Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2019

David Moreira de Almeida

Reconhecimento Facial através de Representações Esparsas

Face Recognition via Sparse Representation

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2019

David Moreira de Almeida

Reconhecimento Facial através de Representações Esparsas

Face Recognition via Sparse Representation

David Moreira de Almeida

Reconhecimento Facial através de Representações Esparsas

Face Recognition via Sparse Representation

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia de Computadores e Telemática, realizada sob a orientação científica da Professora Doutora Ana Maria Perfeito Tomé, Professora Associada da Universidade de Aveiro, e do Professor Armando José Formoso de Pinho, Professor Associado com Agregação da Universidade de Aveiro.

o júri / the jury

presidente / president	**Professor Doutor Arnaldo Silva Rodrigues de Oliveira** Professor Auxiliar, Universidade de Aveiro (por delegação da Reitora da Universidade de Aveiro)
	Professora Doutora Ana Maria Perfeito Tomé Professora Associada, Universidade de Aveiro (orientadora)
vogais / examiners committee	**Doutor Luís Miguel Almeida da Silva** Professor Auxiliar Convidado

agradecimentos / acknowledgements

Gostaria de agradecer à professora Ana Maria Tomé e ao professor Armando Pinho por todo o apoio e orientação que me deram durante o desenvolvimento deste trabalho.

À minha família, por todo o apoio que me deram durante os meus anos formativos, tanto financeiro como pessoal. Sem ela nunca teria tido as oportunidades que tive.

À minha namorada que sempre esteve lá para me apoiar tanto nos momentos altos como nos baixos.
A todos os meus amigos que me acompanharam por este percurso e com quem partilhei momentos que nunca irei esquecer.

Finalmente queria deixar um obrigado a todos os que contribuíram para o meu sucesso pessoal e académico.

Palavras Chave

Reconhecimento facial, codificação esparsa, representações esparsas, aprendizagem de dicionários, visão de computador

Resumo

Recentemente houve um pico de interesse na área de reconhecimento facial, devido especialmente aos desenvolvimentos relacionados com "deep learning". Estes estimularam o interesse na área, não apenas numa perspectiva académica, mas também numa comercial. Apesar de tais métodos fornecerem a melhor precisão ao executar tarefas de reconhecimento facial, eles geralmente requerem milhões de imagens de faces, bastante poder de processamento e uma quantidade substantial de tempo para desenvolver.

Nos últimos anos, representações esparsas foram aplicadas com sucesso a diversas aplicações de visão de computador. Uma dessas aplicações é reconhecimento facial. Um dos primeiros metodos propostos para tal tarefa foi o "Sparse Representation Based Classification (SRC)".

Entretanto, vários diferentes metodos baseados no SRC, foram propostos. Estes incluem métodos de aprendizagem de dicionários e métodos baseados em classificaçao de "patches" de imagens.

O objetivo desta tese é estudar o reconhecimento facial utilizando representações esparsas. Múltiplos métodos vão ser explorados e alguns deles vão ser testados extensivamente de modo a providenciar uma visão compreensiva da area.

Keywords Face recognition, sparse coding, sparse representation, dictionary learning, computer vision

Abstract Face recognition has recently seen a peek in interest due to developments in deep learning. These developments incited great attention to the field, not only from the research community, but also from a commercial perspective. While such methods provide the best accuracies when performing face recognition tasks, they often require millions of face images, a substantial amount of processing power and a considerable amount of time to develop.

In the recent years, sparse representations have been successfully applied to a number of computer vision applications. One of those applications is face recognition. One of the first methods proposed for this task was the Sparse Representation Based Classification (SRC).

Since then, several different methods, based on SRC have been proposed. These include dictionary learning based methods, as well as patch based classification.

This thesis aims to study face recognition using sparse classification. Multiple methods will be explored, and some of these will be tested extensively in order to provide a comprehensive view of the field.

Contents

List of Figures

List of Tables

Acronyms

ALSR Accumulative Local Sparse Representation. 29

ASR Adaptive Sparse Representation. 40

CCTV Closed-circuit television. 3

CNN Convolutional Neural Networks. 1, 7

FISTA Fast Iterative Shrinkage-Thresholding Algorithm. 15

HOG Histogram of Oriented Gradients. 5

ISTA Iterative Shrinkage-Thresholding Algorithm. 15

LARS Least Angle Regression. 15

LASSO Least Absolute Shrinkage and Selection Operator. 15

OMP Orthogonal Matching Pursuit. 15

PCA Principal Component Analysis. 17, 48

SCI Sparsity Concentration Index. 20

SRC Sparse Representation based Classification. 13

SVD Singular Value Decomposition. 26, 48

SVM Support Vector Machine. 7

Chapter 1

Introduction

Recently, the interest in computer vision applications has been increasing at an incredible pace. This increase initiated mainly due to the developments in the area of deep learning, namely, with the use of CNN's (Convolutional Neural Networks).

Face recognition is one of the most popular and important applications of computer vision. The ability to automatically recognize people has innumerous applications in multiple different areas. While face recognition has been studied for decades, recently it has seen a boost in interest due to the developments of deep learning.

While the results attained when using deep learning [4] cannot be matched via traditional face recognition algorithms, in order to train such complex models, unreasonably large datasets need to be used (in the orders of millions of images) and they require an extremely high amount of computing resources. Therefore, the use of more traditional machine learning techniques should not be disregarded entirely on behalf of deep learning techniques.

The idea behind sparse representation is that a vector can be represented using a linear combination of elements from a dictionary. Over the years, sparse representation approaches have been applied to many different computer vision applications. One such application is face recognition. While sparse representation based approaches cannot obtain results on par with the deep learning approaches, they do not require a large amount of training samples. Significant results can still be attained, especially when compared with the more traditional face recognition algorithms.

1.1 Objectives

The main objective of this dissertation was to study, develop and test sparse representation based algorithms for face recognition applications, including:

- An effective and concise description of sparse representation.

- An overview of dictionary learning algorithms, especially the ones aimed to face recognition.
- An exposition of a state of the art face recognition algorithm based on sparse representation.
- Detailed experimental results of sparse representation based algorithms tested using appropriate face recognition datasets.

1.2 Thesis Outline

This dissertation is composed of six chapters. Excluding this introductory chapter, the remaining of this dissertation is organized as follows:

- Chapter 2: This chapter is dedicated to the introduction of face recognition. This includes an explanation of its importance, as well as a description of what constitutes a face recognition system.
- Chapter 3: This chapter aims to explain sparse representation and to relate it to face recognition. It also introduces the dictionary learning algorithms which are commonly used in face recognition.
- Chapter 4: This chapter introduces a recent state of the art face recognition algorithm based on sparse representation.
- Chapter 5: This chapter displays the experimental results of the tests conducted during the development of this dissertation.
- Chapter 6: This chapter presents the conclusions of the work performed.

Chapter 2

Face Recognition

This chapter discusses the importance of face recognition currently. It will explain the different stages of a complete face recognition system, and introduce a few datasets that will be used throughout the dissertation.

2.1 Importance

In recent years, face recognition technology has been rapidly introduced in multiple aspects of our lives. Face recognition algorithms allow two tasks to be performed automatically:

1. Verification: Given a face image, assuring that the image is not from an unknown individual. The face image should belong to a known and authorized subject. This can be used to grant/deny access to a resource.

2. Identification: Given a face image of an unknown subject, determine the identity of that individual by using a database of known identities.

Face recognition technology is currently being applied to a plenty of different scenarios. Some of these application are:

- Authentication: Either personal or corporate authentication are common applications of face recognition technology. In a corporate scenario, this could mean giving access to specific personnel to appropriate resources or locations within a company. Regarding personal authentication, it is becoming increasingly common to use face recognition (instead of a fingerprint scanner or a pin code) in order to unlock personal devices (e.g. personal computers, smartphones).

- Surveillance: Using CCTV cameras, large areas can be easily monitored in order to look for known criminals or unauthorized personnel.

- Marketing: By recognizing the person, and namely his/her characteristics (e.g. gender, age), tailored add/campaigns can be created in real time in order to target that specific person.

- Identity tracking: Amazon is launching stores that do not require a cashier. These stores register the people that enter the store and by means of face recognition track that person's activity. Then, given that person's activity it charges the clients directly to their account upon them leaving the store.

The above applications are only a few of many that underline the importance of face recognition nowadays.

2.2 Face Recognition System

A face recognition system is composed of multiple stages that start with an input (e.g. image or video frame) and ends with an output of the identity of the face/faces present in the input.

The multiple stages of a face recognition system and their sequence are illustrated in Figure 2.1.

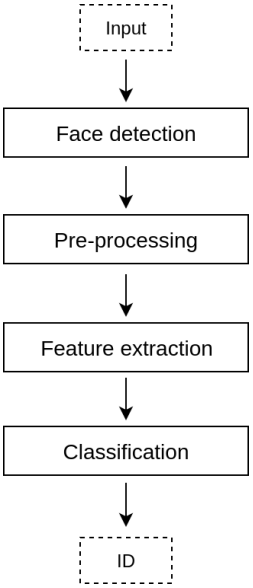


Figure 2.1: Face recognition stages.

2.2.1 Face Detection

There are multiple methods to perform face detection in 2d images. Two of the most famous are the Viola-Jones object detection framework [5] and the Histogram of Oriented Gradients (HOG) method [6].

These algorithms allow for efficient and accurate face detection. They are fast enough to be usable in real time video face recognition. Having detected a face in an image, that face can be isolated and the next stage of the aforementioned stages can proceed.

2.2.2 Pre-processing

There are some difficulties that need to be tackled in order to create a robust face recognition system. Such a system need to be able to deal with situations such as poor lighting or face misalignment. While the classifier should be partially resistant to such factors, there are steps that can be taken in the pre-processing stage in order to improve the accuracy of the face recognition system.

Illumination

Illumination conditions can greatly influence the results of face recognition methods. This is especially true, when the illumination conditions of the testing images are not the same as the training images.

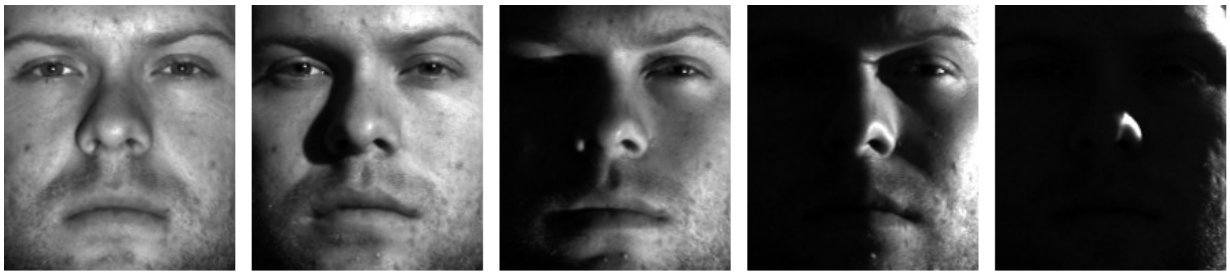


Figure 2.2: Sample images from the YALE Dataset showing different illumination patterns.

Figure 2.2 shows 5 images from the same person with varying illumination conditions. On the left, a well lighted face image is presented. Then, the images start to feature more aggressive lighting conditions as they progress to the right.

There are some pre-processing methods that can be applied in order to attenuate the effects of the lighting condition on the quality of the face image. Examples of these are contrast stretching, histogram equalization and adaptive equalization. Examples of the efficiency of these methods can be observed in [7].

Figure 2.3: Sample images from the YALE Dataset showing different illumination patterns (Histograms equalized).

Figure 2.3 shows the same face images shown previously in Figure 2.2 but with their histograms normalized. Despite not being perfectly matched regarding their illumination, the differences in lighting between the normalized pictures are much less pronounced than their original counterparts.

Despite being able to attenuate the effects of varying illumination by using proper preprocessing methods, in order to achieve good recognition accuracy, the classification algorithms should be able to withstand some illumination inconsistency.

Face misalignment

Different facial poses can also make it difficult to perform a correct classification. Face alignment methods can help normalize the poses of the face images by identifying key points in a face and performing operations such as translation, rotation and scaling. While this works for slightly misaligned face images, when the misalignment becomes substantial these measures are not enough to ensure a correct classification.

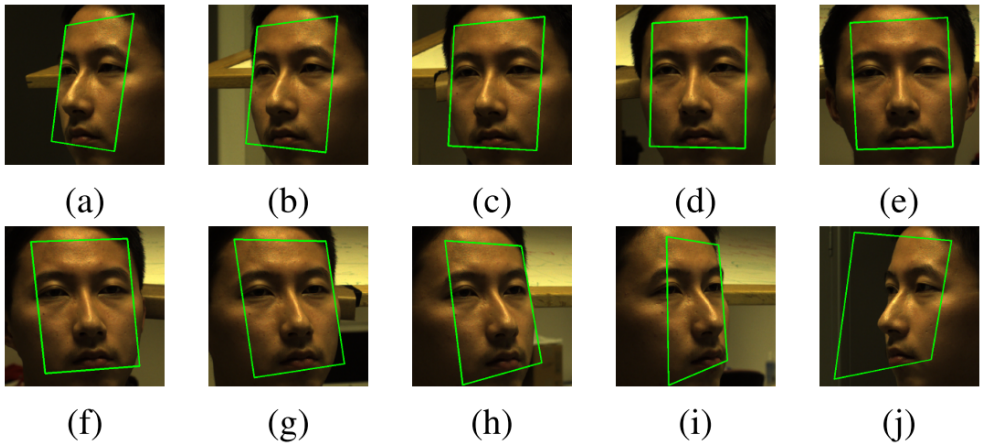


Figure 2.4: Face images with different poses. Taken from [1].

Figure 2.4 shows face images of the same subject in several different poses. The images from (a) to (i) feature misalignments under 45°. The image (j) has a misalignment greater than 45°. After performing image alignment, all the images except for (j) can be correctly classified. Image (j) is too misaligned to allow for proper alignment and classification using sparse representation based face recognition methods.

If possible, face alignment methods should therefore always be applied, in order to improve the accuracy of face recognition systems.

2.2.3 Feature extraction and classification

The feature extraction and the classification of face images depend on the approach of the face recognition system. There are holistic, feature based, and hybrid approaches.

Holistic approaches use the whole image in order to perform the classification. An example of feature extraction of this type of approach is to simply use the pixels of the face image itself as features.

One very popular example that applies an holistic approach is the Eigenface algorithm [8]. This method is based on the PCA technique and allows the calculation of eigenfaces from the face images. The eigenfaces consist of sets of eigenvectors. These can be used to train a classifier. That can in turn be used to classify test images, by extracting the eigenfaces from those images and using the classifier to get the correct classes.

While holistic approaches are easy to implement, they are particularly sensitive to variations on the face images (e.g. illumination, pose, expression).

Feature based approaches only consider some facial features in the classification. An example of feature extraction of such an approach would be to use a trained CNN in order to extract features from the face image. CNN's are trained to look for specific patterns in the face images. The outputs of a CNN can then be used to train and test a classification model (e.g. SVM [9]).

Hybrid approaches combine the characteristics of both holistic and feature based methods in order to improve the ability of the face recognition system.

2.3 Further Challenges

Besides the aforementioned difficulties (illumination, pose variance) there are more challenges that cripple the performance of face recognition systems.

2.3.1 Occlusion

Some face images can feature occluding elements that hinder the accuracy of face recognition algorithms. Examples of such elements are: sunglasses, scarfs and long hair in front of the face.

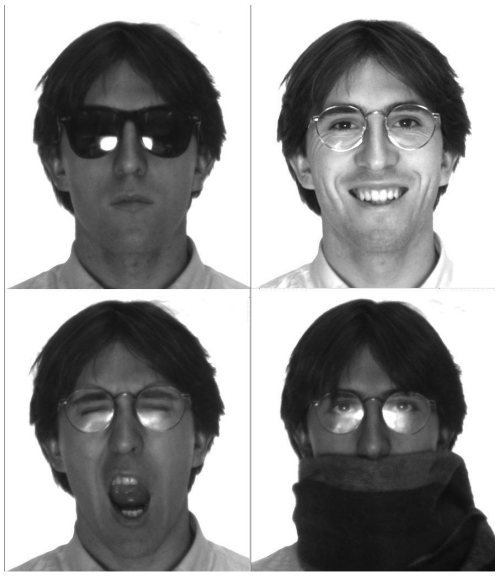


Figure 2.5: Sample images from one subject of the AR Dataset [2].

Figure 2.5 shows some pictures of a subject from the AR Dataset under different occlusion conditions. The picture in which the man is using sunglasses and the picture in which he is using a scarf feature full occlusion of some facial attributes. The pictures where he is only wearing clear glasses feature partial occlusion given the reflections of light seen in the glasses.

Occlusion can be especially challenging for holistic face recognition algorithms (since the face is considered as a whole in order to perform the recognition). Feature based face recognition methods, that rely on extracting different features from the face images, are not as susceptible, since it is possible to ignore the elements being occluded.

2.3.2 Facial expressions

Another factor that difficults the correct classification of face images are different facial expressions. In Figure 2.5, the second and third pictures are of the same person, but with widely different facial expression. If the algorithm was trained with only neutral expressions, getting a correct classification in test images with different, very pronounced expressions will be difficult. This affects both holistic and feature based approaches, since important facial attributes can have a significantly different appearance depending on the facial expression present in the image.

2.3.3 Resolution

The resolution of a face image can affect the face recognition algorithms in different ways. If the resolution is low, the image might not have enough detail for the algorithm to perform the classification. On the other hand, dealing with high resolution face images can cripple the performance of the algorithm, making it run slowly in each test image.

There is another problem deriving from the difference in resolution of face images. In most cases, the pictures are taken in unconstrained situations, possibly with different equipment. This means that the face images can either be captured with devices with different resolutions or the subjects might be at different distances from the capture device. Therefore, more frequently than not, the resolution of the face images will be different from each other. Some algorithms require that the images have the same resolution in order to perform classification. This can be solved by applying downscaling or upscaling to the images in order to assure that they all have the same resolution.

2.4 Datasets

Many different datasets containing face images are available for research. The datasets differ from one another on multiple factors. In order to properly test classification algorithms, there should be a selection of multiple datasets, which cover different conditions. The datasets mentioned below were selected since they all present different characteristics. This way they are able to simulate multiple use cases of face recognition technology.

2.4.1 Georgia Tech Face Database

The Georgia Tech Face Database [10] consists of photos of 50 people taken in an office in an informal setting. Each person has 15 photos, and the photos were taken in three sessions distributed through five months (between 01/06/99 and 15/11/99).

The images present a reasonable amount of inter-class variation, since most of the people present have distinctly different facial attributes. It also presents a reasonable amount of intra-class variation, since images of the same person present some variation in lighting conditions, and each person has multiple photos with different pose misalignments. Some photos also feature partial occlusion (e.g. the use of glasses, hair in front of the face).

Figure 2.6 shows some images from the dataset. These images have been pre-aligned, but the dataset itself is not. For the remaining of this dissertation, this dataset will be referred to as the GT dataset.

Figure 2.6: Sample images from the GT Dataset.

2.4.2 The Extended Yale Face Database B

The Extended Yale Face Database B [11] consists of photos of 39 people taken in a highly controlled environment. Each person has 64 photos. The photos are cropped so that only the face fills the frame.

All the faces are perfectly aligned, but for each person, there are some differences regarding the facial expressions. The focus of this dataset is in illumination. For each person, the photos were captured under 64 different illumination conditions, which along with the differences in facial expressions provides a good amount of intra-class variation.

Figure 2.7 shows some samples of the Extended Yale Face Database B. Each column corresponds to a different person, and it is possible to see some of the different illumination conditions. For the remaining of this dissertation, this dataset will be refereed to as the YALE dataset.

2.4.3 Labeled Faces of the Wild

The Labeled Faces of the Wild dataset [12] contains more than 13000 images of faces. These images were collected from the internet and feature fully unconstrained conditions.

For our tests, a pre-aligned version of this dataset using deep learning will be used. This version is known as deep funneled LFWa [13]. From these images only the people who have 11 or more face images are considered. This subset of images is comprised of 4174 images of 143 people.

Despite the images being pre-aligned, the faces have huge variations in pose, lighting and facial expressions, therefore making this dataset very challenging regarding face recognition.

Figure 2.7: Sample images from the Extended Yale Face Database B.

Figure 2.8 shows some examples from this dataset. For the remaining of this dissertation, this dataset will be refereed to as the LFW dataset.

2.5 Conclusion

This chapter introduced the concept of face recognition by showing its importance and explaining the different stages of a face recognition system. It also introduced the datasets that are used to test the algorithms that will be introduced throughout this dissertation. It should be noted that these datasets are relatively small in size.

In the next chapters, sparse representation approaches will be presented. These approaches only require these small datasets in order to perform classification. This opposes deep learning methods, which need datasets containing millions of labeled examples in order to be trained.

Figure 2.8: Sample images from the deep funneled LFW dataset.

Chapter 3

Sparse Representation

This chapter will explain what sparse representation is and how one can achieve a sparse representation. It will expose the advantages of using a sparse signal, explain what is a dictionary and how one can obtain it. It will describe how to perform Sparse Representation based Classification (SRC), and relate it to dictionary learning methods that use the principles of SRC to perform classification.

3.1 Definition

Sparse Representation aims to provide sparse solutions for a system of linear equations. At its core, the objective is to solve the equation:

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x}, \tag{3.1}$$

where $\boldsymbol{y}$ corresponds to the signal that is to be represented, $\boldsymbol{D}$ to an over-complete dictionary matrix and $\boldsymbol{x}$ to an array of sparse coefficients that, when multiplied with $\boldsymbol{D}$, accurately reconstructs the signal $\boldsymbol{y}$.

Figure 3.1 shows an illustration of (3.1) as well as the dimensions of the operands of that equation. The signal $\boldsymbol{y}$ is illustrated by a vector in which $\boldsymbol{y} \in \mathbb{R}^m$, and can be accurately reconstructed by the product of the matrix $\boldsymbol{D}$, where $\boldsymbol{D} \in \mathbb{R}^{m \times n}$, and the set of coefficients $\boldsymbol{x}$ correspond to another vector, in which $\boldsymbol{x} \in \mathbb{R}^n$.

Each column from the dictionary $\boldsymbol{D}$ is often called an atom. Therefore, after obtaining the sparse representation, the signal $\boldsymbol{y}$ corresponds to a linear combination of the atoms of $\boldsymbol{D}$. The non-zero coefficients of $\boldsymbol{x}$ control which atoms of the dictionary $\boldsymbol{D}$ are suitable to represent $\boldsymbol{y}$.

In order to obtain the dictionary, multiple approaches can be taken. These will be discussed afterwards.

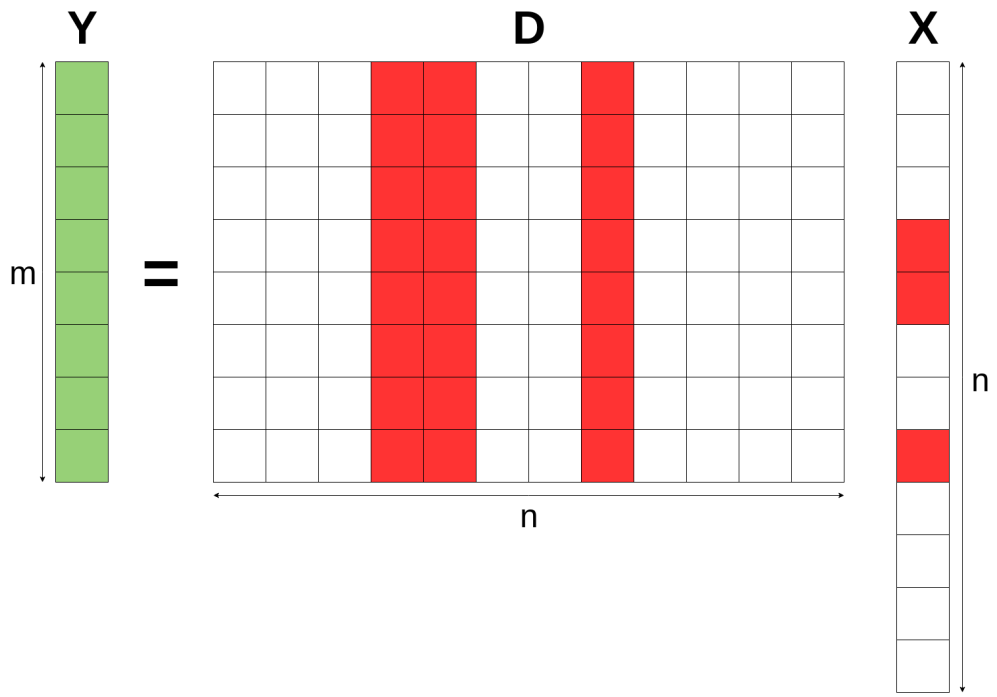


Figure 3.1: Sparse Representation Illustration. Only a few atoms of the dictionary are needed in order to represent the signal $\boldsymbol{y}$ as most of the coefficients of the $\boldsymbol{x}$ are zero.

The concrete objective of sparse coding, given a signal $\boldsymbol{y}$ and an overcomplete dictionary $\boldsymbol{D}$, is to produce the sparsest array of coefficients $\boldsymbol{x}$, which, by means of a linear combination of the atoms of $\boldsymbol{D}$, reconstruct the signal $\boldsymbol{y}$ [14].

In order to reach such solution, a constrained optimization problem needs to be solved, namely,

$$\boldsymbol{x}^* = \min_{\boldsymbol{x}} \ \Psi(\boldsymbol{x}) \quad \text{s.t.} \ \ f(\boldsymbol{y}, \boldsymbol{D}\boldsymbol{x}) = 0. \tag{3.2}$$

This optimization is more commonly represented in its Lagrangian dual form, given by

$$\boldsymbol{x}^* = \min_{\boldsymbol{x}} f(\boldsymbol{y}, \boldsymbol{D}\boldsymbol{x}) + \lambda\Psi(\boldsymbol{x}), \tag{3.3}$$

where usually $f(\boldsymbol{y}, \boldsymbol{D}\boldsymbol{x})$ corresponds to the sum of squared error regarding the original signal and its reconstruction using the sparse coefficients. λ represents a regularization parameter that controls the trade-off between sparsity and the reconstruction error of the solution.

The obvious approach to choosing the function $\Psi(\boldsymbol{x})$ would be to use the l_0-norm, in the form of $\Psi(\boldsymbol{x}) = \|x\|_0$, since this norm is proportional to the sparsity of the solution. While

it would in theory provide the best results, in practice this corresponds to an NP-Hard problem [15] [16]. Still, there are a great deal of greedy or approximate algorithms that provide solutions to these problems (e.g. Orthogonal Matching Pursuit [17]).

Another approach that has had good results in inducing sparsity in the solution is the use of l_1-norm in the form of $\Psi(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$. This variation is usually preferred in literature since it corresponds to an optimization problem of a quasi-convex function (since it can have multiple local minima) and has been proven to produce good results [18].

It has been previously demonstrated that for most applications l_1-norm based algorithms also provide the sparsest solutions [19].

Examples of algorithms to solve this variation of the problem are the Least Absolute Shrinkage and Selection Operator (LASSO) method [20], Least Angle Regression (LARS) [21], Iterative Shrinkage-Thresholding Algorithm (ISTA) and its fast variant Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [22], among others.

3.2 Sparse Representation Applications

Sparse representation has had many different applications over the years. This can be explained by analyzing the advantages that a sparse representation has over a dense signal.

A sparse representation offers the following advantages over a dense representation:

- Data compression: Instead of storing the entire signal, only the coefficients and the dictionary need to be stored. Since the coefficient vector $\boldsymbol{x}$ is sparse by nature it is compact and uses a low amount of memory. Since the dictionary is overcomplete, it is usually considerably bigger than the signals to be represented. Therefore, if data-compression is our goal, using a dictionary would not be a good approach. However, if there is a need to store a large amount of signals, the memory saved by using the coefficients will greatly outweigh the memory used by the dictionary. This is also useful for signal transmission, since, if both parties have the same dictionary, only the sparse coefficients need to be sent, resulting in better transmission performance.

- Parsimony: Since the sparse signal is smaller and only the essential features are highlighted, it is much easier to analyze a sparse representation of a signal than the signal itself. In imaging, given a test image, this makes it easier to see the different components that constitute the image by analyzing the sparse coefficients and the dictionary atoms corresponding to them.

- Denoising: When a signal is partially corrupted with noise, if the underlying signal is sparse in nature, which is often the case, it can be denoised by reconstructing it using

its calculated sparse representation. This is particularly useful for performing image denoising, and to recover lost or corrupted parts of a partially corrupted image.

- Classification: If sparse representation is being used as a classification technique, ideally there is a sparse solution in which the coefficients only correspond to a unique class.

Figure 3.2: Sparse Coding based Denoising. From left to right: original image, noisy image, recovered image. [3]

3.2.1 Choosing a Dictionary

There are multiple approaches in order to obtain an overcomplete dictionary:

- Using a prebuilt dictionary: In some cases a previously built dictionary can be used. This is a dictionary obtained from a third party. Occasionally, it can have good performance (e.g. audio signal [23]), but this approach is never used in face recognition tasks.

- Creating a dictionary: For some applications, a dictionary obtained directly from training samples can be created. When dealing with images, a common approach is to stack the pixel columns in a vector, and each training image will correspond to a column of the dictionary. While this is the most common and simple approach, other approaches like dividing the images into patches and then stacking those patches can also be used.

- Learning a dictionary: A dictionary can also be learned by using dictionary learning algorithms like k-SVD [24] or the method of optimal directions (MOD) [25]. These methods start by using a predefined or random dictionary and then update its atoms by alternating between the sparse coding of the training images and a stage where the dictionary is updated in order to improve the sparse coding performance of the next iteration.

In image based approaches, it is not usual to use a prebuilt dictionary, since prebuilt dictionaries cannot convey correctly the information in the testing images. State of the art results can be achieved with either a dictionary created directly by using a training set of images or with a learned dictionary obtained by learning the dictionary with a training set of images. Learning a dictionary usually results in a more compact dictionary with more descriptive potential, but in some cases, especially if the number of training samples is large, and if there is a fair amount of variation in the images from the same class, using the images directly to build the dictionary can yield better results.

3.3 SRC

While Compressed Sensing (CS) and Sparse Representation had initially been applied successfully in other areas of study (e.g. Signal Denoising [26], Image Super-Resolution [27]), in 2009 a new method proposed by Wright et al. [28] suggested a new classification algorithm based on sparse representation, and applied in a face recognition task. While it is often related to face recognition, this classification algorithm is generic, and therefore should work for different types of signals [29].

3.3.1 Dictionary

In SRC the dictionary is created by directly using the training samples. These training samples are initially converted to grayscale. Since the dimension of the face images can be very large (when using 125x125 images we end up with vectors with 15625 entries) using the images themselves to form the atoms of the dictionary can lead to performance issues. In order to optimize performance, while keeping a good classification accuracy, dimensionality reduction has to be performed on the face images.

Some of the most popular dimensionality reduction methods are: Principal Component Analysis (PCA) [30], Random Projection [31], Image Downsampling [32], among others. In spite of any of them being suitable for dimensionality reduction, in [1] it was verified that, as long as the number of features remains high enough, the chosen method will have a negligible impact in the classification performance. Therefore, for the sake of simplicity, from now on the downsampling method will be used.

Having converted the images to grayscale and downsampled the images, the dictionary can now be formed by stacking the columns of the downsampled training images vertically and adding them as columns to our dictionary $\boldsymbol{D}$. While doing so, a vector $\boldsymbol{l}$ is also created in order to store the labels corresponding with each class of the dictionary columns.

The final dictionary $\boldsymbol{D}$ can be seen as a set of multiple subdictionaries $\boldsymbol{D} = [\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_k] \in \mathbb{R}^{m \times n}$, where $\boldsymbol{D}_i = [\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \ldots, \boldsymbol{v}_{i,n_i}] \in \mathbb{R}^{m \times n_i}$ corresponds to a subdictionary of a particular class, m corresponds to the number of pixels in the

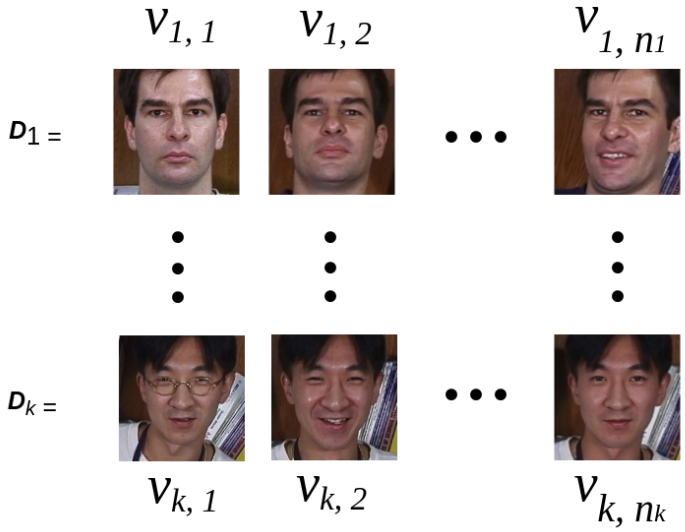


Figure 3.3: Illustration of training samples of faces associated with the different subdictionaries before the preprocessing steps.

---

downsampled image, n to the total number of samples, n_i to the number of samples for the class i, for $i \in [1, k]$, and $\boldsymbol{v}_{i,w}$ to the column number w associated with the class i.

Figure 3.3 shows an illustration of the faces associated with each column of the different subdictionaries.

3.3.2 Classification

After all the training images have been added to the dictionary $\boldsymbol{D}$ the classification of the test images can start. Firstly, one must ensure that the columns of the dictionary $\boldsymbol{D}$ are normalized to unit-norm. Then the sparse representation of the test image can be calculated by performing sparse coding. This can be done using our dictionary and by solving the minimization

$$\min_{\boldsymbol{x}} \ \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 . \tag{3.4}$$

Such minimization corresponds to a variation of (3.3) in which the reconstruction error $\|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_2^2$ is considered as $f(\boldsymbol{y}, \boldsymbol{D}\boldsymbol{x})$ and $\|x\|_1$ as $\Psi(\boldsymbol{x})$.

After this step, $\boldsymbol{x}$ will represent the sparse coefficients that relate the dictionary $\boldsymbol{D}$ to the test sample $\boldsymbol{y}$. Ideally, this will mean that most values in our coefficient vector $\boldsymbol{x}$ will

be zero and that the only non-zero values of the vector correspond to entries of a specific class (e.g. $\boldsymbol{x} = [0, ..., 0, \alpha_1, \alpha_2, ..., \alpha_{n_i}, 0, ..., 0]$).

Most of the time the obtained solution has non-zero coefficients associated with multiple classes. Therefore in order to perform the classification, the class that leads to a smaller residual error r_i has to be determined. For each class i, $i = 1..k$, the coefficients from $\boldsymbol{x}$ that correspond to that class are obtained. This reduced set of coefficients is represented by $\boldsymbol{x}_i$. Likewise, $\boldsymbol{D}_i$ represents a sub-dictionary in which the atoms from $\boldsymbol{D}_i$ only correspond to the class i. Using $\boldsymbol{x}_i$ and $\boldsymbol{D}_i$ the residual error r_i for each class i can be calculated using

$$r_i(\boldsymbol{y}) = \|\boldsymbol{y} - \boldsymbol{D}_i \boldsymbol{x}_i\|_2 . \tag{3.5}$$

The classification of the test sample $\boldsymbol{y}$ corresponds to the class associated with the minimum residual error r_i.

The SRC algorithm is shown in Algorithm 1 in pseudo-code format.

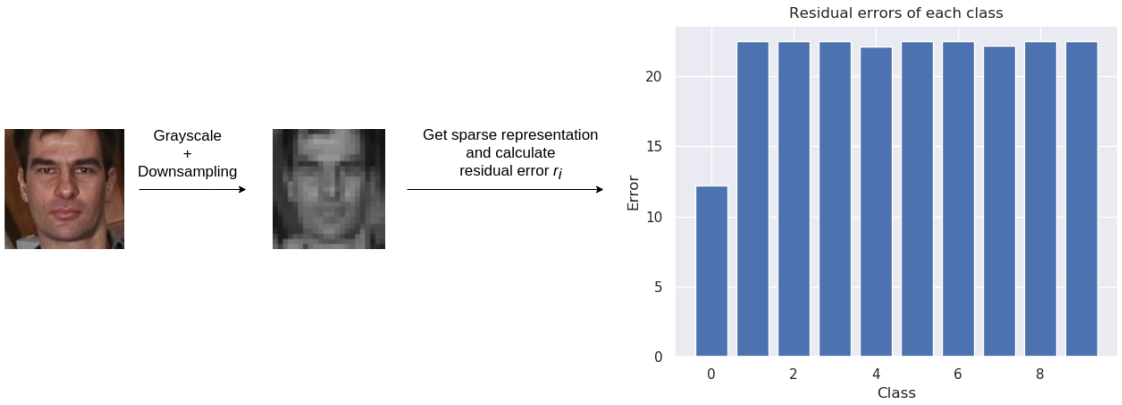


Figure 3.4: Bar plot showing the residual error r of each class for test sample from class 1.

3.3.3 Validation

While classification is the main goal of face recognition approaches, another important aspect that has to be dealt with, if one aims to develop a robust face recognition system, is validation. Validating a classification attempt ensures that the sample being classified is valid. In a face recognition scenario, the validation checks if the sample being recognized is in fact a face and if it belongs to one of the people used to build the dictionary. Otherwise, the sample is rejected as unknown. This is of an extreme importance since, it assures that, if the algorithm is not sure about a classification, it refuses to provide a wrong class.

Algorithm 1 Sparse Representation-based Classification (SRC)

1: $D \leftarrow$ Dictionary formed by training samples
2: $y \leftarrow$ *Test sample to be classified*
3: $\lambda \leftarrow$ Sparsity control parameter *top*:

4: Normalize the columns of $\boldsymbol{D}$ to have l^2-norm
5: Solve the following minimization problem:
$\min_{\boldsymbol{x}} \ \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1$
6: Compute the residual errors for each class i:
$r_i(\boldsymbol{y}) = \|\boldsymbol{y} - \boldsymbol{D}_i\boldsymbol{x}_i\|_2 \quad \text{for} \ \ i = 1, ..., k$
7: **Output:** $identity(\boldsymbol{y}) = \min_i r_i(\boldsymbol{y})$

In order to perform such validation, the residual error r_i mentioned previously can be used. If the minimum residual error is above a certain established threshold, then the sample is considered as invalid, and the classification outputs the class as unknown.

While using the residual error can provide reasonable results, it only considers each class individually. Since in sparse representation the coefficients for all the classes are computed jointly, considering the sparseness of the entire coefficient vector instead of the residuals of each individual class is a better approach that leads to more accurate validation [28].

A better alternative that utilizes sparsity in order to perform the validation is the use of the Sparsity Concentration Index (SCI) given by

$$SCI(\boldsymbol{x}) = \frac{k \cdot \max_i \|\boldsymbol{x}_i\|_1 / \|\boldsymbol{x}\|_1 - 1}{k - 1}. \tag{3.6}$$

This value will vary from 0 to 1, depending on the sparsity of the individual classes of the coefficient vector. Therefore, if we have a coefficient vector in which only the coefficients corresponding to class i are different from zero (e.g. $[0, 0, ..., 0, a_{i,1}, a_{i,2}, ..., a_{i,n_k}, 0, ..., 0, 0]$), then the SCI will be 1. Given the nature of sparse representation, as long as a valid sample is provided, the SCI should always approximate 1.

Figure 3.5 shows that, by testing a face image belonging to a person whose face images were used to form/learn the dictionary, in the resulting sparse coefficients it is possible to clearly point out the class of the test sample. The SCI of such sample is close to 1. However, while using a random image, the coefficients are less sparse and it is not possible to infer any conclusion. The SCI of such image will be near 0.

Figure 3.5: Plot of the sparse coefficients $\|\boldsymbol{x}\|$ of a face image and a random image.

3.3.4 Difficulties

While SRC can in theory provide good classification performance, there are some circumstances that prevent it from always being a good choice regarding face recognition:

- Dataset adequacy: The quality and variation of the samples used to form or learn the dictionary are crucial to have good performance in SRC. The more complete the dataset is regarding changes in illumination, expression and small pose changes the better the algorithm will perform. Consistency regarding the different classes is also recommended, since if one class has an extreme case of illumination and the other classes do not, there is a high probability that if a test sample of a different class has the same case of extreme illumination the algorithm will output a wrong classification.

- Pose variance: The poses of the face images should be aligned if one expects to have good results using SRC as a classification method. While there are face alignment

methods that rely on sparse representations [33], any face alignment method can be used in order to maximize classification accuracy. It should be noted that even using aligned images, the method usually fails if the original face image is too misaligned.

- Performance: since the more training samples used the better the recognition performance is, there is a motivation to use as many samples as possible during the training phase. But, unfortunately, the time it takes to perform the l_1 minimization increases exponentially with the number of samples used in building the dictionary [34]. This can lead to performance problems when dealing with systems with a large number of classes.

3.4 Dictionary Learning

While SRC provides good accuracy in some cases, there are situations that it cannot overcome, resulting in low classification accuracy or bad performance. In order to solve these problems, one can, instead of creating a dictionary directly from the training samples, use dictionary learning methods in order to learn a dictionary iteratively.

A learned dictionary, instead of one created directly from training samples, can model illumination and pose variance and, if needed, can be made more compact in order to provide better testing times.

There have been many dictionary learning algorithms proposed over the years. Despite these algorithms using the same principles for learning the dictionaries and for performing the classification (which is usually done in a similar manner to SRC), they differentiate themselves from one another by using different approaches in order to ensure that the learned dictionary has discriminative potential, so as to improve classification performance.

3.4.1 K-SVD

The K-SVD algorithm [24] is usually used as a benchmark dictionary learning algorithm for face recognition problems.

The K-SVD algorithm works by iterating between the sparse coding stage and the dictionary update stage, until either the algorithm has converged, meaning that there are no more significant updates between iterations, or until a fixed amount of iterations is reached.

Lets consider a matrix $\boldsymbol{Y}$ of size $m \times l$, where each column represents a different signal, a dictionary $\boldsymbol{D}$ of size $m \times n$ and a matrix of sparse coefficients $\boldsymbol{X}$ of size $n \times l$, where each column corresponds to the sparse coefficients of the each signal in $\boldsymbol{Y}$. Each row in $\boldsymbol{X}$ is related to each atom of the dictionary $\boldsymbol{D}$.

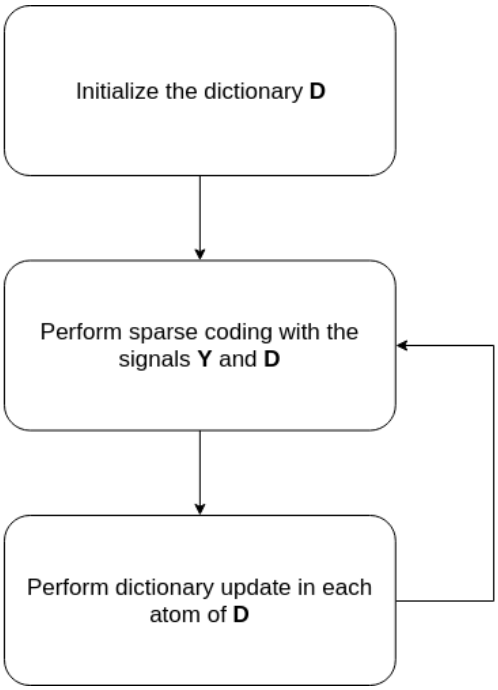


Figure 3.6: Illustration of the K-SVD dictionary learning algorithm.

The sparse coding stage is the same as seen previously. For each signal $\boldsymbol{y}_i$, for i in $1..l$, (3.4) or a different variant from (3.3) has to be solved, in order to find a sparse solution. When the solution has been found, the dictionary update stage can be executed.

In the dictionary update stage, each atom is optimized individually. Initially, an atom k is selected, and that atom alone will be updated. k can be selected sequentially or randomly.

For the selected atom k, any signals that the atom k did not contribute to are disregarded. This can be easily achieved by analyzing the k-th row $\boldsymbol{x}^k$ of the coefficient matrix $\boldsymbol{X}$. The atom k only contributes for any signal i if the value of $x_{k,i}$ is non-zero. The columns of $\boldsymbol{Y}$ and of $\boldsymbol{X}$ that do not use the atom k can also be disregarded. These reduced matrices will be represented by $\bar{\boldsymbol{Y}}$ and $\bar{\boldsymbol{X}}$ respectively.

The optimization problem is then simplified to only use the signals and coefficients that use the atom k i.e.

$$\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \implies \left\|\bar{\boldsymbol{Y}} - \boldsymbol{D}\bar{\boldsymbol{X}}\right\|_F^2. \tag{3.7}$$

Figures 3.7 and 3.8 show the creation of the matrices $\bar{\boldsymbol{Y}}$ and $\bar{\boldsymbol{X}}$. The signals and columns of the coefficient matrix that were not related to the atom k were removed in

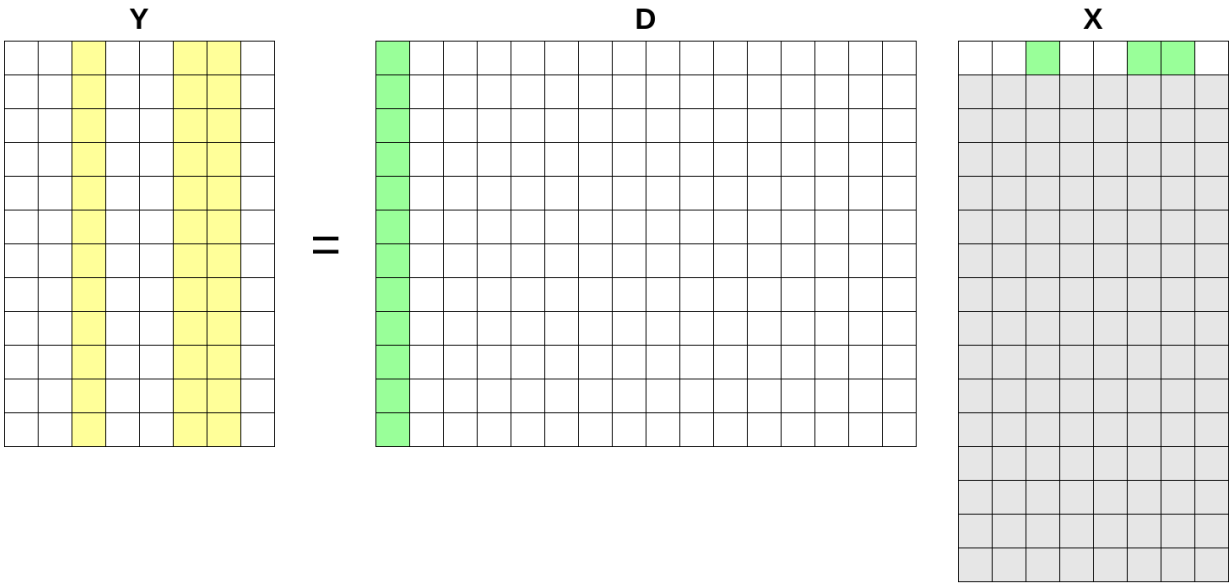


Figure 3.7: Selection of the atom $k = 1$. Signals in yellow represent the signals that the atom has contributed to. Coefficients in green represent the coefficients that use the atom.

---

order for the optimization to be performed. The removal of these additional coefficients and signals assures that the sparsity of this solution is maintained on the next iteration of the algorithm. The atoms in red represent the additional atoms, besides the atom k, that are used by the coefficients $\bar{\boldsymbol{X}}$ to represent the signals $\bar{\boldsymbol{Y}}$.

Another way of representing $\left\|\bar{\boldsymbol{Y}} - \boldsymbol{D}\bar{\boldsymbol{X}}\right\|_F^2$ is by expressing it as a linear combination of terms. This representation makes it easier to understand the optimization process of the atom, as can be seen in

$$\left\|\bar{\boldsymbol{Y}} - \boldsymbol{D}\bar{\boldsymbol{X}}\right\|_F^2 = \left\|\bar{\boldsymbol{Y}} - \sum_{d=1}^{n} \boldsymbol{d}_j \bar{\boldsymbol{x}}^j\right\|_F^2 \tag{3.8}$$

Where $\boldsymbol{d}_j$ denotes a column of the matrix $\boldsymbol{D}$ and $\boldsymbol{x}^j$ denotes a line of the matrix $\boldsymbol{X}$.

The atom k is then isolated from the rest of the atoms, according to

$$\left\|\bar{\boldsymbol{Y}} - \sum_{d=1}^{n} \boldsymbol{d}_j \bar{\boldsymbol{x}}^j\right\|_F^2 = \left\|(\bar{\boldsymbol{Y}} - \sum_{\substack{d=1 \\ d\neq k}}^{n} \boldsymbol{d}_j \bar{\boldsymbol{x}}^j) - \boldsymbol{d}_k \bar{\boldsymbol{x}}^k\right\|_F^2 \tag{3.9}$$

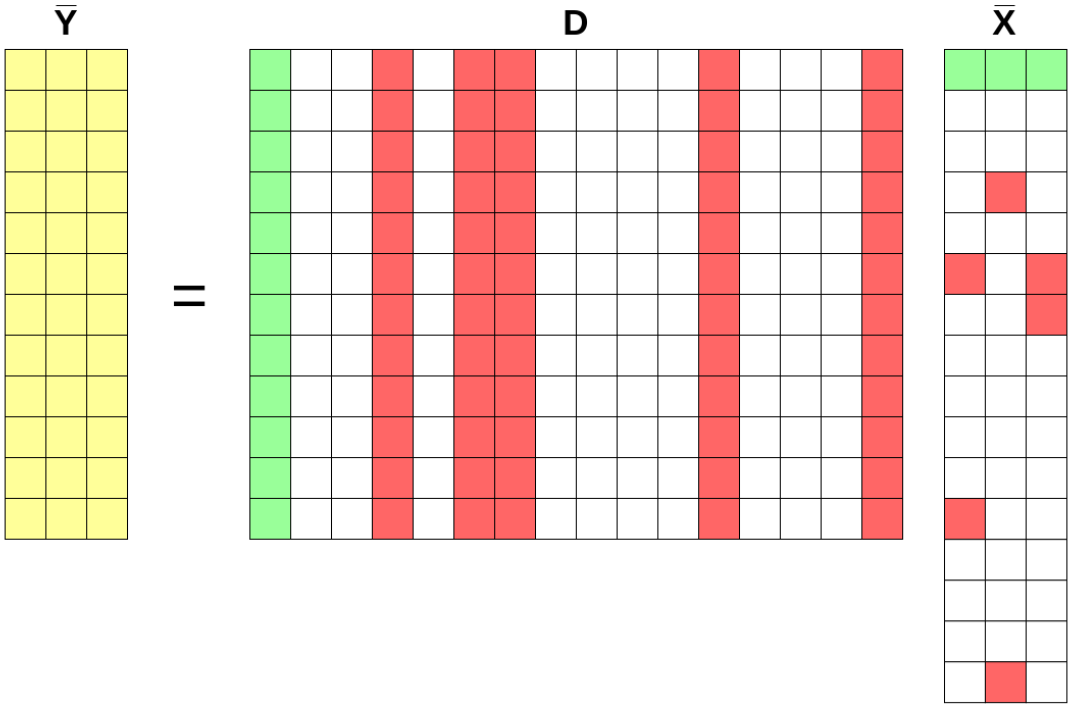


Figure 3.8: Selection of the atom $k = 1$. The redundant signals and coefficient columns were removed.

---

where $\boldsymbol{E} = \bar{\boldsymbol{Y}} - \sum_{\substack{d=1 \\ d \neq k}}^{n} \boldsymbol{d}_j \bar{\boldsymbol{x}}^j$ corresponds to the representation error when the atom k is not considered, leading to

$$\left\| (\bar{\boldsymbol{Y}} - \sum_{\substack{j=1 \\ j \neq k}}^{n} \boldsymbol{d}_j \bar{\boldsymbol{x}}^j) - \boldsymbol{d}_k \bar{\boldsymbol{x}}^k \right\|_F^2 = \left\| \boldsymbol{E} - \boldsymbol{d}_k \bar{\boldsymbol{x}}^k \right\|_F^2 \tag{3.10}$$

The matrix $\boldsymbol{E}$ is of size $n \times w$, n being the number of rows of the matrix $\boldsymbol{Y}$ and w the number of signals to which the atom k has contributed to. These are the same dimensions as in matrix $\bar{\boldsymbol{Y}}$. Considering m as the number of columns of $\boldsymbol{Y}$, since the sparse coding step produces a sparse matrix $\boldsymbol{X}$, w is much smaller than m.

The final objective in the dictionary update stage is to improve $\boldsymbol{d}_k$ and $\bar{\boldsymbol{x}}^k$ in order to solve

$$\min_{\boldsymbol{d}_k, \boldsymbol{x}^k} \left\| \boldsymbol{E} - \boldsymbol{d}_k \bar{\boldsymbol{x}}^k \right\|_F^2 . \tag{3.11}$$

The best way to do this is to use Singular Value Decomposition (SVD) [35] to decompose the matrix $\boldsymbol{E}$, i.e.

$$\boldsymbol{E} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{3.12}$$

which can also be written as a linear combination:

$$\boldsymbol{E} = \boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T + \boldsymbol{u}_2\sigma_2\boldsymbol{v}_2^T \cdots + \boldsymbol{u}_w\sigma_w\boldsymbol{v}_w^T \tag{3.13}$$

Since in an SVD the first terms of the decomposition are in decreasing order of importance, the first term $\boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T$ can be used to extract the new value of $\boldsymbol{d}_k$ and of $\bar{x}_K$, i.e.

$$\boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T \implies \boldsymbol{d}_k^* = u_1, \;\; \bar{\boldsymbol{x}}^{k*} = \sigma_1\boldsymbol{v}_1^T. \tag{3.14}$$

When the dictionary update stage has iterated through all the atoms in the dictionary, if the algorithms has not converged, the process repeats itself, starting in the sparse coding stage.

The K-SVD algorithm is an excellent algorithm for learning overcomplete dictionaries focused on reconstructing the samples, but it is not a good fit for classification problems, since it is focused solely in minimizing the reconstruction error of the samples and not in the discriminative power of the dictionary.

3.4.2 Face Recognition and Dictionary Learning

While K-SVD is an excellent algorithm for creating overcomplete dictionaries with excellent reconstruction potential, when dealing with classification, it is not the most adequate solution, since it doesn't focus on increasing the discriminatory ability of the dictionary. Many dictionary learning algorithms have been proposed in order to provide discriminatory ability to the dictionaries. This leads to improvements in the classification, leading to more accurate results. These dictionaries use the same principles of K-SVD in order to learn the dictionary, but they use different techniques in order to improve the discriminative ability of the learned dictionary.

There are many different approaches to dictionary learning regarding face recognition:

- Shared Dictionary Learning: K-SVD also belongs in this category. These algorithms learn a dictionary using the training samples from all classes, and expect the dictionary to have discriminatory capabilities. Another shared dictionary algorithm that improves upon K-SVD regarding the discriminatory capabilities of the dictionary is the LC-KSVD (Label Consistent K-SVD) [36]. This approach assigns labels to the

dictionary atoms and then uses a discriminative sparse coding error term by using said labels.

- Class Specific Dictionary Learning: Since the intra-class variation of a person can vary greatly with different poses and illuminations, learning a sub-dictionary independently for each class can help capture the main characteristics of the face images. The Fisher Discrimination Dictionary Algorithm (FDDL [37]) is an example of such a dictionary learning algorithm. Its goal is to learn independent dictionaries for each class, while at the same time ensuring that the dictionaries are discriminative by penalizing the objective function in the case that multiple dictionaries are similar.

- Commonality and Particularity Dictionary Learning: In face recognition problems, both the inter-class and intra-class variation of the face images can be quite large. The commonality and particularity dictionary learning algorithms can be used to cope with this variance. The algorithm proposed by Wang and Kong [38] suggests the creation of a class specific dictionary (particularity) for each category, in order to capture the discriminative features of that class and a common dictionary to all classes (commonality), in order to contribute to the representation of all classes instead of discrimination.

- Auxiliary Dictionary Learning Algorithm: Sometimes we can be facing a situation in which we have a low amount of samples to perform face recognition. In sparse coding this can be problematic, since the dictionaries will not have neither good reconstruction nor discriminate power, when dealing with small amount of labeled samples. The adaptive dictionary learning algorithm proposes that, in order to deal with such problems, two dictionaries are learned: A descriptive dictionary with the available training samples to perform classification, and an auxiliary dictionary from a generic training set to improve the classification and reconstruction performance in the cases that there is not an ideal number of samples available.

The different dictionary learning algorithms are appropriate for different situations. In a recent survey [39], the different algorithms were compared using multiple face recognition datasets and while there were cases in which different algorithms had an edge over one another at some point, in general the FDDL seems to consistently provide the best results.

3.5 Conclusion

In this chapter, the core concepts of sparse representation were explained. It was explained how sparse representation can be useful regarding imaging application and how one can perform sparse representation-based classification using the SRC algorithm. The strengths and weaknesses of the SRC algorithm were also presented. Multiple popular dictionary learning algorithms were exposed, and it was explained briefly why dictionary learning can be a better option regarding face recognition, when compared with SRC.

It should be noticed that, when dealing with a large number of samples per person, SRC can achieve better results than the dictionary learning algorithms exposed in this chapter [39]. Therefore, it should not be discarded completely on behalf of the dictionary learning algorithms. Instead, a better approach should be to evaluate the face recognition task, in order to choose the algorithm appropriately.

The next chapter will present an algorithm strongly based on SRC, that has proven to provide better results than the dictionary learning algorithms mentioned previously.

Chapter 4

Accumulative Local Sparse Representation

While SRC and the dictionary learning algorithms mentioned in Chapter 3 provide state of the art results regarding face recognition problems, recently it has been proposed in the literature new methods based on sparse representation that provide even better accuracy in the face recognition task. This chapter will introduce the Accumulative Local Sparse Representation (ALSR) [40] algorithm. This algorithm has been shown to provide higher accuracy in face recognition tasks, especially when a huge number of training samples is not available.

4.1 Introduction

ALSR is heavily based on the SRC algorithm. The main difference is that, instead of using the full face images to create the dictionary, it extracts patches from the images and creates the dictionary using them.

Regarding the classification of facial images, not all the facial attributes are of equal importance. Some parts of a face image can be more distinctive than others. This is why a patch based approach can provide better results than an approach that evaluates the whole image.

The use of patches aims to address some of the usual problems associated with face recognition. The use of patches instead of the entire image can solve, or at least diminish, some of these problems:

- The use of face accessories (e.g. sunglasses, scarfs). These elements can lead to wrong classifications. The use of a patch based approach means that the patches containing the occluding elements can be disregarded.

- Some facial attributes can be more discriminative than others. For example, patches containing facial attributes like the mouth and nose should be more relevant than patches containing the forehead.
- With face images with bad illumination, only the patches that are in well lit areas are considered.
- While using the traditional sparse representation based approaches like SRC, there was a need for highly aligned face images, in order to achieve high recognition performance. By using patches to perform the recognition task, this problem partially can be partially solved, by not considering only the patches from one fixed position, but also the patches from the neighbourhood of that position.
- While in SRC we only considered the similarity of whole images to one another, using a patch based approach we can search for similar face parts in all the dictionary, instead of just looking for similar images. E.g. a test image that cannot be correctly classified using the training images, can perhaps be correctly identified by using patches from multiple images.

Figure 4.1: Example of important and unimportant patches in a face image.

Figure 4.1 shows a face image and some highlighted patches. The patch highlighted in red is not important for the classification, since it is on top of the sunglasses that are occluding the facial attributes. The patches in green, however, since they are on top of important facial attributes, namely the nose and the lips, have a higher importance in the classification.

4.2 Algorithm

The ALSR algorithm consists of two stages. The learning stage and the testing stage. During the learning stage, the patches are extracted from each training image and added

to the global dictionary $\boldsymbol{D}$ as atoms. The coordinates and label of the patches are also stored alongside the atoms for later use. The testing stage will output the classification of a testing image by extracting patches from it and analyzing each of them individually. The final classification is relatively similar to a voting system, where each patch from the testing image votes for a class. In the end, the class with the most contributions is considered the class of the image.

4.2.1 Learning Stage

In the learning stage, the objective is to build the global dictionary $\boldsymbol{D}$. Each column of this dictionary corresponds to a patch from one of the training images. Assuming that there are n_c training images of each class, that there are k classes, that n_p patches are extracted from each image and that each patch has $w \times w$ pixels, the global dictionary $\boldsymbol{D}$ will have $n_c \times k \times n_p$ columns and $w \times w$ rows. Alongside the dictionary there are also two arrays $\boldsymbol{c}$ and $\boldsymbol{l}$, of length $n_c * k * n_p$, that correspond, respectively, to the coordinates and labels of the patches associated with each column of the dictionary.

For each training image, n_p patches are extracted. The patches are extracted in a grid like manner. The horizontal and vertical number of patches are n_{p_x} and n_{p_y} respectively. For each patch ρ extracted from the image, its pixels are stacked vertically so it becomes a column. This column is then added to the dictionary $\boldsymbol{D}$. The coordinates of the patch, and the label corresponding to the image it belongs to, are stored in the vectors $\boldsymbol{c}$ and $\boldsymbol{l}$.

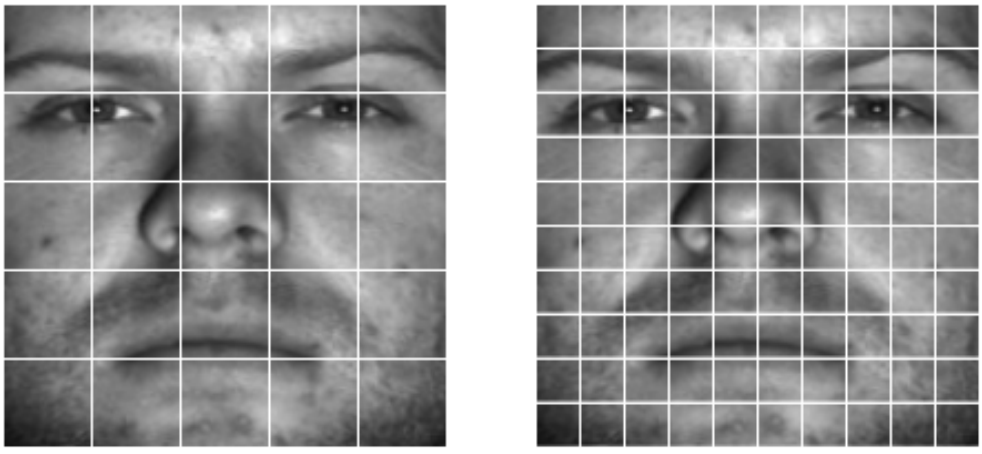


Figure 4.2: Different number of patches in the same image. The image on the left has been split into $n_p = 25$ patches ($n_{p_x} = 5, n_{p_y} = 5$). The image on the left has been split into $n_p = 100$ patches ($n_{p_x} = 10, n_{p_y} = 10$)

Figure 4.3 shows an illustration of the process mentioned above.

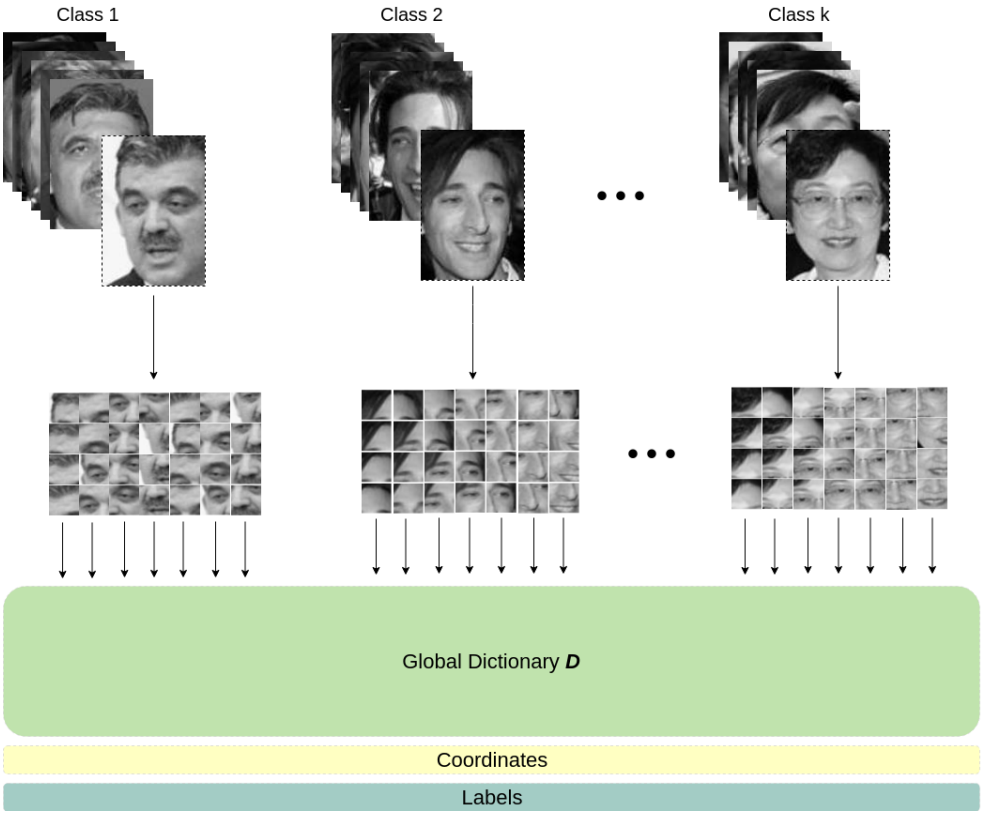


Figure 4.3: Adding the extracted patches to the global dictionary $\boldsymbol{D}$

4.2.2 Testing Stage

In the testing stage, given a testing image $\boldsymbol{I}$, the objective is to output the class of that image. The image $\boldsymbol{I}$ is split into n_p patches, in the same manner as the training images were. This assures that the patches cover the same areas from the image and are of the same size (which they must be in order to perform sparse coding). Then the following steps are taken, which lead to the classification:

- Building the Local Dictionary
- Sparse Coding
- Analysis of contributions

Figure 4.5 shows an illustration of the processing of a patch, namely the building of the local dictionary and the sparse coding steps. The test image is split into patches, then, for each patch, a smaller local dictionary $\boldsymbol{D}^*$ is built. Using this dictionary, the sparse representation $\boldsymbol{x}$ of the patch can be obtained in the sparse coding step. Having $\boldsymbol{x}$, the contribution vector $\boldsymbol{s}$ can be created. This process is repeated for all the patches. Once

all the patches have been processed, the analysis of contributions step can be performed. This step will output the predicted class of the test image.

Building the Local Dictionary

In SRC, the dictionary is usually of a small size, due to it being formed by downsampled images. In ALSR, this is not the case. While in SRC the images could be downsampled to relatively low resolutions, in ALSR, in order for the patches to retain enough information for them to be discriminative, the image must preserve a large size. The number of atoms of the dictionary is also much less in SRC than in ALSR, since in SRC each atom corresponds to an image and in ALSR each atom corresponds to a patch. This, coupled with the fact that the patches can partially overlap each other, leads to a significantly larger dictionary than one used in SRC. Performing sparse coding on such a large dictionary would be too resource intensive, therefore taking far too much time. Hence, the use of a smaller dictionary is required to be able to perform the classification efficiently. This can be achieved by using a subset of $\boldsymbol{D}$. This sub-dictionary is called the local dictionary and it is represented by $\boldsymbol{D}^*$.

$\boldsymbol{D}^*$ can be used instead of $\boldsymbol{D}$ due to two factors:

- Only the patches from the same neighborhood from the patch being tested are relevant, since all the facial attributes are expected to remain approximately in the same area.
- Only patches that are similar to the patch being tested should be considered. This allows for patches that are significantly different and that would not bring any benefit by remaining in the local dictionary to be discarded.

Considering a patch $\boldsymbol{\rho}$ being tested, in order to build the local dictionary $\boldsymbol{D}^*$ two steps need to be performed. The patches from the global dictionary $\boldsymbol{D}$ that are in the same neighborhood as the patch $\boldsymbol{\rho}$ are copied to $\boldsymbol{D}^*$. In order to select these patches the coordinate information stored in the training stage is used. Only patches that are within a distance d to the patch $\boldsymbol{\rho}$ are selected for $\boldsymbol{D}^*$. Then the cosine similarity of those patches to $\boldsymbol{\rho}$ is computed, and any patch whose similarity is below a threshold θ_{sim} is discarded.

Figure 4.4 shows a patch ρ is highlighted in green and its neighborhood area is highlighted in yellow. Only patches from $\boldsymbol{D}$ that are within that area and are similar to $\boldsymbol{\rho}$ are used for $\boldsymbol{D}^*$.

Sparse Coding

Having created the local dictionary $\boldsymbol{D}^*$, the sparse coding of ρ can now be performed. This step is performed in the same way as in SRC, except that now, instead of the whole

Figure 4.4: Neighborhood of a patch

image, ρ will serve as the $\boldsymbol{y}$ and, instead of the full dictionary $\boldsymbol{D}$, the local dictionary $\boldsymbol{D}^*$ is used, according to

$$\min_{\boldsymbol{x}} \ \|\boldsymbol{\rho} - \boldsymbol{D}^*\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 . \tag{4.1}$$

Having performed the sparse coding of $\boldsymbol{\rho}$ leaves us with the coefficient vector $\boldsymbol{x}$. By using the labels of the patches from $\boldsymbol{D}^*$, it is possible to obtain subsets of $\boldsymbol{x}$ associated with each class i, for $i = 1..k$. A subset from $\boldsymbol{x}$ only containing the coefficients associated with the class i is represented by $\boldsymbol{x}_i$.

The contributions of the multiple classes are stored in a vector $\boldsymbol{s}$ of length k, where each element, s_i, for $i = 1..k$, corresponds to the absolute sum of the contributions of the patches from the class i. This corresponds to the l_1-norm of $\boldsymbol{x}_i$, i.e.

$$\boldsymbol{s} = [s_1, ..., s_k] \tag{4.2}$$

$$s_i = \|\boldsymbol{x}_i\|_1 . \tag{4.3}$$

Figure 4.5 shows the process until this point. Initially, the patches are extracted from the testing image on the left. Then, for each patch, a local dictionary is created by selecting from the global dictionary only the patches from the same area as the patch being tested. From these patches, any patches whose similarity to the patch being tested is below a similarity threshold are discarded. Then, by using the dictionary $\boldsymbol{D}^*$, the sparse coefficients $\boldsymbol{x}$ of the patch are calculated. The contribution vector $\boldsymbol{s}$ can be created using $\boldsymbol{x}$.

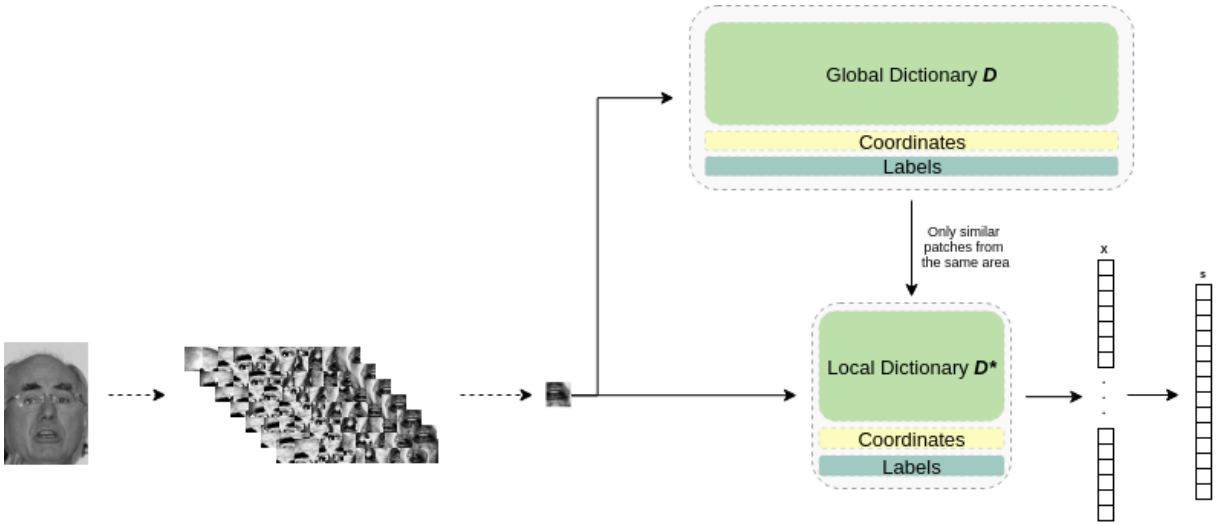


Figure 4.5: Processing a patch from a testing image.

In order to decide if the patch ρ is to be used in the classification process, a variation of the SCI mentioned before in (3.6) can be used. This will evaluate the sparsity of the contributions vector $\boldsymbol{s}$. The value of the SCI will be between 0 and 1. Values close to 1 mean that the vector $\boldsymbol{s}$ is sparse. If the patch is discriminative, its SCI value is expected to be closer to 1,

$$SCI(\rho) = \frac{k \max_i s_i / \|\boldsymbol{s}\|_1 - 1}{k - 1}. \tag{4.4}$$

The patch will be considered for classification, if its SCI is greater or equal than a threshold θ_{SCI}.

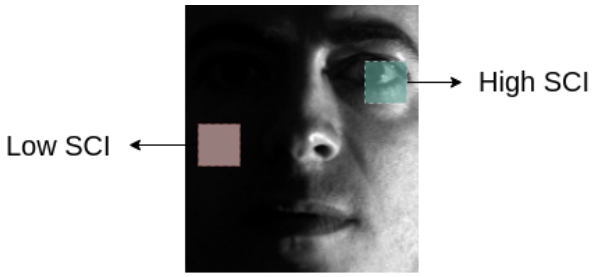


Figure 4.6: Representation of a high SCI patch (green) and a low SCI patch (red).

Figure 4.6 shows an image with two highlighted patches. The patch in green corresponds to a well lighted area with discriminative potential (since it is positioned over the eye). These two factors contribute to the high SCI of the patch. The patch in red, on the

contrary, is located in a poorly lighted area with no clear face attributes. Therefore, this patch has a low SCI.

Analysis of contributions

The next step is to analyze the contributions of all the patches of the image, in order to perform the classification. For each coefficient vector $\boldsymbol{s}$ associated with each patch $\boldsymbol{\rho}$, if that patch is considered relevant given its SCI ($SCI(\boldsymbol{\rho}) \geq \theta_{SCI}$), then its contributions are normalized by its maximum value, i.e.

$$\bar{\boldsymbol{s}} = \boldsymbol{s} / \max_i s_i. \tag{4.5}$$

After the normalization of the contribution vectors $\bar{\boldsymbol{s}}$, any values in those vectors that are under a threshold θ_c are replaced by 0. This removes the noise contributions from the contribution vectors.

The contributions of all the patches $\boldsymbol{\rho}_i$ of the test image $\boldsymbol{I}$ are added. This sum is a vector of length k, that is represented by $\boldsymbol{z} = [z_1, ..., z_k]$, where each element corresponds to the contributions of all the patches for each possible class,

$$\boldsymbol{z}(I) = \sum_{i=0}^{p} \bar{\boldsymbol{s}}_i. \tag{4.6}$$

The test image $\boldsymbol{I}$ is classified as being of the class i, where z_i is maximal.

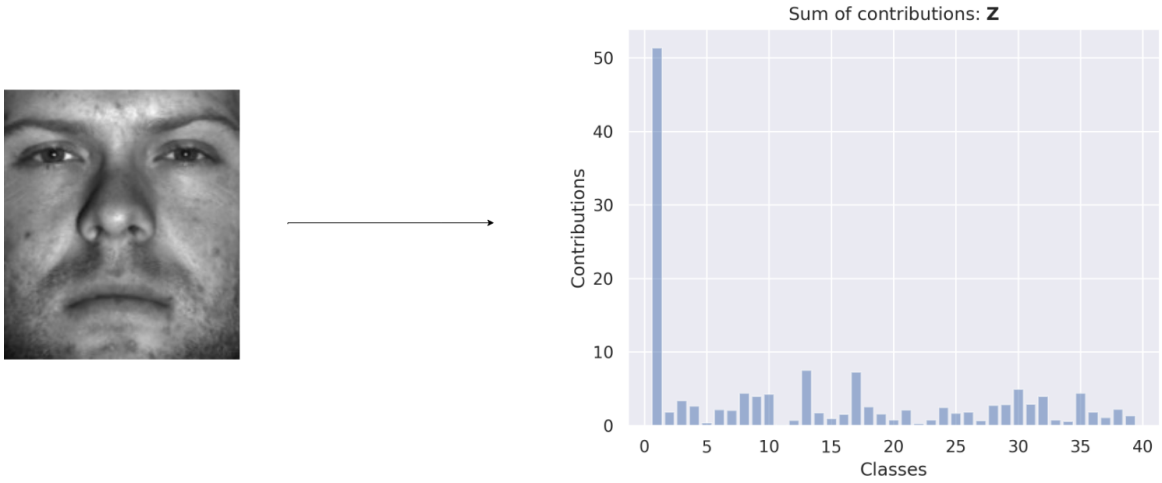


Figure 4.7: Plot representing the vector $\boldsymbol{z}$ associated with a face image.

Algorithms 2 and 3 show a pseudo-code implementation of the ALSR training and testing stages.

Algorithm 2 Adaptive Local Sparse Representation (ALSR) - Training Stage

input: $\boldsymbol{T} \leftarrow$ *array of training images*
output: $\boldsymbol{D} \leftarrow$ *global dictionary*

$\boldsymbol{D} \leftarrow$ new empty dictionary matrix
$\boldsymbol{l} \leftarrow$ new empty class label array
$\boldsymbol{c} \leftarrow$ new empty coordinate array

for each image $\boldsymbol{I}$ in $\boldsymbol{T}$ **do**
    for each patch $\boldsymbol{\rho}$ in $\boldsymbol{I}$ **do**
        $\boldsymbol{h} \leftarrow$ stacked columns of $\boldsymbol{\rho}$
        add $\boldsymbol{h}$ to $\boldsymbol{D}$ as column
        add $class(\boldsymbol{I})$ to $\boldsymbol{l}$
        add $coordinates(\boldsymbol{\rho})$ to $\boldsymbol{c}$
return $\boldsymbol{D}$

Algorithm 3 Adaptive Local Sparse Representation (ALSR) - Testing Stage

input:
$\boldsymbol{I} \leftarrow$ *test image*
$\boldsymbol{D} \leftarrow$ *global dictionary*
$d \leftarrow$ *distance between coordinates*
$np \leftarrow$ *number of patches in global dictionary*
output:
class of test image $\boldsymbol{I}$

$\boldsymbol{z} \leftarrow$ empty contributions array of length k
for each patch $\boldsymbol{\rho}$ in $\boldsymbol{I}$ **do**
    $\boldsymbol{D}^* \leftarrow$ new empty local dictionary matrix
    $\boldsymbol{l}^* \leftarrow$ new empty local labels array
    for i in $0 \ldots np - 1$ **do**
        if $dist(c[i], coordinates(\boldsymbol{\rho})) < d$ **then**
            add $\boldsymbol{D}[i]$ to $\boldsymbol{D}^*$ as column
            add $\boldsymbol{l}[i]$ to $\boldsymbol{l}^*$
    $\boldsymbol{D}^* \leftarrow$ get only similar patches to $\boldsymbol{\rho}$ from $\boldsymbol{D}^*$
    $\boldsymbol{x} \leftarrow sparse\_coding(\boldsymbol{\rho}, \boldsymbol{D}^*)$
    $\boldsymbol{s} = contributions(\boldsymbol{x})$
    if $SCI(\boldsymbol{s}) < \theta_{SCI}$ **then**
        Skip to the next patch
    $\bar{\boldsymbol{s}} = \boldsymbol{s}/max(\boldsymbol{s})$
    for $\bar{s_i}$ in $\bar{\boldsymbol{s}}$ **do**
        if $\bar{s_i} < \theta_c$ **then**
            $\bar{s_i} = 0$
    $\boldsymbol{z}$ += $\bar{\boldsymbol{s}}$
return i where z_i is maximal

4.3 Difficulties

The improvement in accuracy gained by using ALSR against the previously mentioned methods, like SRC or one of the dictionary learning algorithms, comes at the cost of the performance of the algorithm.

This performance penalty is due to a number of different factors:

- The size of the dictionary is much larger in ALSR. In SRC and in the dictionary learning algorithms discussed previously, the images could be downsampled without compromising accuracy. While in ALSR the images can also be downsampled if their size is unreasonable large, they cannot be reduced to the dimensions as low as with SRC or the dictionary learning algorithms. Thus, the patches used as columns in the ALSR can have approximately the same resolution than the entire image in SRC. This coupled with the fact that each training image in ALSR occupies n_p columns of the dictionary, while in SRC each image corresponds to only one column, makes the sparse coding step very time consuming.
- The construction of the local dictionary, which is done for each patch ρ of an image, has to be done p times for each testing image and is comprised of two resource intensive steps:
  1. The discovery of all the patches in $\boldsymbol{D}$ whose coordinates are within the distance d from the current patch being tested.
  2. The discovery of the most similar patches to the patch being tested.
- The sparse coding step, while generally faster, since it is dealing with small patches and a relatively small local dictionary $\boldsymbol{D}^*$, has to be performed p times for each testing image.

All of the above factors coupled together lead to a very resource intensive algorithm, with a testing time per image in the order of several seconds. While this is reasonable for some applications, for most it is not.

4.3.1 Improving performance

The original paper of ALSR [40] only provides a high level description of the algorithm. Therefore, it does not delve into details regarding its implementation. Despite it being a resource intensive algorithm, there are some approaches that can be taken in order to reduce testing time of the algorithm:

- The first step of the building of the local dictionaries can be done in the training stage. Since the patches from the training and testing images share the same pairs of coordinates, it is possible to create the sub-dictionary only containing the patches within a certain distance d for each different pair.

- The use of a highly optimized similarity search library (e.g. FAISS [41]) can speed up the search for the similar patches when dealing with very large datasets, especially if the library utilizes GPU resources. If needed, these libraries also offer methods for approximate search that are much faster than the regular methods, while sacrificing almost no precision. This step combined with the previous decreases substantially the time it takes to build the local dictionary.

Even by performing the steps above, the biggest bottleneck of the testing stage is by far the sparse coding step. It is crucial to use an highly optimized sparse coding library (e.g. SPAMS [42]) in order to achieve acceptable testing times per face image.

It should also be noted that these sub-dictionaries created in the steps above share a large amount of the same patches with each other. Therefore, the amount of system memory used by this approach can be quite large when dealing with larger datasets, since there is a lot of duplicate information. Another possibility, albeit slower, is to store only the indexes of the patches and then fetching them from the global dictionary during the testing stage.

4.4 Variations

There are some possible variations of the algorithm that can be used if better testing times are required. These approached are also patch based, but they do not reach the same levels of accuracy than ALSR:

- Using a fixed amount of patches in the subdictionary - Instead of using all the patches that are similar to a patch being tested, only the m most similar patches are selected. This ensures that the time taken per test image is approximately constant and the number of patches can be used as a trade-off between accuracy and speed.

- Using random patches - This idea was explored by the same authors of ALSR in another algorithm called Adaptive Sparse Representation (ASR) [43]. The algorithm extracts random patches from the training images and then, by using k-means clustering, builds small local dictionaries in order to perform the sparse coding and the classification.

4.5 Conclusion

In this chapter, a new state of the art algorithm was introduced. Both the advantages and the disadvantages of such an algorithm were explained, as were solutions to address such difficulties.

ALSR can be considered as an excellent algorithm for performing face recognition, since it tackles many of the downfalls of such a task. It should be noted that it is a resource intensive algorithm, and if time is a concern, especially when dealing with bigger datasets, other approaches should be considered. Otherwise ALSR is the algorithm to beat regarding face recognition via sparse representation.

Chapter 5

Experimental Results

In this chapter, the experimental results of this dissertation will be presented. The chapter will focus on the SRC and ALRS algorithms. The different parameters of each algorithm will be tested and their influence on the results will be explained. Other results from other algorithms will be mentioned for reference, but these algorithms will not be tested in depth. The chapter will also compare the different algorithms regarding multiple factors, like performance and accuracy. The datasets used will be the three introduced in Chapter 2.

5.1 SRC

There are multiple factors that influence the accuracy obtained by using the SRC algorithm to perform the recognition of face images. The algorithm itself only has one parameter. The parameter λ, previously seen in (3.4), regulates how sparse the set of coefficients $\boldsymbol{x}$ should be. The influence of this parameter on the accuracy of the algorithm will be tested.

Besides the value of λ, different conditions that influence the accuracy and performance of SRC will be studied. Those conditions are:

- Amount of training samples
- Pre-alignment of the face images
- Dimensionality of the samples

In order to better understand the performance constraints of the sparse coding step, varying dictionary sizes will also be tested.

All three datasets will be used during the tests, with the exception of the image alignment test, since the YALE extended dataset and the deep funneled LFW are already aligned.

In order to measure the accuracy of the algorithm, the Monte Carlo cross validation [44] method is used. The samples are shuffled randomly and the testing and training sets are formed after. The algorithm is then run and the accuracy measured. This process is repeated five times,and the final accuracy is the average accuracy of the five iterations.

It should be noted that the datasets have a different number of classes. The bigger the number of classes, the more difficult it is to perform a correct classification. Therefore, since the LFW dataset has many more classes than the rest of the datasets (143 against 39 for the YALE dataset and 50 for the GT dataset), it is natural that it cannot achieve the same levels of accuracy.

5.1.1 Lambda

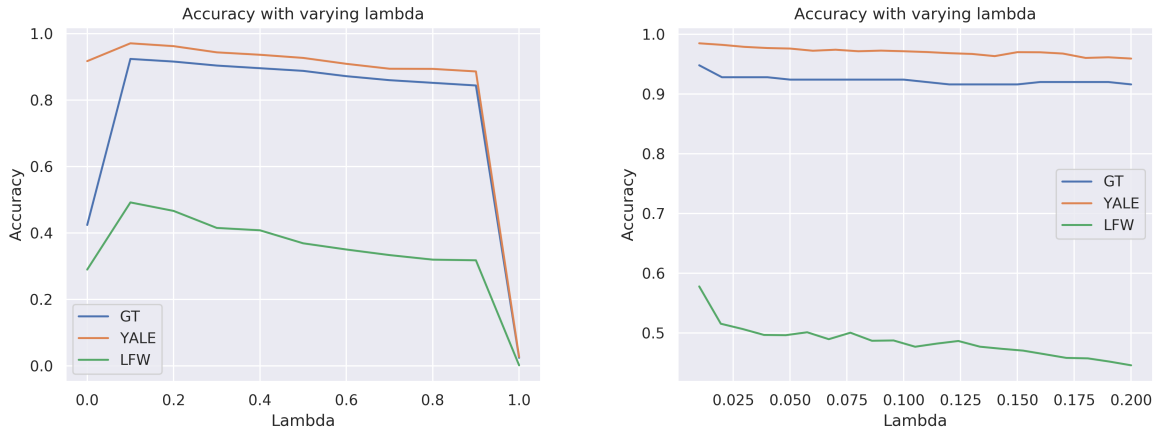
As discussed previously in Chapter 3, the parameter lambda (λ), present in (3.4) regulates the trade-off between the sparsity of the solution and the reconstruction error. High values of lambda lead to a more sparse solution, but with more reconstruction error, while low values of lambda lead to less reconstruction error, but are prone to less sparsity.

It should be noted that the images from the GT dataset were pre-aligned, since in this test we only care about the influence of the lambda parameter. All the images have been downsampled to a smaller, yet generous size (the size of the downsampled images is (25, 30)) in order to reduce test times without compromising on accuracy [45].

For the GT dataset, ten samples were used for training, while the remaining five were used for testing. Regarding the YALE dataset forty samples were used for training and the remaining twenty for testing. For the LFW dataset, ten images were used for training, and the remaining were used for testing. The number of testing samples per class in the LFW dataset varies depending on the class.

The plots of Figure 5.1 show how the algorithm performs for varying values of λ. Figure 5.1a shows that the accuracy decreases with increasing values of λ. This means that focusing too much on the sparsity of the solution can lead to less accurate results. It's also interesting to note that when $\lambda = 1$ the accuracy is close to zero. This occurs because by setting the value of λ so high, the algorithm sets all the coefficients to zero, since this minimizes (3.4).

Another interesting observation is when $\lambda = 0$. This means that the algorithm is just trying to reconstruct the testing samples as faithfully as it can, without any concern for the sparsity of the solution. As one would expect, since our classification relies on the sparsity of the solution, this leads to inferior results regarding accuracy.



(a) Lambda (λ) $\in$ [0.0, 1.0]

(b) Lambda (λ) $\in$ [0.01, 0.2]

Figure 5.1: Plots showing the accuracy of the classification when varying the parameter lambda.

The plots shown demonstrate that the best results are achieved for small values of λ, but that the regularization parameter is still essential in order to improve the accuracy of the algorithm. Figure 5.2 shows that the optimal value of λ for all the datasets is 0.002, but any value of lambda around this value yields approximately the same results.

5.1.2 Number of training samples

Utilizing the optimal value of lambda ($\lambda = 0.002$) discovered previously, in this section the number of training samples will vary in order to see the impact in the accuracy of the algorithm.

The plots shown in the the Figures 5.3a, 5.3b and 5.4 show how the number of training samples used relates to the accuracy of the algorithm. Naturally, by increasing the number of training samples the accuracy increases for all the datasets.

It is interesting to note that the increase in accuracy behaves like a logarithmic function, since the increase in accuracy observed using a low amount of training samples is significantly higher than the increase observed when using a high amount of training samples. This becomes especially obvious when analyzing a dataset like the YALE dataset, where it can be seen that there is almost no benefit in using more than 15 training samples per class, since the increase in accuracy from 15 onward is minimal.

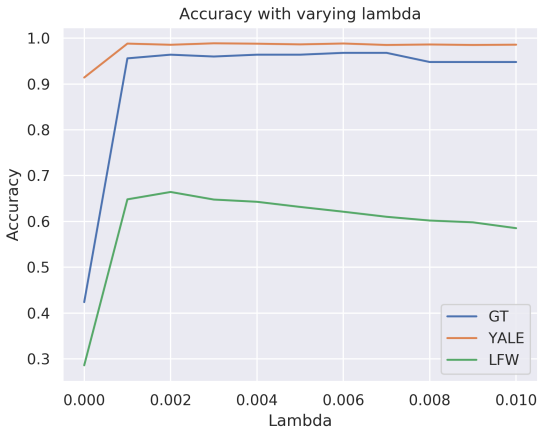
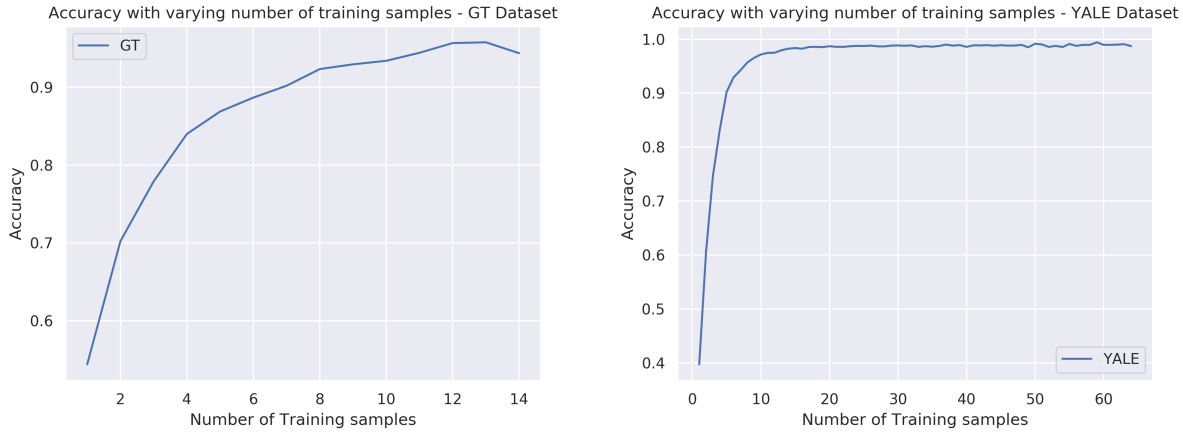


Figure 5.2: Lambda (λ) $\in$ [0.0, 0.01]



(a) Varying number of training samples GT Dataset

(b) Varying number of training samples YALE Dataset

5.1.3 Face Alignment

The SRC algorithm is by nature very susceptible to misalignment. In the case of downsampling as a dimensionality reduction method, the algorithm compares the downsampled images pixel by pixel. If the images are misaligned, this leads to comparisons between different facial attributes, which in turn leads to misclassifications.

In Table 5.1 the influence of the face alignment can be observed. Only the GT and LFW datasets are used, since the images from the YALE dataset are all aligned.

As expected, the drop in accuracy by utilizing the unaligned, instead of the aligned face images, is significant. This results in about half of the classification accuracy in the GT

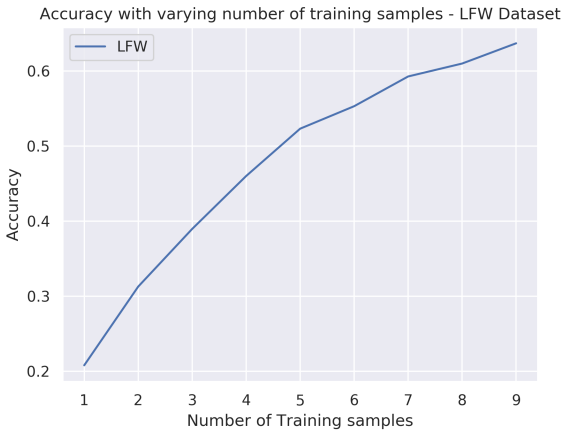


Figure 5.4: Varying number of training samples LFW Dataset

	Aligned	Unaligned
GT	95.61%	47.91%
LFW	65.61%	20.84%

Table 5.1: Accuracy for the aligned and unaligned LFW and GT datasets.

dataset and about a third of the accuracy in the LFW. This is due to the poses of the faces in the LFW being more misaligned than those in the GT dataset.

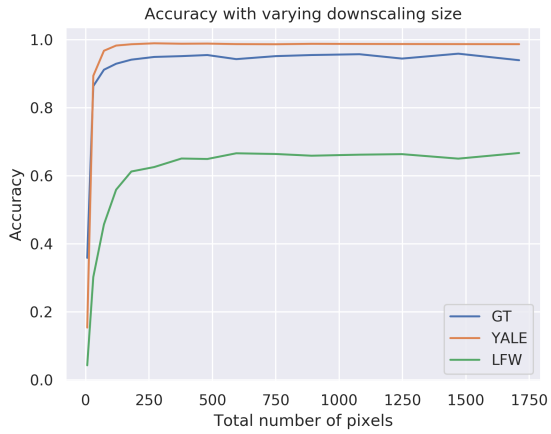
In conclusion, performing face alignment before the classification when utilizing the SRC algorithm is a must in order to achieve good classification accuracy.

5.1.4 Dimensionality Reduction

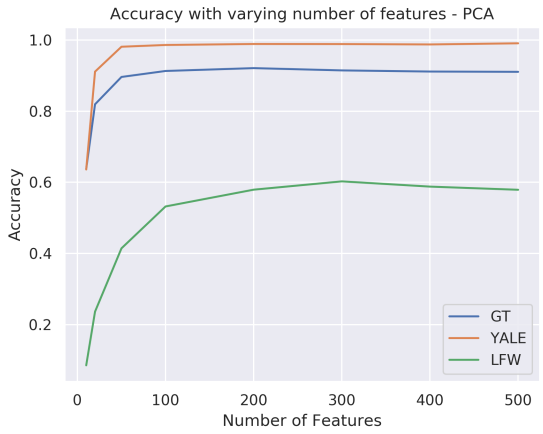
Dimensionality reduction allows the SRC algorithm to run much faster than by using the original images directly. As long as the reduced images are still reasonably large, the drop in classification accuracy should be minimal.

Three dimensionality reduction techniques will be used: PCA, Random Projection and Downscaling.

Figures 5.5a, 5.5b and 5.6 illustrate the variation in accuracy depending on the number of features outputted by the three dimensional reduction methods mentioned above. Overall, all of the methods show the same behaviour. The accuracy increases at an accelerated rate when dealing with a small number of features, and the increase starts to stagnate once a higher number of features has been reached. By observing the LFW dataset accuracy, it is possible to conclude that this number is about 500 for the downscaling method, 300 for



(a) Accuracy given varying sizes - Downscaling



(b) Accuracy given varying number of samples - PCA

the PCA and more than 500 for the random projection. When using samples with high dimensionality, all the methods can achieve approximately the same accuracy.

It should be noted that, while using PCA as a dimensionality reduction method, the maximum number of dimensions that can be outputted is the number of training samples. This happens since, in the SVD performed during the PCA algorithm, the maximum number of singular values is the number of training samples.

	Downscaling	PCA	Random Projection
100	1.3563s	2.7081s	1.8197s
500	1.4761s	4.5964s	1.9845s
1000	1.5017s	8.3496s	2.1544s

Table 5.2: Times for the dimensionality reduction methods - LFW dataset

The time it took for each method can be seen in Table 5.2. While the time always increases in accordance with the number of features, the downscaling and random projection methods only have a slight increase for any number of features. Using PCA takes much more time, especially if the number of features is high.

The downscaling was performed using OpenCV [46]. The PCA and random projection were performed using scikit-learn [47].

The findings presented in this section are consistent with the literature. If the dimensionality of the samples remains reasonably high, the dimensionality reduction method becomes redundant in terms of accuracy. Still, for the datasets used in these tests, the

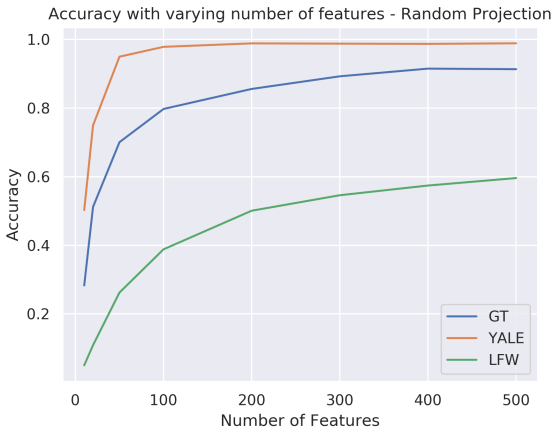


Figure 5.6: Accuracy given varying number of samples - Random Projection

downscaling method still provides slightly more accuracy across the board. It is also the fastest dimensionality reduction method from all the tested, and it does not suffer a high penalty in performance with the increase of the number of features, as can be seen in the PCA method.

5.1.5 Dictionary size and performance

This section aims to analyze the performance of the sparse coding step with dictionaries of varying dimensions. Since this step is the most resource intensive step in the classification, it is interesting to see how the dictionary size relates to the time it takes to sparse code a signal.

In order to preform these tests, a random signal and dictionary are generated with varying sizes.

Firstly, a signal with length 100, a dictionary with 100 rows and a variable number of atoms are tested. In SRC, this would simulate the sparse coding of a test sample with a dictionary with a variable amount of training samples.

Figure 5.7 shows the time it takes to perform the sparse coding step with a dictionary with a variable number of atoms. The time becomes irregular as the size of the dictionary increases. Since the signal and dictionary are created randomly, some sparse coding steps may differ slightly regarding the number of iterations it takes to reach the sparse representation. Nevertheless, it is clear that, an increase in the number of columns of the dictionary results in a linear increase of the time it takes to compute the sparse representation of a signal.

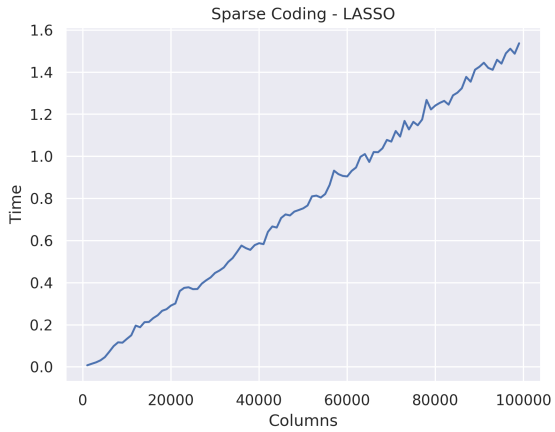


Figure 5.7: Time it takes for the sparse coding step with a variable number of columns.

Now, a signal of variable length and a dictionary of 100 columns with a variable number of rows are tested. In SRC, this scenario would correspond to a fixed amount of training samples, but with different dimensions.

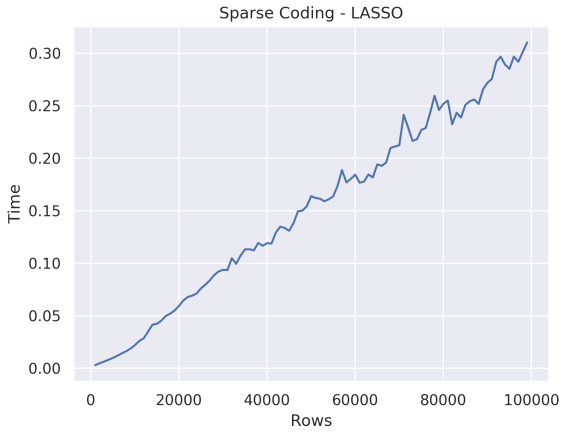


Figure 5.8: Time it takes for the sparse coding step with a variable number of rows.

Figure 5.8 shows the time it takes to perform the sparse coding step, when varying the number of rows of the dictionary, as well as the length of the signal to be represented. Interestingly, the pattern seems to be the same as seen before. By increasing the number of rows of the dictionary and the signal length, a linear increase in the time can be observed. Its also worth mentioning, that in spite of the dimensionality growth of the dictionary being the same as in the previous case, the time increases at a much slower rate.

In a face recognition scenario, the first test shows that the amount of time taken to perform sparse coding of a face image increases linearly with the amount of training samples

in the dictionary. The second test shows that the same can be said for if there's an increase of the dimensions of the samples, but this performance hit is not as pronounced as the one taken from increasing the amount of training samples.

5.2 ALSR

In this section, the different conditions that influence the accuracy and performance of ALSR will be studied. This algorithm features a high number of parameters:

- p_x - number of horizontal patches
- p_y - number of vertical patches
- w - size of the patches ($w \times w$)
- d - distance of the patches from the same neighborhood
- θ_{sim} - similarity threshold
- θ_{sci} - SCI (Sparsity Concentration Index) threshold
- θ_c - noise threshold
- Sparse coding algorithm (lars/omp)

Therefore, the analysis performed on this algorithm will not focus on the parameters, but instead focus on these particular conditions:

- Number of training samples
- Image Alignment
- Patch Overlap

This algorithm will be compared mostly with the SRC. Since the number of parameters is substantial, and given that the algorithm takes a long time to test, optimizing it for each test would be too time consuming. Therefore, the tests shown below should not be considered as the best results possible of the algorithm. They only aim to provide an insight on how the above mentioned situations affect its accuracy. In the end, a summary of the best accuracy in each dataset will be provided, where it is possible to compare ALSR to other sparse representation based classification algorithms (e.g. dictionary learning algorithms) and other popular face recognition algorithms.

Given that the ALSR algorithm is a resource intensive algorithm, it takes a considerable amount of time to test, especially in large datasets. Therefore, the results will not be presented in the same manner than the SRC results. Instead, only the results deemed interesting will be presented, as well as the parameters used to achieve those results.

In order to measure the accuracy of the algorithm, the first images of each class will be used for training, and the remaining for testing.

5.2.1 Number of training images

ALSR is an algorithm which is expected to perform well even with a low amount of training samples.

	Accuracy	
Number of training samples	SRC	ALSR
1	54.0%	58.4%
3	73.8%	78.0%
5	81.4%	91.4%
7	88.5%	94.5%
10	95.6%	97.2%

Table 5.3: Accuracy with varying number of training samples (GT Dataset).

Table 5.3 shows the accuracy of the SRC and ALSR algorithms for the GT dataset. It should be noted that, unlike for SRC, the parameters used for the ALSR were not optimized, since optimizing such a large number of parameters would take too much time. Still it performed admirably, beating the SRC algorithm by a significant margin for any amount of training samples.

5.2.2 Image Alignment

The ALSR algorithm is somewhat resistent to misaligned face images due to two factors:

1. Portions of a face image that are slightly misaligned will still be considered, since it considers patches from a wide area.

2. If the image is severely misaligned, only a few discriminative patches should have a high enough SCI to be considered.

The factors mentioned above help the ALSR algorithm perform reasonably well when using misaligned images. However, aligned images are still preferred as they achieve the best results. Given that the SRC algorithm has no precautions against misaligned images, ALSR is expected to perform much better under such conditions.

Once again, the GT and LFW datasets will be used, since they both feature misaligned face images. The ten first images from each class will be used for training and the remaining for testing. In the case of the GT dataset this means that each class has five test images. In the case of the LFW dataset, each class has a variable number of testing images.

	SRC		ALSR	
	Aligned	Unaligned	Aligned	Unaligned
GT	95.6%	44.4%	97.2%	70.8%
LFW	64.5%	23.4%	80.6%	48.4%

Table 5.4: Accuracies for aligned and misaligned datasets (SRC and ALSR).

Table 5.4 shows the accuracies obtained for the aligned and misaligned versions of the datasets. As expected, the accuracy of both algorithms decreases when using the misaligned versions of the datasets. The ALSR, however, shows an acceptable drop in accuracy, performing only about 30% worse. By comparison, the SRC algorithm exhibits a drop in accuracy of around 60%.

In conclusion, the ALSR algorithm shows a much greater resistance to misalignment, when compared to the SRC algorithm. Despite not being tested, the dictionary learning algorithms mentioned previously on Chapter 3 are also expected to perform poorly when used with misaligned images, since they also do not have precautions in place against this issue.

5.2.3 Patch Overlap

In the ALSR algorithm, the patches can overlap with each other or not. This will depend on the number of patches and their size. As an example, if the image has 90 pixels of width and the patches have 10 pixels of width, if $p_x > 9$ than the patches will have overlap.

Figure 5.9 shows the same face image divided into two different configuration of patches. The patches from both images are squared and are of the same size. In the first image $p_x = 6$, and there is no overlap of the patches. Therefore, each pixel of the image only belongs to one patch. In the second image $p_x = 12$, and the patches overlap each other

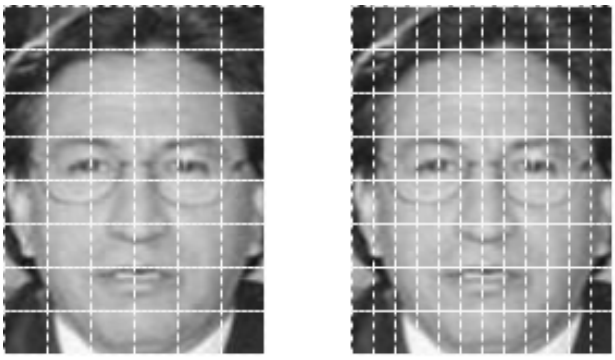


Figure 5.9: Overlapping of patches.

horizontally. Consequently, each pixel of the image can belong to multiple patches (in this particular case to 2 patches). In neither image the patches overlap themselves vertically.

While increasing the amount of patches results in a duplication of some information, therefore making the algorithm more resource intensive, it improves the chance of encountering discriminative patches that result in greater accuracy.

In the following tests, for each dataset, ten images were used for training and the reaming images were used for testing.

	Overlap	No Overlap
GT	99.2%	97.2%
LFW	80.6%	50.3%

Table 5.5: Accuracy with/without patch overlap.

Table 5.5 shows the accuracies obtained when utilizing overlapping patches and when not. The GT dataset, since it features reasonably constrained conditions, features a high percentage of accuracy in either case, only benefiting slightly from overlapping the patches. It should be noted that by overlapping the patches, since the number of patches and overall information increases, so does the time it takes to test one face image. With the overlap, the algorithm takes roughly five times more to test. A much bigger increase in accuracy is observed when dealing with the LFW dataset. Since the dataset features fully unconstrained conditions, the increase in the amount of patches clearly improves the discriminative capability of the algorithm.

5.2.4 Best Results

In this section, the best results achieved with the SRC and ALSR algorithms are disclosed and compared with other face recognition algorithms.

For reference, the eigenface algorithm achieves around an accuracy of approximately 60% on the LFW dataset [48] and deep learning methods, like the VFF-G [4], achieve an accuracy of 97.7%.

For this test, the ten first images of each class were used as training images and the remaining images were used for testing.

	GT	Yale	LFW
SRC	95.6%	96.3%	64.5%
ALSR	99.2%	98.0%	80.6%
FDDL	-	-	74.8% [40]
ASR	-	-	78.5% [40]

Table 5.6: Table showcasing the best results obtained with the respective algorithms.

Showcased in Table 5.6 are the best results obtained by each algorithms in the test conditions mentioned above. As the table shows, the ALSR algorithm achieves the highest accuracy of the group in all datasets. It does so at the expense of testing time. Since it is an extremely computational intensive algorithm, the test time for each image on the LFW dataset is around 17 seconds. While this problem is not as obvious in smaller datasets (on the YALE dataset it takes about 1 seconds per image), the time if takes to test an image is still orders of magnitude greater than the test time of SRC (in this test SRC took about 8 seconds to test all the 2106 testing images of the YALE dataset). The ASR algorithm can be seen as a lighter version of the ALSR, since it uses a smaller amount of randomly located patches to perform the classification, instead of patches from the whole image. While it does not reach the same level of accuracy of the ALSR, it still performs admirably and is less resource intensive by nature.

It should be noticed that the parameters of the ALSR algorithm were tuned manually. An automatic parameter search could potentially yield better results, but it would require either an unreasonable amount of time or a great amount of computational resources.

5.3 Conclusion

As seen by the experimental results above, the SRC and ALSR algorithms are both suitable candidates for performing face recognition. In constrained conditions, the use of SRC is recommended, since the results are approximate to those of the ALSR, while being much faster to perform the recognition. On the other hand, if the conditions are absolutely unconstrained and there is no limitation of either testing time, or computational capabilities, then the ALSR provides the best results.

While this dissertation did delve into dictionary learning algorithms, they are also a good option regarding face recognition based on sparse representation. While they take more time to train, they provide better results than the SRC algorithm and take a similar amount of time to test.

Chapter 6

Conclusion

The main objective of this dissertation was to provide an overview of sparse representation and namely its application in the field of face recognition. It successfully reached its goal, by explaining in a correct and concise manner what is sparse representation, how one can obtain a sparse representation and how it relates to face recognition. This was achieved by describing multiple algorithms for classification that were applied to face recognition scenarios thorough the literature.

As demonstrated by the experimental results, sparse representation based classification algorithms provide remarkable results in constrained and semi-constrained environments, as seen by their performance on the YALE and GT datasets, and reach very good results on fully unconstrained environments, as seen by the results on the LFW dataset.

While impressive results in face recognition can be attained via sparse representation, it is my personal opinion that these results are nowadays overshadowed by the accuracies attained when using deep learning, namely using CNN's. Still it remains impressive that sparse representation methods can achieve such high accuracies, only using such a small amount of training samples, when compared with the millions of samples required by deep learning methods.

6.1 Future Work

There are two different directionS that can be taken from the work done for this dissertation.

Firstly, the studies of sparse representation based face recognition presented can be further improved by testing new algorithms, including the dictionary learning based ones mentioned in Chapter 3. An attempt to improve upon these algorithms can also be pursued.

Secondly, one could attempt to optimize the algorithms presented in this dissertation. Both the minimization performed for sparse representation and the dictionary learning algorithms are highly parallelizable. Therefore, it would be interesting to attempt an implementation of these algorithms to be run in a GPU, in order to measure the performance gains. This would benefit the ALSR algorithm the most, since the patches could also be processed in parallel. If the increase in speed was significant, this would allow a faster automated parameter search, in order to optimize the algorithm and reach greater accuracies.

Bibliography

[1] Arvind Ganesh, Andrew Wagner, Zihan Zhou, Allen Y. Yang, Yi Ma, and John Wright. Face recognition by sparse representation. *Compressed Sensing: Theory and Applications*, pages 515–539, 2009.

[2] Aleix Martínez and Robert Benavente. The AR face database. Technical Report 24, Computer Vision Center, Bellatera, Jun 1998. Cites in Scholar Google: http://scholar.google.com/scholar?hl=en&lr=&client=firefox-a&cites=15042646876214698l2.

[3] Brendt Wohlberg. SPORCO. Technical report, 2017.

[4] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *BMVC*, 2015.

[5] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, Dec 2001.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[7] Shaikh Muhammad Allayear, Md Fakrul Abedin Bhuiyan, Mirza Mohtashim Alam, S. Rayhan Kabir, Md Tahsir Ahmed Munna, and Md. Samaun Hasan. Human face detection in excessive dark image by using contrast stretching, histogram equalization and adaptive equalization. *International Journal of Engineering & Technology*, 7(4):3990–3994, 2018.

[8] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, June 1991.

[9] Masakazu Matsugu, Katsuhiko Mori, and Takashi Suzuki. Face recognition using SVM combined with CNN for face detection. In Nikhil Ranjan Pal, Nik Kasabov, Rajani K. Mudi, Srimanta Pal, and Swapan Kumar Parui, editors, *Neural Information Processing*, pages 356–361, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[10] Georgia tech face database. http://www.anefian.com/research/face_reco.htm. Accessed: 2019-07-10.

[11] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[12] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[13] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In *NIPS*, 2012.

[14] Stefan Lee. Sparse coding for object recognition. https://www.semanticscholar.org/paper/Sparse-Coding-for-Object-Recognition-Lee/3db4bdee7bc03239ef25d23cf8dc14ce5a5300e2, 2013. Accessed: 2019-04-14.

[15] Arthur Szlam, Karol Gregor, and Yann LeCun. Fast approximations to structured sparse coding and applications to object classification. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 200–213, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[16] Martin Rehn and Friedrich T. Sommer. A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of Computational Neuroscience*, 22(2):135–146, 2007. QC 20100916.

[17] T. T. Cai and L. Wang. Orthogonal matching pursuit for sparse signal recovery with noise. *IEEE Transactions on Information Theory*, 57(7):4680–4688, July 2011.

[18] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Found. Trends Mach. Learn.*, 4(1):1–106, January 2012.

[19] David L. Donoho. For most large underdetermined systems of equations, the minimal 1-norm near-solution approximates the sparsest near-solution. 2006.

[20] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[21] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 04 2004.

[22] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 693–696, April 2009.

[23] Chenghong Yang and Hongjuan Zhang. Low-rank sparse representation with pre-learned dictionaries and side information for singing voice separation. *Advances in Pure Mathematics*, 08:419–427, 01 2018.

[24] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, Nov 2006.

[25] K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, volume 5, pages 2443–2446 vol.5, March 1999.

[26] Lijun Bao, Wanyu Liu, Yuemin Zhu, Zhaobang Pu, and Isabelle E. Magnin. Sparse representation based MRI denoising with total variation. *International Conference on Signal Processing Proceedings, ICSP*, pages 2154–2157, 2008.

[27] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[28] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust Face Recognition via Sparse Repression. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[29] Dong Wen, Peilei Jia, Qiusheng Lian, Yanhong Zhou, and Chengbiao Lu. Review of sparse representation-based classification methods on EEG signal processing for epilepsy detection, brain-computer interface and cognitive impairment. *Front Aging Neurosci*, 8:172–172, Jul 2016. 27458376[pmid].

[30] Alaa Tharwat. Principal component analysis - a tutorial. *International Journal of Applied Pattern Recognition*, 3:197, 01 2016.

[31] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: Applications to image and text data. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 07 2001.

[32] A Youssef. Analysis and comparison of various image downsampling and upsampling methods. page 583, 01 1998.

[33] Andrew Wagner, John Wright, Arvind Ganesh, Zihan Zhou, Hossein Mobahi, and Lei Yu. Toward a practical face recognition system: Robust alignment and illumination by sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34:372 – 386, 03 2012.

[34] Yongkang Wong, Mehrtash Tafazzoli Harandi, and Conrad Sanderson. On robust face recognition via sparse encoding: the good, the bad, and the ugly. *CoRR*, abs/1303.1624, 2013.

[35] Dan Kalman. A singularly valuable decomposition: The SVD of a matrix. *College Math Journal*, 27:2–23, 1996.

[36] Z. Jiang, Z. Lin, and L. S. Davis. Label consistent K-SVD: Learning a discriminative dictionary for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2651–2664, Nov 2013.

[37] Meng Yang, Lei Zhang, Xiangchu Feng, and David Zhang. Fisher Discrimination Dictionary Learning for sparse representation. *Proceedings of the IEEE International Conference on Computer Vision*, pages 543–550, 2011.

[38] Shu Kong and Donghui Wang. A dictionary learning approach for classification: Separating the particularity and the commonality. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 186–199, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[39] Yong Xu, Zhengming Li, Jian Yang, and David Zhang. A Survey of Dictionary Learning Algorithms for Face Recognition, 2017.

[40] Domingo Mery; Sandipan Benarjee. Recognition of faces and facial attributes using accumulative local sparse representations. 2018.

[41] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[42] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11:19–60, March 2010.

[43] D. Mery and K. Bowyer. Face recognition via adaptive sparse representations of random patches. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 13–18, Dec 2014.

[44] Qing-Song Xu and Yi-Zeng Liang. Monte carlo cross validation. *Chemometrics and Intelligent Laboratory Systems*, 56(1):1 – 11, 2001.

[45] Liansheng Zhuang, Tsung Han Chan, Allen Y. Yang, S. Shankar Sastry, and Yi Ma. Sparse Illumination Learning and Transfer for Single-Sample Face Recognition with Image Corruption and Misalignment. *International Journal of Computer Vision*, 114(2-3):272–287, 2015.

[46] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[48] LFW classification results. `http://vis-www.cs.umass.edu/lfw/results.html#eigenfaces`. Accessed: 2019-07-02.

Appendix A

Tools

- Programming Language - Python
- Sparse Coding library - SPAMS [42]
- Similarity Search - FAISS [41]
- Image Loading/Processing - OpenCV [46]