class SVM for Learning in Image Retrieval," *IEEE ICCV Image Processing*, 2001.

[9] C. Elkan. "Nearest Neighbor Classification," *University of California, San Diego*, January 2011. http://cseweb.ucsd.edu/ elkan/151/nearestn.pdf

[10] N. Seo. "A Comparison of Multi-class Support Vector Machine Methods for Face Recognition," *University of Maryland*, December 2007.