Implemenation and Comparison of PCA, ICA and LDA algorithms using the SVM Classifier for Face Recognition

Robert Barbulescu, MSc Intelligent Systems & Robotics, De Montfort University, Leicester

***Abstract*— Face Recognition has seen a significant increase in popularity during the past decade and is one of the most successful applications of image analysis. Due to this increase in adoption, different statistical methods have been proposed in recent years and various research is being undertaken in the field, but inconsistent implementations and datasets have caused contradictory results when comparing them. Face Recognition compromises two main steps, feature representation and face classification, the goal of this paper is to develop an implementation of a chaining PCA-SVM system, tune it, and adjust it for usage with similar appearance-based algorithms in order provide a comparative study between the most popular techniques. The main objective of this paper will be the development and testing of the PCA-SVM system with subsequent implementations of ICA-SVM and LDA-SVM, and contrast of results. Labeled Faces in the Wild (LFW) dataset will be used for all testing in order to preserve consistency.**

Keywords—Biometrics, Cyber Security, Face Recognition, Computational Intelligence, PCA, SVM, ICA, LDA, Eigenfaces.

1. INTRODUCTION

Face recognition is today one of the most popular and successful applications of image analysis and can have the following overview, as presented by Delac et al. (2005): *when provided with a still or video image, a system should be able to identify or verify one or more persons using a stored database of faces.* There has been significant research and various techniques developed for face recognition which can be classified into (1) *appearance-based*, methods that are applied to either the entire face or to specific parts in a face image and use holistic texture features, and (2) *feature-based*, which focus on the relationship between geometrical facial features (Delac et al., 2005, online).

This paper aims at developing and implementing a face recognition system based on the Principal Component Analysis (PCA), a dimensionality reduction technique that can determine the most representative projection vectors in order to allow for projected samples to maintain most information about the original samples, and Support Vector Machine (SVM), a supervised learning algorithm that finds the optimal linear decision surface using a binary classification method, this is based on the concept of structural risk minimization (Phillips, 1998, online). This system will be developed and tested using Labeled Faces in the Wild (LFW) dataset as it is a comprehensive collection of unconstrained face images which allows for an evaluation with a large variation in pose, expression, race, background, gender, age, etc. (Huang et al., 2008, p.1). The LFW dataset also allows for consistency with other studies in the field.
The rest of this paper will be structured as follow: Chapter 2 provides an overview of the algorithms and techniques used in this research, Chapter 3 presents our implementation approach and tuning methodology, Chapter 4 reports our experimental results and our interpretation, Chapter 5 provides contrast of the results achieved using different techniques with our implementation, and Chapter 6 concludes the paper. It should be noted that for the purpose of this work we are implementing the algorithms described throughout the paper in Python.

2. ALGORITHMS AND TECHNIQUES

While most of the algorithms presented in this chapter are well known and researched, we will provide an overview in order to allow for completeness of the research, as Delac et al. (2005) describes, by algorithms we refer to the subspace projection method. Several of the algorithms we will be using are *subspace analysis methods*, in the context of face recognition subspace analysis aims at determining the basis vector which can optimally cluster projected data in accordance to their class labels, we can look at subspace as a subset of a larger space that has the same properties as he larger space (Ashok Rao and Noushath, 2010, p.2). Because face images data are often high dimensional which can, in the context of matching based recognition, create a computational intensive load that will cause processing difficulties, we need to search for suitable subspace that inherits most or the entire properties of the training space (Ashok Rao and Noushath, 2010, p.2). This is explained by Delac et al. (2005) as follow: by taking a two-dimensional image by the symbol $\boldsymbol{\Gamma}$ that has m rows and n columns and concatenate its rows and columns, we can view it as a vector in N dimensional image space ($N = m \times n$). Due to the way it is derived, the space is substantial and during recognition the algorithms will derive lower dimensional spaces (Delac et al., 2005, online). This is because using techniques such as PCA can effectively represent a face image as a feature vector of low dimension and features in such a subspace will have a more salient and richer information for recognition than the raw image (Ashok Rao and Noushath, 2010, p.2).
Generally, the main steps involved in a subspace-based algorithm in the context of face recognition is presented by Ashok Rao and Noushath (2010) as follow:

Subspace Based Algorithm for Face Recognition

Input:

- A set of N training samples belonging to C classes.
- k, number of projection vectors to be used
- A test image T.

Output:

- Feature Extraction of training samples: (Training Phase)
- Class label/identity of the test sample T.

Steps:

(1) Acquire the training samples A_i $(i = 1, \ldots, N)$.
(2) Compute the basis vectors W_k using any subspace algorithms (for eg: PCA, LDA, LPP, ICA, etc).
(3) Project the training samples A_i $(i = 1, \ldots, N)$ on to W to extract features: $Feature\_Training_i = A_i * W$ $(i = 1, \ldots, N)$
(4) Project the test sample T onto W: $Feature\_Testing = T * W$.
(5) Compute the identity by finding the nearest match between $Feature\_Testing$ with $Feature\_Training_i$ $(i = 1, \ldots, N)$.

*1) **Principal Compenent Analysis (PCA)***

Principal Component Analysis (PCA) is one most popular method used to tackle the high dimensionality problem of a face recognition system. PCA is a 'projection technique' that discovers a set of projection vectors which retains most properties of the original data (Slavkovic and Jevtic, 2012, p.121). A highly popular and effective PCA approach, which we will be using in our work, is eigenface approach where faces are converted into a subset of the main features *(eigenfaces)* and are used as the initial training set for learning. In this context recognition is performed, as presented by Turk and Pentland (1991), by having any new addition (image) projected into the eigenface subspace, after which the position in eigenface space is compared to the position of known individuals.

In our experiments we have implemented the Eigenface/PCA algorithm as presented in Turk and Pentland (1991), and Slavkovic and Jevtic (2012). Step one is creating the training set, a two-dimensional image can be projected as a one-dimensional vector by having its rows concatenated as shown in below:

$$\boldsymbol{I} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}_{mxn} \xrightarrow{CONCATENATION} \begin{bmatrix} x_{11} \\ \vdots \\ x_{1n} \\ \vdots \\ x_{2n} \\ \vdots \\ x_{mn} \end{bmatrix}_{1xN} = \boldsymbol{x}.$$

Given a random vector $\{x_1, x_2, x_3, \ldots x_n\}$ with x_i belongs to R^d, we compute: (1) the mean, (2) the covariance matrix, (3) the eigenvalues and eigenfaces of the covariance matrix.

- *(1) the mean is:*

$$\mu = \frac{1}{n}\sum_{i=0}^{n} x_i$$

- *(2) the covariance matrix is:*

$$S = \frac{1}{n}\sum_{i=0}^{n}(x_i - \mu)(x_i - \mu)^T$$

- *(3) the eigenvalues λ_i and eigenvectors v_i for S:*

$$Sv_i = \lambda v_i, \text{ where } i = 1, 2, \ldots n$$

The eigenvectors are being ordered in descendent order of their eigenvalues, and the eigenvectors are the principal components associated to the largest eigenvalues. The final step is recognition, the image of the person (principal components - *k*) we want to find in the training set is converted to a vector *x*, reduced by the mean value, and projected with a matrix of the eigenfaces:

$$i = W^T(x - \mu), \text{ where } W = (v_1, v_2, \ldots v_k)$$

Reconstruction can be performed with the following formula:

$$x = Wy + \mu$$

A simple overview of how PCA works in face recognition is:

- *All training samples are projected into PCA subspace,*
- *Test image is projected into PCA subspace,*
- *Closest neighbor between the projected training images and the projected test image is discovered.*

Note that classification is performed by calculating the distance between vectors, for the purpose of our work we are using the default method, which is Euclidean distance, but for the sake of completeness we are providing the following approaches:

Given A, B two vectors of length D, we can determine the distance (Slavkovic and Jevtic, 2012, p.124):

- *Manhattan distance:*

$$d(\boldsymbol{A},\boldsymbol{B}) = \sum_{i=1}^{D}|a_i - b_i|;$$

- *Euclidean distance:*

$$d(\boldsymbol{A},\boldsymbol{B}) = \sqrt{\sum_{i=1}^{D}(a_i - b_i)^2} = \|\boldsymbol{A} - \boldsymbol{B}\|.$$

*2) **Independent Component Analysis (ICA)***

Independent Component Analysis (ICA) is often viewed as a generalization of PCA, what PCA does is decorrelates the input data with second-ordered statistics which creates compressed data with minimum mean-squared reprojection error, in contrast ICA minimizes both second and higher order dependencies in the input data and tries to discover the basis where the data is *statistically independent* (Draper et al., 2003, p.8).

According to Draper et al. (2003) this is related to the Blind Source Separation (BSS) problem presented by (Stewart Bartlett et al. (1998), in which the aim is to decompose an observed signal into a "linear combination of unknown independent signals" (Draper et al., 2003, p.8). Considering the following:

- *s – vector of unknown source signals,*
- *x – observed mixtures vector,*
- *A – unknown mixing matrix,*

Then the mixing model is:

$$x = As$$

If we assume the source signals are independent and *A* is invertible, ICA will attempt to discover either the *mixing matrix A, or separating matrix W* so that:

$$u = Wx = Was$$

is an estimation of the independent source signals (Draper et al., 2003, p.8). This is shown in Figure 1.

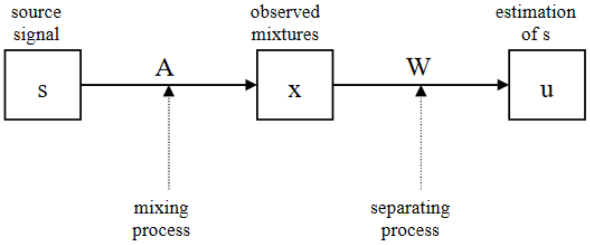

Figure 1 - BSS model (Draper et al., 2003, fig.1).

Fundamentally, it is important to know that there are two different approaches for applying ICA for face recognition, Bartlett et al. (2002) provides the two architectures: *Architecture I - Statistically Independent Basis Images*, and *Architecture II - Factorial Code Representation*, for our implementation we are focusing on the first method, as well as we will be using PCA to reduce dimensionality before performing ICA.

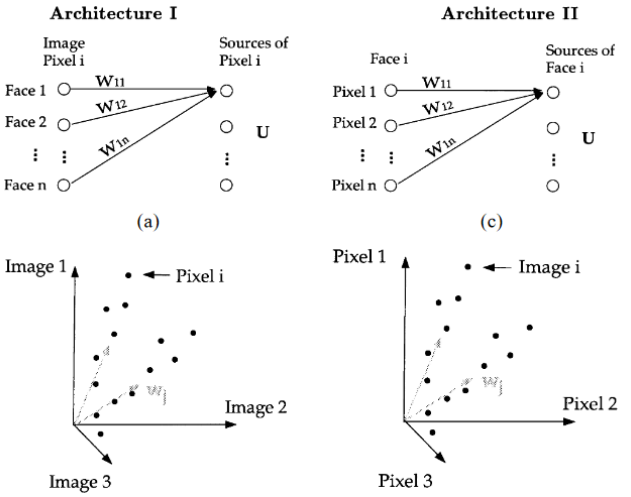

Figure 2 - ICA Architectures (Bartlett et al., 2002, fig.3).

Draper et al. (2003) explains the methodology behind architecture 1:

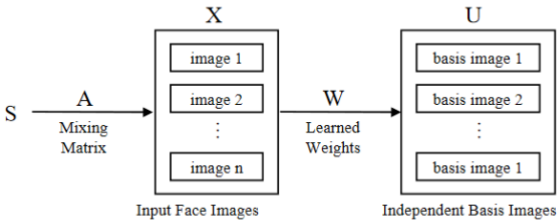

Figure 3 – Architecture 1 (Draper et al., 2003, fig.2).

-in **(X)** the input faces are a linear mixture of statistically independent basis images **(S)** combined by unknown matrix **(A),**
-in **(U)** the rows are a set of independent basis images which are recovered by the weight matrix **(W)**, ICA learns the weight matrix,
-some observation about this architecture include: the face images are variables with the source separation being performed in face space and projecting the input images to the learned weight vectors generates the independent basis images (Draper et al., 2003, p.10).
The second architecture is out of scope for our work but for easy comparison please see the figure shown below and note that while first architecture uses the images directly, the second architecture is focused on coefficients.

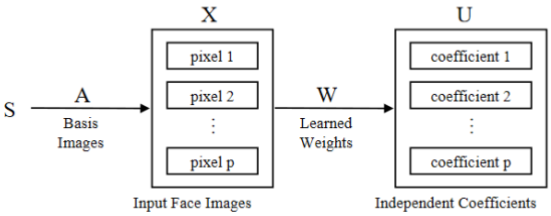

Figure 4 - Architecture 2 (Draper et al., 2003, fig.4).

3) *Linear Discriminant Analysis (LDA)*

Linear Discriminant Analysis, also known as Fisher's Discriminant Analysis, Fisher (1936), is a traditional approach in pattern recognition, the technique looks for a linear combination of features or linear transformation that allows for the separation of two classes or objects, this can be achieved through scatter matrix analysis (Sodhi and Lal, 2013, p.32).
According to Zhou et al. (2013), the underlying methodology behind LDA is as follow:

Give a training set:

$$T = (v_1, v_2, ..., v_m),\ T \text{ belongs to } R^{n \times m}$$

There are two classes to which each image belongs to and each face image has pixel values stored in each column, the classes are:

- *The between-class scatter matrix S_w:*

$$S_w = \sum_{i=1}^{c} \sum_{j=1}^{N_i} (V_j - \mu_i)(V_j - \mu_i)^T$$

- *The within-class scatter matrix S_b:*

$$S_b = \sum_{i=1}^{c} N_i (\mu_i - \mu_0)(\mu_i - \mu_0)^T$$

- *N_i = number of samples for class V_i*

The objective of LDA is to find a group of basis vectors that creates different class samples and have the largest between-class scatter and the smallest within-class scatter, if the between-scatter matrix is nonsingular than we can solve the generalized eigenvalue problem in order to obtain the optimal projection matrix (Zhou et al., 2013, p.5600).

- *Generalized eigenvalue problem:*

$$S_b W_i = \lambda_i S_w W_i, \quad i = 1, 2, \cdots, c-1$$

Finally, the total scatter matrix S_t is given by the formula:

$$S_t = \sum_{j=1}^{m} (V_j - \mu)(V_j - \mu)^T.$$

For more details on LDA please see Belhumeur et al., (1996).

*4) **Support Vector Machine (SVM)***

Support Vector Machine (SVM) is a learning machine method that implements the basic idea of non-linear transform so that the sample space will be linearly separable after changing its characteristics (Zhou et al., 2013, p.5600). What SVM does essentially is a separation or discrimination between two classes, an example of a two-class classification problem where various linear classifiers that could separate the data are possible but only one allows the maximization of the margin, which is "the distance between the hyperplane and nearest data point of each class", is shown in Figure 1 (Guo et al., 2001, p.632).

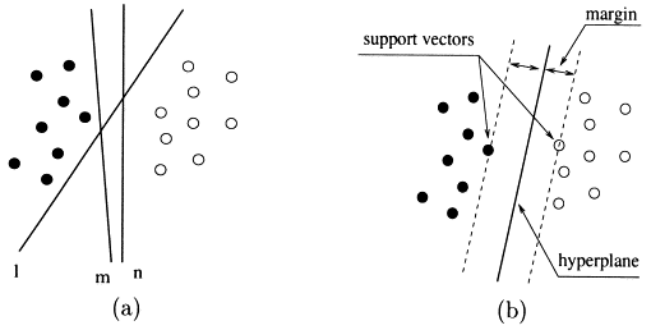

Figure 5 - Classes classification with hyperplanes (Guo et al., 2001, fig.1)

SVM also provides a method for linear non-separable data using a technique called 'kernel mapping' to map the data in input space to a high-dimensional feature space, the data then becomes linear (Zhou et al., 2013, p.5600). Some of the most popular kernel functions, which will also be used in our implementation, are presented by Zhou et al. (2013):

- *(1) Polynomial Function*

$$K(x, x_i) = [(x, x_i) + 1]^d$$

Has the following SVM classifier:

$$f(x, \alpha) = sign\left(\sum y_i \alpha_i [x_i \cdot x + 1]^d - b\right)$$

- *(2) Radial Basis Function (RBF)*

$$K_r(\left|x - x_i\right|) = \exp\left\{-\frac{(x-x_i)^2}{\sigma^2}\right\}$$

The SVM is a Gaussian RBF classifier with the optimal decision function:

$$f(x) = sign(\sum_{i=1}^{n} \alpha_i K_r(|x - x_i|) - b)$$

- *(3) Sigmoid Function*

$$K(x, x_i) = \tanh(\gamma x_i^T x_j + b)$$

There are several reasons and motivations behind the choice of classifiers, the methods presented so far have been researched extensively in the field and allows us to contrast our findings with different models, furthermore the algorithms are based on different techniques which will allow us to determine if the prediction and accuracy is constant among the algorithms.

3. Experimental Design

All three algorithms were implemented along with the SVM classifier in Python due to the highly advantageous **Scikit-learn** module, which is now considered one of the most popular python libraries for machine learning (scikit-learn, online). All the techniques and algorithms presented in our paper are available in the **scikit-learn** platform and allows for easy access and deployment, furthermore, the dataset can be imported through the library and provides access to pre-processing functionality such as ***data normalization, train and test images splitting, accuracy metrics, and confusion matrix, etc.*** this will enable us to create an initial system using the PCA and SVM classifiers, perform fine tuning to obtain best recognition results, and subsequently adjusted it to implement LD and ICA with SVM. Note that all fine tunning will be performed on the base system which is PCA-SVM.

A. Dataset Pre-processing

After importing the dataset, we have the following statistics:

Number of Data Samples: 1288
Size of a data sample: 1850
Number of Class Labels: 7
Dimensions: 62x47

Note that as the dataset can have multiple images of the same person, for that reason we are restricting the number of faces per person to 70. The dataset is then structured as a ratio of 25% testing and 75% training which is presented by Belhumeur et al. (1997) to be the ideal test/train split for optimal recognition. As shown above we have 1288 samples with 7 classes corresponding to 7 people and image dimensions of 62x47 that is defined by the number of pixels as 2914 in total. In order to use PCA to reduce that high dimensionality, we first need to find the optimum number of principle components, this is done by looking at the variance.

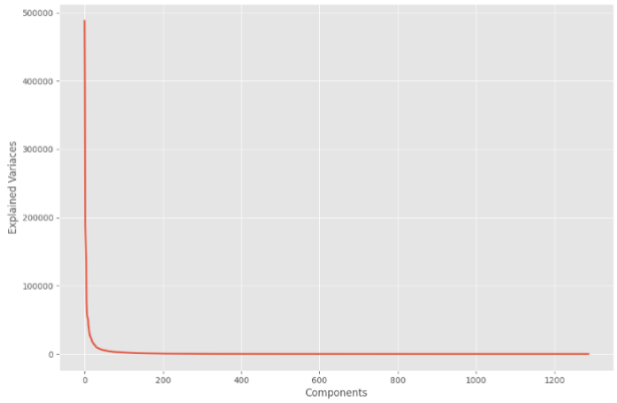

Figure 6 - Explained Variance.

Explained variance is the variance explained by each of the principal eigenvectors, which represent the principal components that contain most of the information needed to represent the features (Kumar, 2020, online). In the figure above we can see that 100 and above components can represent the entire data, this is also shown by looking at the cumulative variance shown below that finds the ideal number of components to be above 140, this will be tested later on our paper, but as an initial design the number the number of components is set to 150.

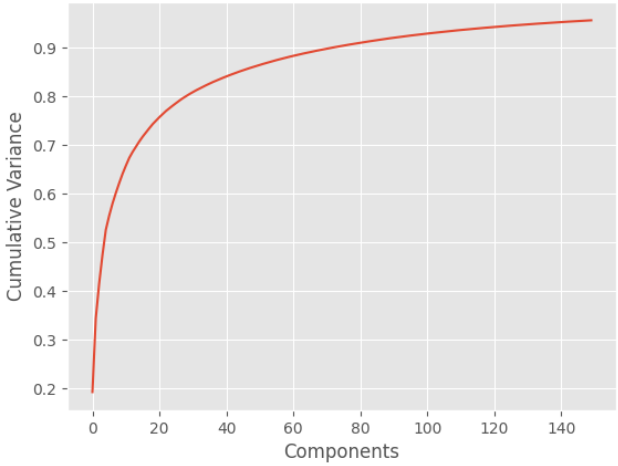

Figure 7 - Cumulative Variance.

B. *System Developemnt*

Before applying the PCA method to our data, we are first interested in determining the optimal kernel function for our SVM classifier, for that purpose we have used our dataset against the three main methods for classification: Naïve Bayes, Polynomial, and RBF. Note that as we are interested in the best function, the comparison only requires 2% percent as testing to determine the ideal function.

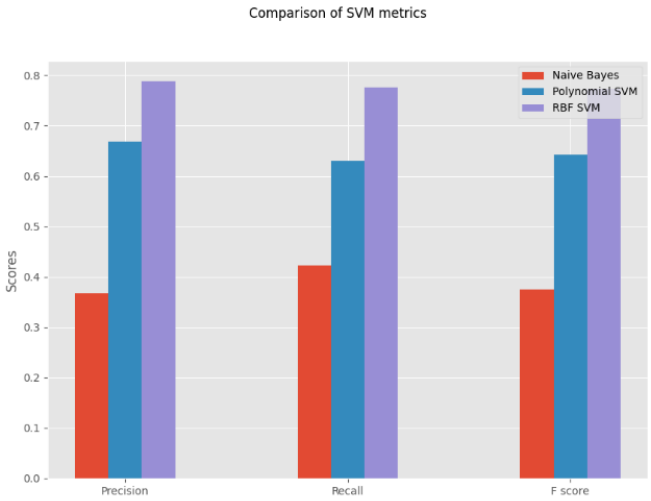

Figure 8 - SVM Metrics Comparison.

As shown in the figure above from our comparisons the RBF is the optimal function to use for our dataset. After we applied PCA the high dimensionality has been reduced from 2914 to 150 and our training set dimensions are as follow:

Training Set	Testing Set
(966, 1850)	(322, 1850)

In order to configure the SVM classifier properly we require several parameters to be tuned, as we already found the kernel, we are now looking to determine the **C (Regularization),** a penalty parameter, which represents misclassification or error term, and **Gamma,** which determines how far influences the calculation of possible line of separation (Clare Liu, 2020, online). We can search for the right hyper-parameters (parameters that are not directly learnt within estimators) using Grid Search, which can "methodically build and evaluate a model for each combination of algorithm parameters specified in a grid" (Clare Liu, 2020, online).

After applying the Grid Search, we have found the estimator:

Best estimator found by grid search:
SVC(C=1000.0, class_weight='balanced', gamma=0.005)

The PCA-SVM system for 150 components had an accuracy of 84%, the prediction is shown below, it is important to note that the training data was provided already as a greyscale image and did not required any modifications.

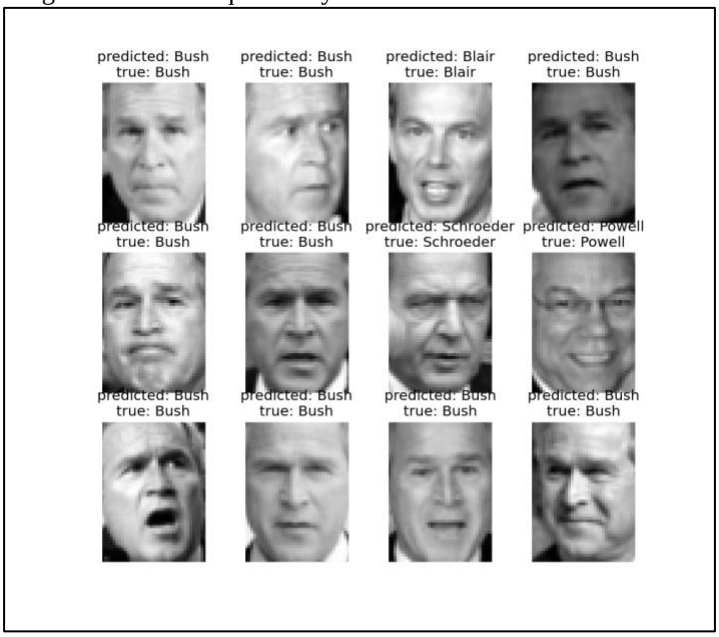

Figure 9 - Prediction for PCA-SVM

4. RESULTS

The initial PCA-SVM system has been adjusted to allow for using the parameters described above with ICA-SVM and LDA-SVM, dimensionality is reduced using PCA and for the case of LDA-SVM, it has been used as a pre-step to applying the algorithm. The three models were trained on 50, 100, 150, and 200 components for each of the algorithms. We are providing an overview of the results below:

Number of Components	Accuracy		
Eigenfaces	**PCA**	**ICA**	**LDA**
50	83.23%	74.53%	75.78%
100	85.71%	80.12%	81.99%
150	84.47%	83.54%	85.40%
200	84.47%	83.54%	83.23%

Number of Components	Time		
Eigenfaces	**PCA**	**ICA**	**LDA**
50	13.413	15.844	17.275
100	20.495	23.405	8.68
150	29.79	34.348	6.561
200	37.064	44.104	4.661

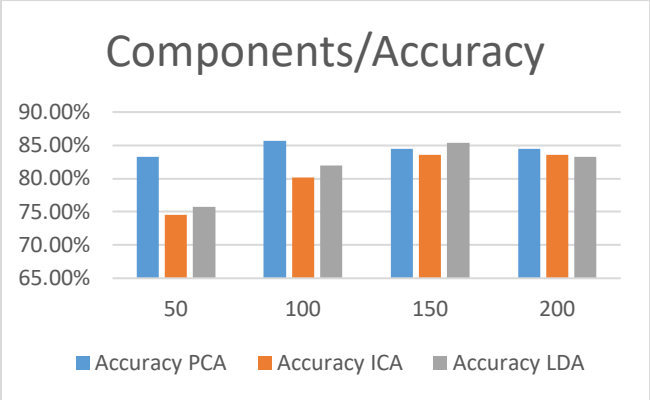

Figure 10 - Accuracy by number of components.

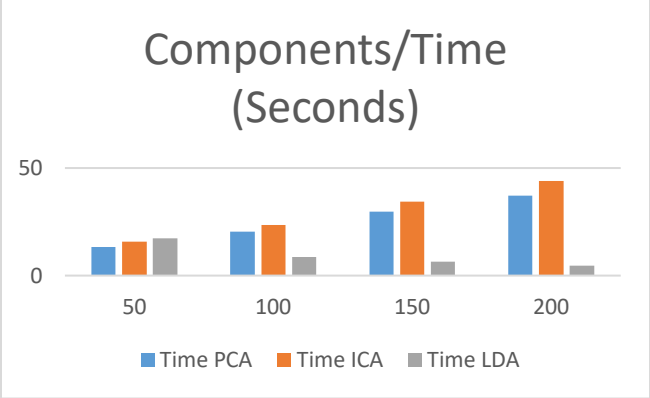

Figure 11 - Training time by number of components.

Generally, by looking at Figure 10 and Figure 11 we can see that most algorithms will converge with the highest accuracy for 150 components, PCA does appear to keep a consistent accuracy regardless of the component number in contrast with the other two method which increase in accuracy with the number of components.

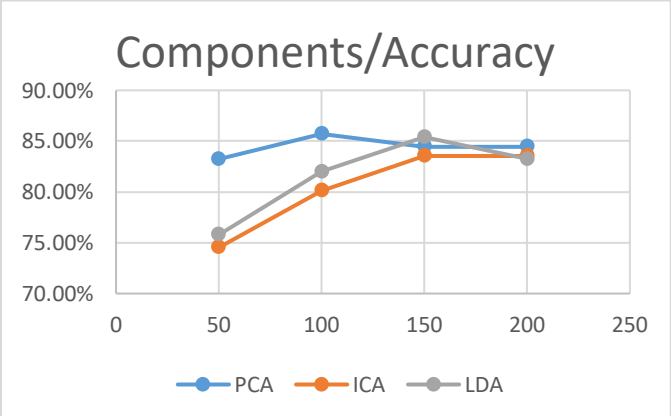

Figure 12 - Components vs Accuracy.

In the graph above we can see the increase in accuracy consequently with the increase in component numbers, except PCA which appears to reach its highest accuracy level at the 100 components mark and remains steady after that. Similarly, ICA-SVM shows an increase with the number of components reaching the highest accuracy at 150 components and remaining steady afterwards. LDA-SVM on the other hand reaches the highest accuracy also at 150 components but then registers a decrease in accuracy with the increase of components over 150.

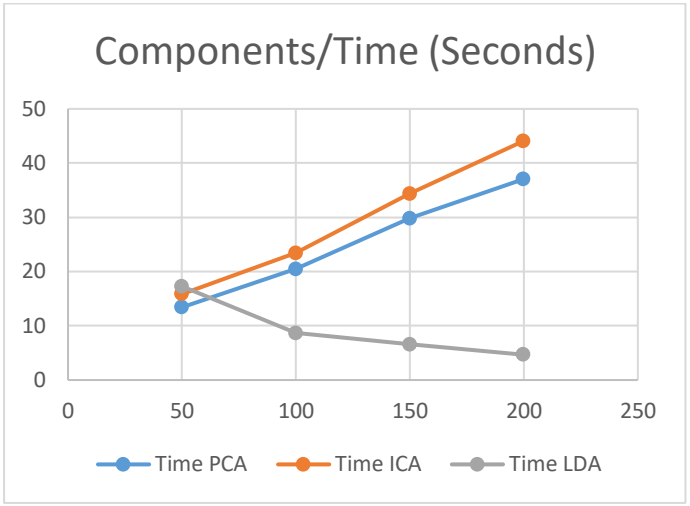

Figure 13 - Components vs Training Time.

It is interesting to note that LDA-SVM is decreasing in training time with the increase of components while PCA-SVM and ICA-SVM are seeing an increase in training time with the increase in the number of components. Looking at the reconstruction of the faces, as expected the eigenfaces generated by PCA-SVM contains features such as shadows and brightness, as well as each eigenface after the other has less information and increased noise.

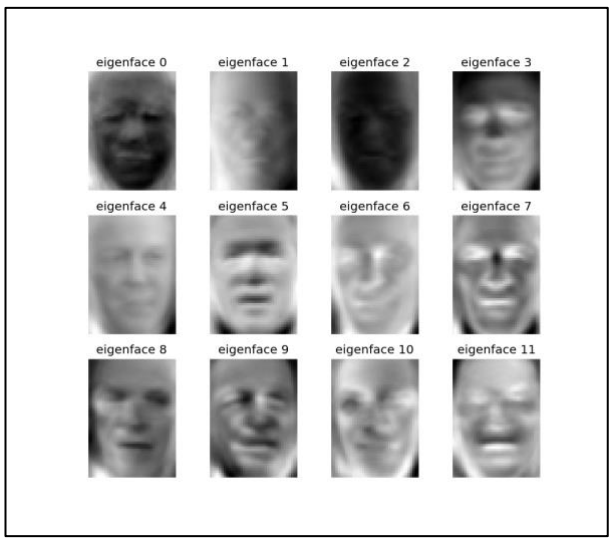

Figure 14 - PCA Eigenfaces.

In contrast, the ICA-SVM in Figure 15 captures more facial elements such as features like the nose, eyes, lips, and shape, but it is less intuitive than the PCA eigenfaces, the overall smoothness of the face is considerably lower than in the PCA.

The LDA-SVM' fisherface's representations (Figure 16) appear better in contrast with PCA, ICA's eigenfaces with the overall face structure clearer and more defined, though we are losing features such as the eyes, nose and mouth. Regarding the accuracy and time, we observed that all three models perform relatively similarly, while the face reconstruction techniques visibly differ they allow for face recognition.

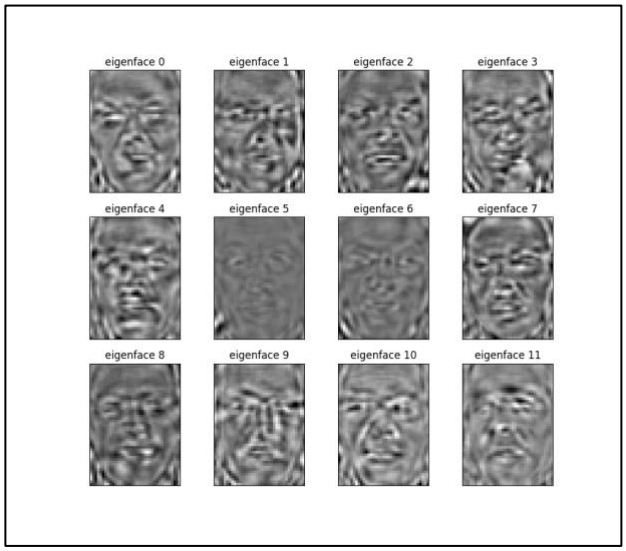

Figure 15 - ICA Eigenfaces.

In our work, when using the LDA algorithm we implemented a combination of PCA and LDA classified by SVM, this was the optimal solution that guarantees the dimensionality reduction and provide better classification.

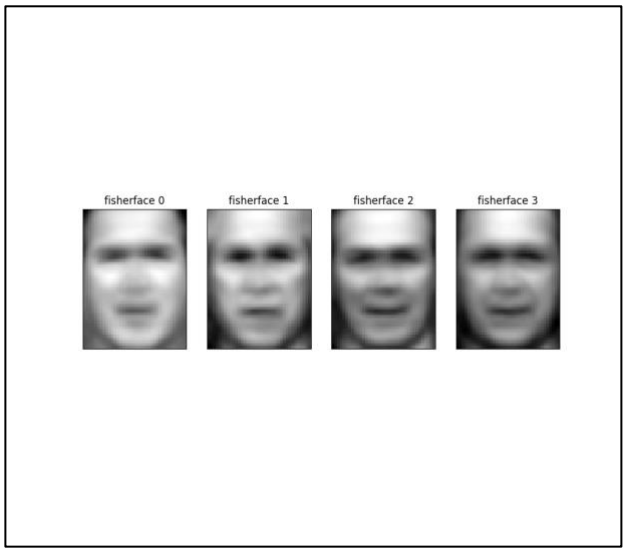

Figure 16 - LDA Fisherfaces.

As we presented throughout our paper PCA was initially designed for dimensionality reduction, in this work (LDA-SVM) its objective was to prepare data for LDA in order to perform the fisherface reconstruction shown above in Figure 16, the within-class scatter matrix we discussed previously is, since we are dealing with high-dimensional data singular. A singular matrix is not invertible and does not allow for further calculation of projection vector w. The idea behind combining PCA and LDA is that since our within-class scatter matrix keeps majority of the number of samples and number of classes as non-zero eigenvalues, then PCA can keep most eigenvectors number of samples and classes. By performing LDA to compute the projections in the lower-dimensional subspace, after an input image is provided, by projecting on to the same vector w can compare the distance between the two classes.

As the PCA-SVM has performed best in terms of accuracy, we are providing below the confusion matrix. Please note that the confusion matrix and plots of each method's heatmaps are available in the results we provided with this paper.

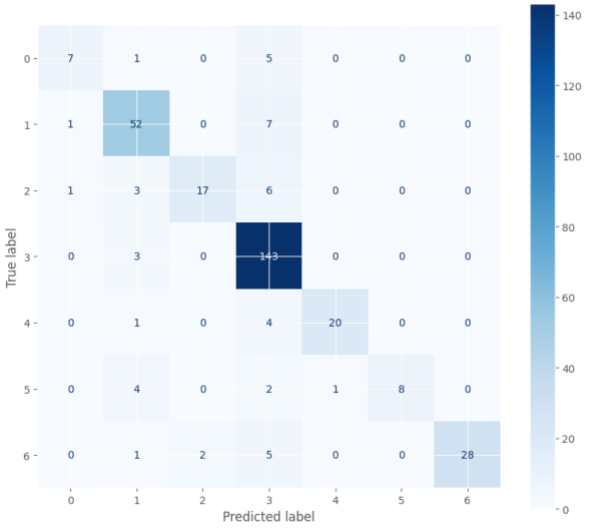

Figure 17 - PCA-SVM Confusion Matrix.

From the figure above we can see that most cases were classified correctly with some margin of error present, but overall, the precision is good.

5. Conclusion

This paper presented a machine learning implementation using the SVM Classifier and three popular appearance-based algorithms PCA, ICA, LDA, and a comparative study of the algorithms using equal conditions across all methods. We found that while no metric can replace real-world conditions, when using the LFW dataset with the parameters presented in the paper we can achieve a score of above 80% accuracy. The PCA-SVM implementation has performed best with a score of 85.71%, followed by the LDA-SVM with 85.40% and ICA-SVM with 83.54%. Further research is needed and could include testing with various datasets, changing the SVM's classifier method, or replacing the metrics. Such research could produce a better and deeper understanding of individual algorithm performance and possible allow for a framework to be developed using a combination of methods.

REFERENCES

Ashok Rao and Noushath, S. (2010) Subspace methods for face recognition. *Computer Science Review*, 4(1), pp. 1–17.

Bartlett, M.S., Movellan, J.R. and Sejnowski, T.J. (2002) Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13(6), pp. 1450–1464.

Belhumeur, P.N., Hespanha, J.P. and Kriegman, D.J. (1996) Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. In: Buxton, B. and Cipolla, R. (eds.) *Computer Vision — ECCV '96*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 43–58.

Belhumeur, P.N., Hespanha, J.P. and Kriegman, D.J. (1997) Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7), pp. 711–720.

Clare Liu (2020) *SVM Hyperparameter Tuning using GridSearchCV*. [Online] Velocity Business Solutions Limited. Available from : https://www.vebuso.com/2020/03/svm-hyperparameter-tuning-using-gridsearchcv/ [Accessed 06/05/21].

Delac, K., Grgic, M. and Grgic, S. (2005) *A Comparative Study of PCA, ICA AND LDA*. [Online] undefined. Available from : https://www.semanticscholar.org/paper/A-Comparative-Study-of-PCA%2C-ICA-AND-LDA-Delac-Grgic/18727adf3e63de90674fcafd8b1f5e0059669e84 [Accessed 02/05/21].

Draper, B.A. et al. (2003) Recognizing faces with PCA and ICA. *Computer Vision and Image Understanding*, 91(1–2), pp. 115–137.

Fisher, R.A. (1936) THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. *Annals of Eugenics*, 7(2), pp. 179–188.

Getting Started — scikit-learn 0.24.2 documentation. Available from : https://scikit-learn.org/stable/getting_started.html [Accessed 06/05/21].

Guo, G., Li, S.Z. and Chan, K.L. (2001) Support vector machines for face recognition. *Image and Vision Computing*, 19(9–10), pp. 631–638.

Huang, G. et al. (2008) *Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments*. [Online] undefined. Available from : https://www.semanticscholar.org/paper/Labeled-Faces-in-the-Wild%3A-A-Database-forStudying-Huang-Mattar/c6b3ca4f939e36a9679a70e14ce8b1bbbc5618f3 [Accessed 02/05/21].

Huang, G.B. et al. (2008) Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments. In: Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition.

Kumar, A. (2020) *PCA Explained Variance Concepts with Python Example*. [Online] Data Analytics. Available from : https://vitalflux.com/pca-explained-variance-concept-python-example/ [Accessed 06/05/21].

Phillips, P.J. (1998) Support Vector Machines Applied to Face Recognition. [Online] Available from: https://www.nist.gov/publications/support-vector-machines-applied-face-recognition [Accessed 02/05/2021].

scikit-learn *scikit-learn: A set of python modules for machine learning and data mining*.

Slavkovic, M. and Jevtic, D. (2012) Face recognition using eigenface approach. *Serbian Journal of Electrical Engineering*, 9(1), pp. 121–130.

Sodhi, K.S. and Lal, M. (2013) *FACE RECOGNITION USING PCA, LDA AND VARIOUS DISTANCE CLASSIFIERS*. [Online] undefined. Available from : https://www.semanticscholar.org/paper/FACE-RECOGNITION-USING-PCA%2C-LDA-AND-VARIOUS-Sodhi-Lal/a019606dbd26c6e1acd1277e8f7b2d526710b9b7 [Accessed 05/05/21].

Stewart Bartlett, M., Lades, M.H. and Sejnowski, T.J. (1998) Independent component representations for face recognition. In: Rogowitz, B.E. and Pappas, T.N. (eds.) Photonics West '98 Electronic Imaging. San Jose, CA, pp. 528–539.

Turk, M. and Pentland, A. (1991) Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1), pp. 71–86.

Zhou, C. et al. (2013) Face recognition based on PCA image reconstruction and LDA. *Optik*, 124(22), pp. 5599–5603.