Technical University of Crete

School of Electrical and Computer Engineering

Digital Image and Signal Processing Laboratory

Design and Comparison of Machine Learning and Computer Vision Methods in Face Recognition

by

Stelios Bantourakis

a thesis submitted in partial fulfillment of the requirements for the Diploma in Electrical and Computer Engineering

August 2017

Abstract

In recent years, face recognition systems have reached mainstream popularity through smartphones and social networking sites. At the same time, the general field of image recognition has been boosted by the revitalization of neural networks in the form of deep neural networks (deep learning). Thus, the need to study and understand their implications in subfields of image processing, such as face recognition, becomes of particular importance. In this thesis, we examine both areas of classic methods and state-of-the-art deep learning networks, for a deeper and more complete understanding of the application area. More specifically, Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Independent Component Analysis (ICA) were tested under the additional twist of Discrete Wavelet Transform (DWT) as a form of preprocessing. This proposed combination of wavelet decomposition for feature extraction and statistical data organization for feature reduction proves to be an efficient prospect in face recognition. DWT is a prominent method in digital image processing as it is utilized by the JPEG compression standard but also in the security field through steganography. Regarding the current state-of-the-art in deep learning, we tested transfer learning as an option for reutilizing already trained networks on big datasets. A high-performing network, pre-trained on a different dataset than the test one was used and managed to achieve accurate results, thus confirming that deep neural networks form the future in the field. Furthermore we attempt a conceptual association of the two methodologies tested. In particular, the classic methodology is based on rigorous statistical concepts, whereas deep learning is still using network modules as black boxes. Through our study, we can provide concrete associations of processing modules that enable the rigid justification of deep neural network units.

Acknowledgements

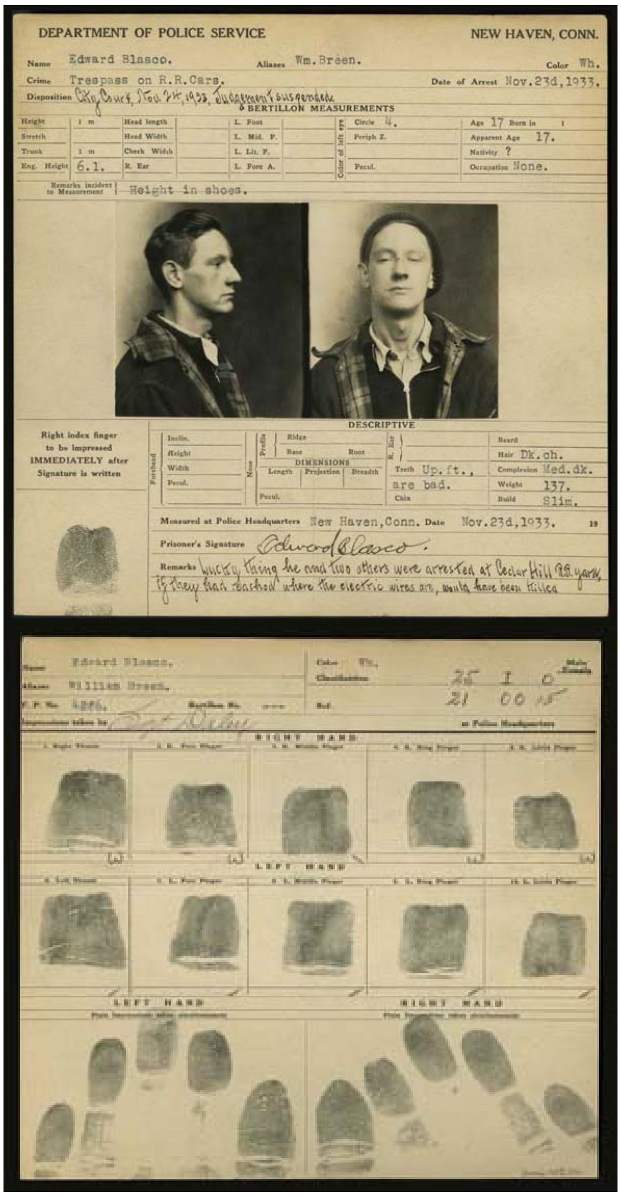
This work would not be possible without the contribution of many great people. First and foremost, I would like to thank my supervisor, Professor Michael Zervakis, for giving me the chance to work on the field of face recognition. His guidance was invaluable. I am also grateful to my family, for always believing in me. Thank you my friends, Manos M. and Manos P. for putting up with me and providing much needed support. The latter also proofread this thesis, so contact him for any errors you may find. Last but not least, I would like to thank the committee for taking the time to grade this thesis.

Table of Contents

List of Figures

1 Introduction - Motivation

In recent years there has been a boom in the usage of body or behavioral characteristics as a means of improving security in systems and applications [1], replacing the classic password authentication approach. These characteristics are called biometrics. Biometrics however, are not a new concept. As early as the 14th century, merchants used ink to take children's fingerprints. More recently, in the late 19th century, Alphonse Bertillon, a French criminologist, proposed a method of verifying criminals using their body measurements. This method was called the Bertillon system, or bertillonage [2]. It was widely used until fingerprint analysis was proven to be more robust.



DEPARTMENT OF POLICE SERVICE NEW HAVEN, CONN.

Name Edward Blasco. Aliases Wm. Breen. Color Wh.

Crime Trespass on R.R.Cars. Date of Arrest Nov.23d,1933.

Disposition City Court, Nov. 24, 1933, Judgement suspended.

BERTILLON MEASUREMENTS

						Color of left eye			
Height	1 m	Head length		L. Foot			Circle 4.	Age 17	Born in
Stretch		Head Width		L. Mid. F.			Periph. Z.	Apparent Age	17.
Trunk	1 m	Cheek Width		L. Lit. F.				Nativity	?
Eng. Height	6.1.	R. Ear		L. Fore A.			Pecul.	Occupation	None.

Remarks incident to Measurement: Height in shoes.

Right index finger to be impressed IMMEDIATELY after Signature is written

DESCRIPTIVE

Forehead	Inclin.	Nose	Profile: Ridge	R. Ear		Beard	
	Height		Base / Root			Hair	Dk. ch.
	Width		DIMENSIONS: Length / Projection / Breadth	Teeth	Up. ft., are bad.	Complexion	Med. dk.
	Pecul.					Weight	137.
			Pecul.	Chin		Build	Slim.

Measured at Police Headquarters New Haven, Conn. Date Nov.23d,1933. 19

Prisoner's Signature Edward Blasco.

Remarks Lucky thing he and two others were arrested at Cedar Hill R.R. yard, if they had reached where the electric wires are, would have been killed

Name Edward Blasco. Color Wh.

Aliases William Breen. Classification 25 I 0 / 21 00 15

F. P. No. 4266. Bertillon No. ---- Ref.

Impressions taken by [illegible] at Police Headquarters

RIGHT HAND

LEFT HAND

LEFT HAND RIGHT HAND

Figure 1-1: Arrest card utilizing Bertillon's method [2]

Although fingerprinting is still used in many biometric applications, the availability of more resources (computational power, cameras) has led to the increase in popularity of more complex methods, like iris and face recognition. The term recognition usually refers to two similar but at the same time very different tasks. In the context of faces, face identification is the task of assigning an identity to an unlabeled face image presented to the recognition system. Face verification is the task of confirming or denying that a face image and an identity presented to the system, match. As face verification can be seen as a yes or no question, face identification is considered to be a more complex and demanding task. Naturally, it also handled differently. Faces are considered to be the primary biometric characteristic by the International Civil Aviation Organization and are now used in all passports, while fingerprint analysis is a secondary and not mandatory biometric. Even the FBI actively uses and maintains a face recognition database comprised of half the US population [3]. One advantage of face recognition is that it is more flexible, in the sense that images from different angles or sources (e.g. CCTV) can be used and without the subject complying, as is the case with iris and fingerprint analysis.

In this thesis, the problem of face identification (referred to as face recognition from now on) will be tackled. A face recognition system generally looks like the figure below.

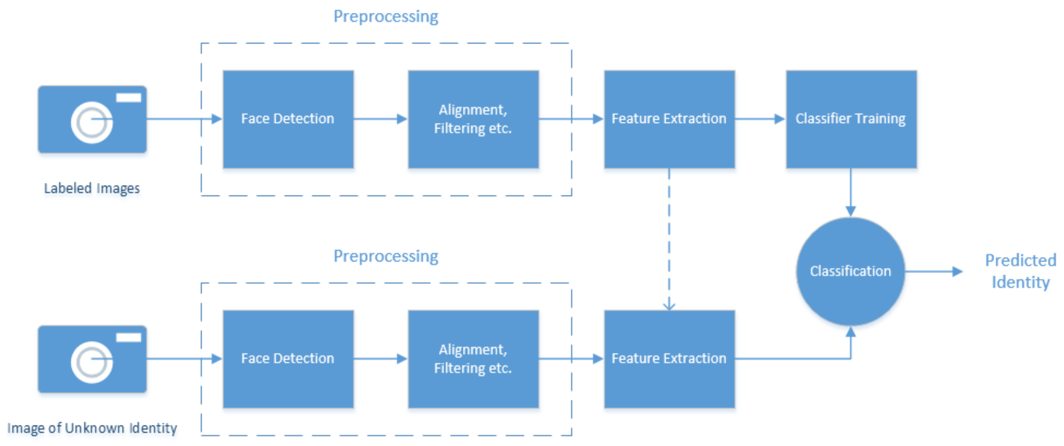


Figure 1-2: An example of a face recognition system

First, images containing examples of known identities are gathered (labeled images). These images can range from portraits, to full body images and be taken under varying conditions. As such, there is a need to somehow “normalize” them. A first step is to actually locate the faces in the images and crop them, to minimize the unrelated information (non-face pixels). Afterwards, more preprocessing steps can be applied, like alignment and filtering (to increase contrast for example). Alignment is very important when using methods that are not translation invariant in the following step. When the images are ready, a wide range of methods can be used to extract valuable information suited for classification. This information has the form of coefficients called features and the process is called feature extraction. We don’t want to represent the images by simply using their pixel values, because they can contain redundant or unwanted information. By applying feature extraction methods, we wish to generate data from our original images that can be used to better discern between identities when using the computer (in contrast to when we use our eyes, where pixels are obviously more useful). After the features from our labeled images are extracted, a classifier is trained using the features and their labels. A trained classifier can be seen as a function with input an unlabeled sample and output one of the training labels. After the classifier is trained, unlabeled images to be classified can enter the system. The same preprocessing and feature extraction steps are applied to extract the unlabeled feature vector. Sometimes, the feature extraction method requires information from the labeled data and that is represented by the dashed line. Such methods can for example project both labeled and unlabeled images to a common space defined by the labeled data to extract the features. In the last step, the unlabeled features are presented to the classifier and it produces a label as an output.

Moving forward, we will focus on feature extraction methods, their performance through classification accuracy (mostly) and possible associations between them. More specifically, we will study some global statistical and non-statistical methods. The term global refers to the handling of the image during feature extraction. Local methods handle different regions on an image separately (e.g. Local Binary Patterns) while global methods handle the image as a whole. We will examine the potential of new approaches regarding feature decomposition in order to attempt to organize and cluster those features. For this purpose, we process features obtained in the wavelet domain through data dimensionality methodologies such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Independent Component Analysis (ICA). Following this analysis, we explore the issues of deep neural networks in preserving and organizing useful features for face recognition. Thus, we use deep networks pre-trained with large databases and tested the efficiency in face recognition problems, a process called transfer learning. For this purpose, we remove the classification layer and replace it with our own which is then trained on a new dataset, based on the features of the previous layers.

The motivation behind the choice of these methods is simple. PCA, LDA and ICA are popular linear dimensionality reduction techniques utilized in many different domains, as well as face recognition, so studying them provides a much needed solid foundation. The wavelet transform is another versatile technique, used in digital signal and image processing. More notably, the image compression standard JPEG utilizes the wavelet transform [4]. It has also been proven useful in security applications through steganography [5]. Deep Learning, which is the use of deep neural networks, has been proven to be the current state-of-the-art in many applications and as such can’t be ignored.

Through our analysis, we recognize certain similarities of deep neural networks and data decomposition and reduction schemes such as those in our first approach and based on that we propose specific explanation and justification for the various layers of deep neural networks which at this point are mainly used in a black box form.

1.1 Related Work

Face recognition via deep leaning and face recognition in general is an active research area with rising popularity.

It is becoming so widespread that it is even offered “on the cloud”. Kairos is a Software as a Service (SaaS) provider, specializing in face recognition and emotional analysis [6]. It even offers free access to its API for personal (limited) use. Google’s Vision API [7] and Amazon’s Rekognition [8] also offer face recognition capabilities, among other things. The two major pros of using a SaaS are that we only need to worry about providing the data and treat the rest of the system as a black box that returns an output and that any computations are executed remotely, without burdening the local hardware.

Facebook has also implemented a face recognition scheme. It creates biometric templates based on face characteristics (e.g. distance between the eyes) from tagged pictures and suggests identities in untagged pictures that match that template. [9]

Additionally, commercial handheld devices have started including biometric recognition schemes, albeit not always with the desired degree of success, as we will see in the second to last section of this thesis [10]. MasterCard is ready to start verifying online payments using faces [11]. In a more obscure case, face recognition algorithms have also been implemented experimentally in comedy clubs in Spain with positive results. The system detects how many times a customer laughs and increases the ticket price accordingly [12].

Research-wise, the effects of wavelet decomposition and statistical dimensionality reduction techniques have been tested before. In [13], the authors combined PCA with wavelet decomposition and attempted to find the most suitable wavelet for the task of face recognition. In [14], the dimensionality reduction technique used was ICA. LDA was used in [15]. Since the resurgence of deep neural networks, many deep architectures have been proposed for face recognition. DeepFace [16] is a popular architecture by the Facebook research team. DeepID [17] is another architecture that is in development since 2014 that already has two revisions (DeepID2 and DeepID3) and has achieved record setting performance.

1.2 Contributions and Outline

This thesis is a study of both "classic" linear statistical methods and the current non-linear state-of-the-art. Even though dimensionality reduction methods have been combined with wavelet decomposition for face recognition, most studies focus on only one method. The difference in our approach is that we will implement all three major algorithms and compare their results directly. We will also make comparisons about the wavelet choice, as it is still up to debate which the optimal one is and make associations with deep learning.

The next chapters are organized as follows. In the second chapter, we briefly describe our process for selecting datasets and present the two that we will be using. The third chapter, provides the needed theoretical background on the statistical methods used and on the classification process. The fourth chapter is comprised of an introduction to the state-of-the-art, deep learning, some problems commonly faced and the approach we will be taking. Among other things, popular layer architectures are explained and related bibliography is presented regarding the concept of transfer learning. Chapter 5 goes into detail about the specific combinations of methods proposed, libraries used and the training scheme. Chapter 6 contains our results and comparisons with recent bibliography. In chapter 7, we include examples of threats in machine learning-assisted face recognition. Even though in this thesis we didn't study countermeasures, it is very important to be security-aware as we engineers have a responsibility to build not only working but also secure systems. Lastly, in chapter 8, further avenues of related studies are mentioned and we provide some associations between the two face recognition approaches we implemented, as a possible means of understanding neural networks better.

2 Face Database Selection

To actually try face recognition, we need a set of data first. There is a (not complete) list of available datasets at face-rec.org/databases/. The specifications of each dataset can vary greatly. Some can contain occlusions, significant rotations of the face or expression variations, others were shot in a controlled environment or were taken from video frames etc. They also differ in how easy they are to access. Due to the sensitive nature of the data, some databases will ask you to sign an agreement before you download them. There exist some that require you to be a professor (not a student or an employee) or even a member of a university's legal department. Naturally, there also some that charge for a license. We preferred to avoid any licensing procedures and luckily there are such databases available. Another constraint we imposed was that there shouldn't be a need for significant preprocessing before using a dataset, as we want to focus our study on feature extraction. In our case, that means that the images should be cropped around the face, with not much of the background showing. Furthermore, occlusions or heavy lighting variations should not be present. We also favored a pre-centered dataset. Centering the images is a vital preprocessing step because the methods we will be testing (or most of them) are not translation invariant. That means that differences in the positions of the faces would reduce the effectiveness of those methods (as would the other characteristics we "boycotted"), making it more difficult to extract valuable conclusions about face recognition. A not uncommon way to solve the centering problem, would be to manually select the pixels where the centers of the eyes are in each image and process the images accordingly so that they match. However, we would need to do that for each wavelet level. It certainly would not be impossible to do or even that time consuming, but since we found pre-centered datasets, there was no reason not to use them. Perhaps the most important constraint was that the database should have already been widely used for face recognition related studies. This makes sure that we are working on a high quality dataset and of course makes it easy to make comparisons with other scientific work that utilizes it.

2.1 The ORL Database of Faces

The ORL Database of Faces [18] seemed to be a good match for our needs. It contains 400 images of 40 individuals (10 per person) that were taken between 1992 and 1994. The resolution is 112 x 92. The background used is the same in all pictures and homogenous. Some differences in lighting are present but are not major. Small face rotations can be present. The facial expressions are varied and include closed eyes, smiling etc. and such variations are not consistent across all individuals. Also, some wear glasses in some of their pictures. All of the above are desirable properties that make the recognition problem not trivial. In research, it has been used by over 900 papers in IEEE alone, with a variety of local and global methodologies, including neural networks and wavelets.

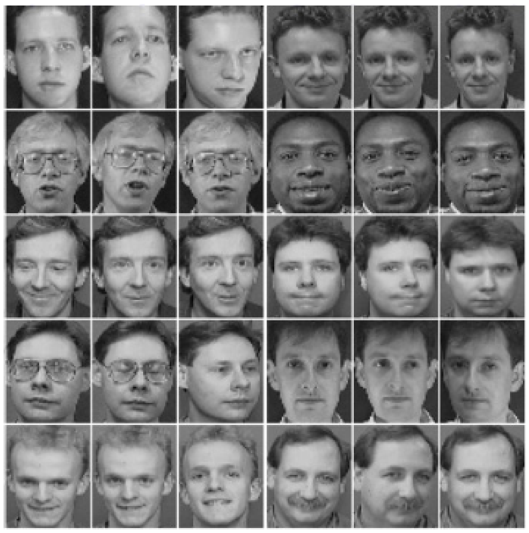


Figure 2-1: Sample from the ORL Database of Faces

Even though the variations mentioned above are present, this dataset is not exceptionally difficult to work with and we wouldn't choose it if we wanted to benchmark a new proposed state-of-the-art face recognition system. Our focus is the study of feature extraction algorithms in the past and now. For that purpose, both its content and its size (with a notable exception we will see later) are satisfactory. A more appropriate dataset for competitive benchmarking is Labeled Faces in the Wild (LFW) [19].

2.2 Georgia Tech Face Database

As its name suggests, this dataset was created in the Center for Signal and Image Processing at Georgia Institute of Technology [20]. It contains 15 images of 50 subjects (750 images total) of varying lighting, expression and most importantly, scale and background. Their dimensions vary from slightly to significantly larger than those in the ORL database (e.g 270 pixels vs 92). The subjects were not photographed in a totally controlled environment like in the ORL Database and thus were photographed from not the same distance or in the same place. Some resized samples can be seen below. A few more extreme than in the ORL database pose variations can be seen.

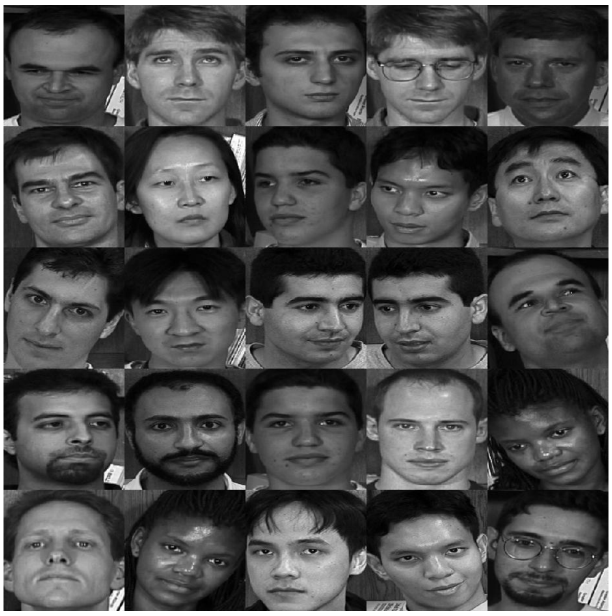


Figure 2-2: Scaled samples from the Georgia Tech database

Even though some datasets provide proposed splits for a training and a testing set, both of the datasets mentioned above allow any split.

3 Classical Decomposition, Feature Extraction and Classification Approaches

3.1 Wavelet Transform (WT)

The Wavelet Transform is a powerful tool when we need to gain information about the frequency of a signal (like the Fourier Transform), while also keeping the information about the time domain (unlike the Fourier Transform). This is especially useful when dealing with non-stationary signals, i.e. signals whose frequency varies in time.

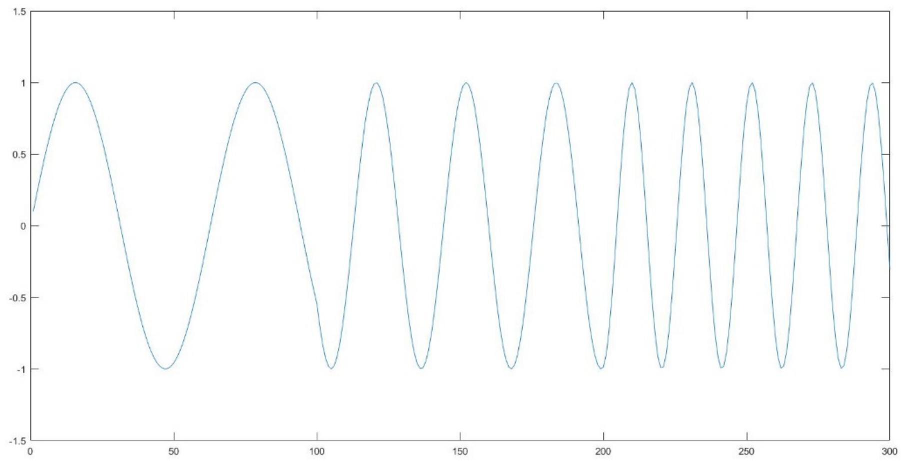


Figure 3-1: A non-stationary signal

To better understand the significance of WT, we should briefly take a look at an earlier technique, the Short-Time Fourier Transform (STFT). In STFT, the signal is divided into small enough segments, where these segments of the signal can be assumed to be stationary (constant frequency). For this purpose, a window function w is chosen. The width of this window must be equal to the segment of the signal where its stationarity is valid. This function is shifted in time by an arbitrary interval, beginning at t = 0 (the beginning of the signal). For every shift, the product of the signal and the window is computed and then Fourier Transform is applied. Here's an example (ignore the axes, they are normalized)

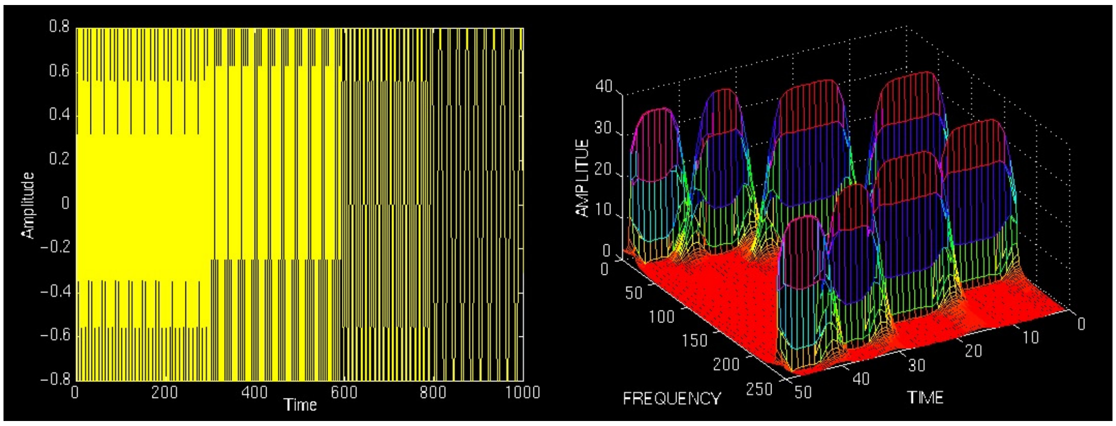


Figure 3-2: A signal and its STFT [81]

We can clearly see four peaks at four different times (and their symmetric counterparts). So, why do we need WT? This becomes obvious as we vary the window size. A narrow window gives better time resolution (basically it becomes easier to separate the peaks in time) and worse frequency resolution, while a wider window does the opposite.

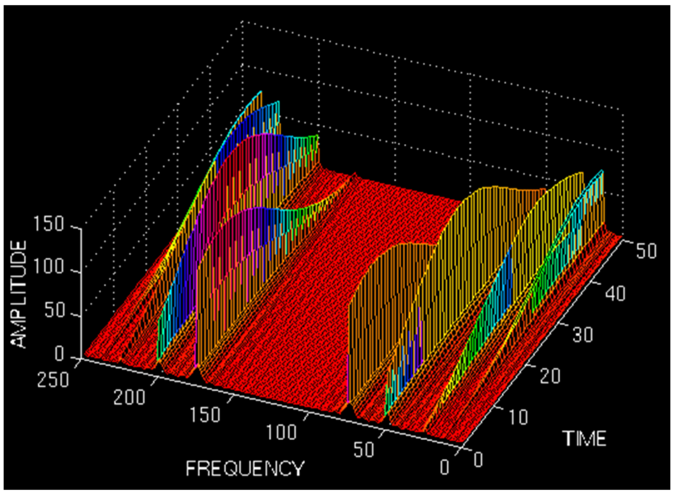


Figure 3-3: An example of really bad time resolution [81]

WT gives us the ability to achieve different resolutions for different frequencies. In the continuous signal case, it is defined by

$$CWT_x(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^*(\frac{t-\tau}{s})\, dt$$

where τ represents the shifting in time, ψ is the mother wavelet function (equivalent to the window in STFT) and s, called scale, corresponds to the inverse of the frequency and compresses or dilates the wavelet. The term wavelet, means a small wave, small in the sense that is of finite length and wave because it is oscillatory. So, in WT the window/wavelet is not only shifted in time but also compressed or dilated, before interacting with a signal.

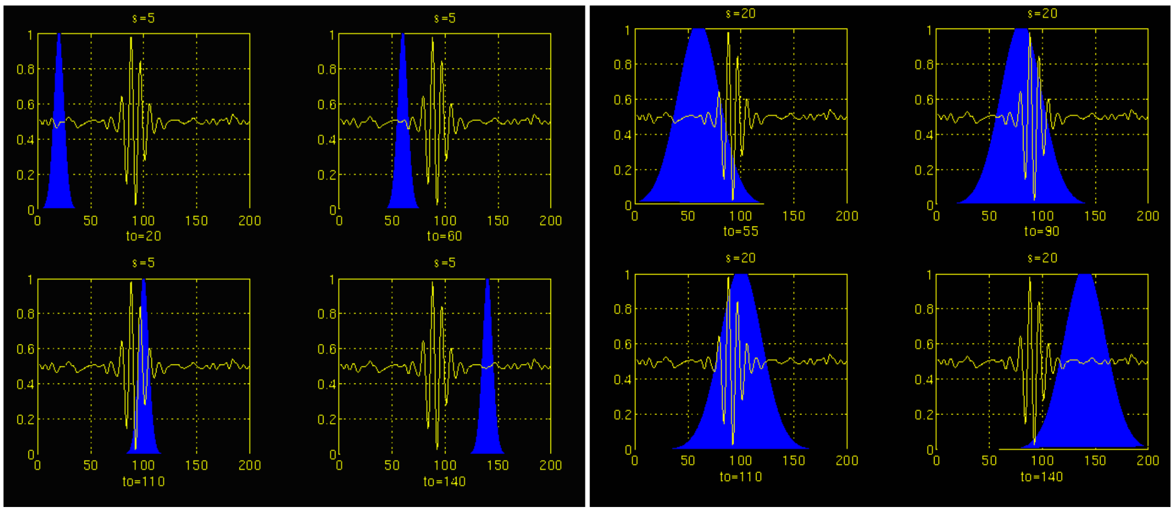


Figure 3-4: Visual example of CWT for two different scales [81]

When the wavelet is narrow (high scale – low frequency) the CWT is computed over a small portion of the signal (in the time domain), resulting in better time resolution, but bad frequency resolution. When the wavelet is wide (low scale – high frequency), however, the wavelet can cover most of the signal. That makes it difficult to discern information in time, but easier to do so in frequency. For the example above, when s = 20, the oscillation in the middle is only represented with 3 CWT values (coefficients) because the signal and the wavelet only overlap 3 times. Fortunately, in practical applications, low scales (high frequencies, like noise) appear in short bursts while high scales (low frequencies) last the duration of the signal.

In the discrete case the WT (DWT) is applied through a filter bank, which is an array of filters with different bands.

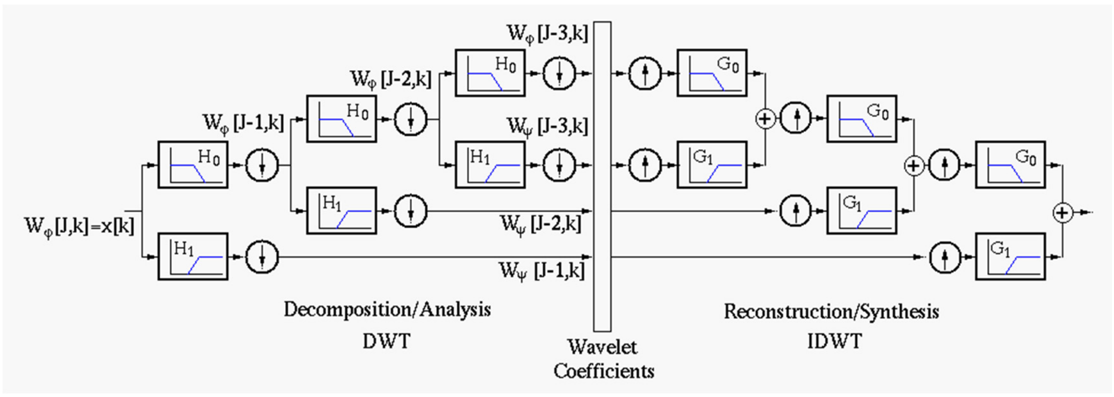


Figure 3-5: DWT and Inverse DWT filter bank [82]

At each level (or scale) of the filter bank, the signal is filtered using the same high and low pass filters and then is downsampled (or upsampled, in Inverse DWT) by a factor of two. Downsampling reduces the sampling rate of a signal by keeping only every Xth sample (X=2 when downsampling by 2).

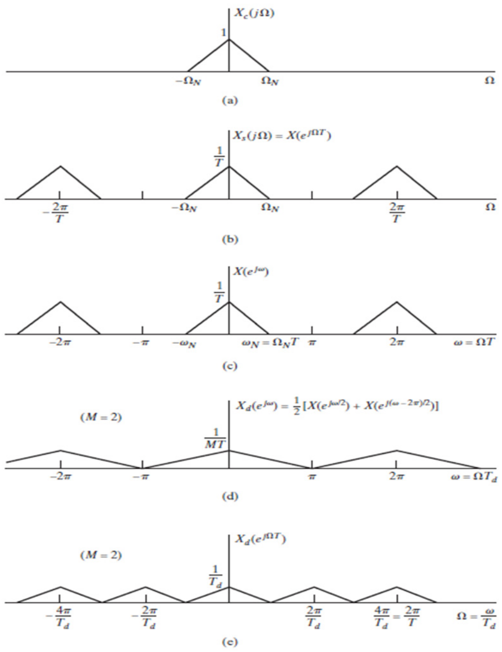


Figure 3-6: Effect of downsampling on the frequency domain [83]

When our signal is 2-D (like an image) DWT is generalized by the below architecture

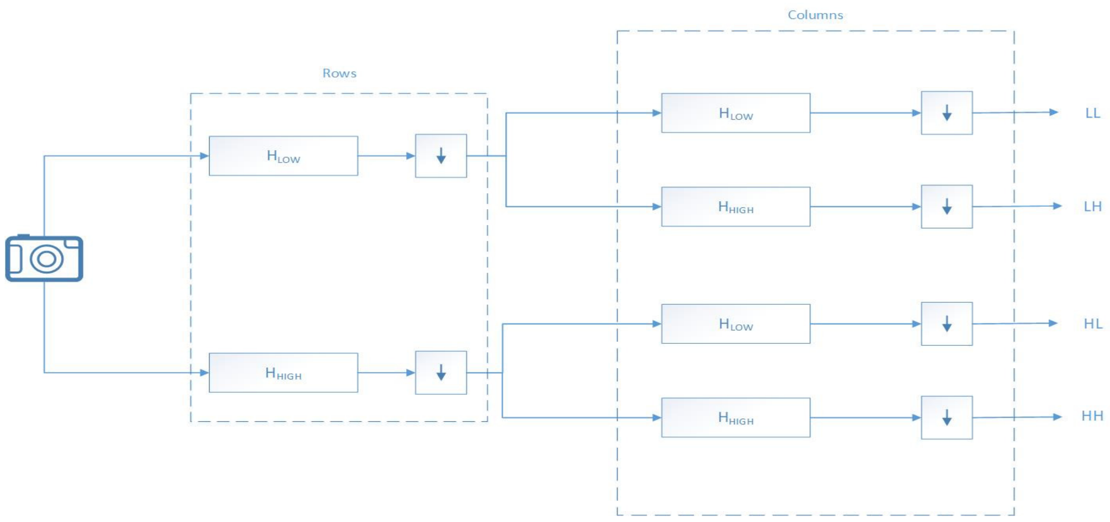


Figure 3-7: 2-D DWT filter bank

The rows and the columns are filtered with both the high pass and the low pass filters. The result is four filtered and downsampled bands/signals the Low Low (LL), Low High (LH), High Low (HL) and the High High (HH) band. Similarly to the 1-D case, DWT can be applied for multiple levels.

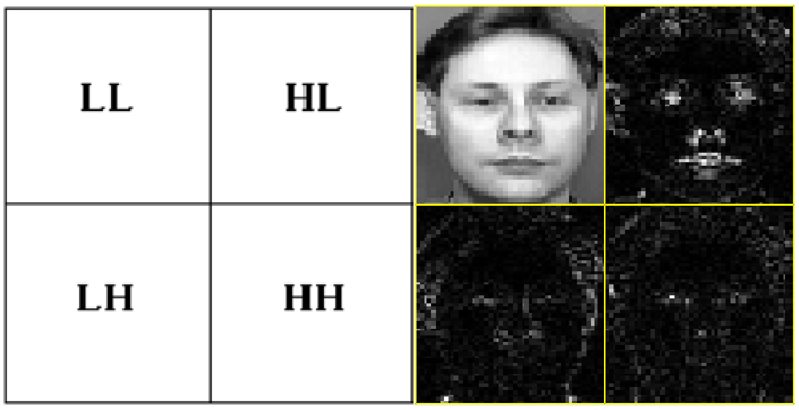


Figure 3-8: 1-level DWT from the ORL database

It is obvious that the LL band retains the most information and that is why it is also called the 'approximation' band. The other bands contain vertical, horizontal and diagonal details.

We should address the choice of wavelet function. Up to this point, we have purposefully ignored it, as there in no golden rule, the choice is application specific. We will, however, very briefly mention one of the most popular wavelet families, the Daubechies wavelets [21]. As the name suggests, they were based on the work of Ingrid Daubechies and each wavelet type consists of two sets of a high and a low pass filter, one for DWT and one for Inverse DWT.

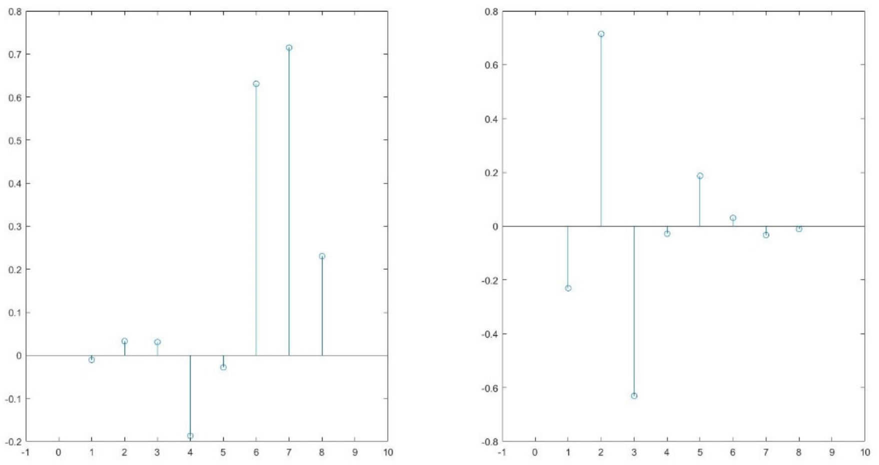


Figure 3-9: Low and high pass filter coefficients for Daubechies 4 (db4) DWT

Before we close this section off, we need to stress that the above are just some basic points of the Wavelet Transform. WT is a big part of Digital Signal Processing and as such, has a very strong and complicated mathematical basis. A proper introduction that would do more than just scratch the surface would be tens of pages long. Any further interested individual is encouraged to check another piece of Daubechies' work [22] for an in-depth look at wavelets.

3.2 Principal Component Analysis (PCA)

The idea behind PCA originated in a 1901 paper by Pearson [23]. In 1931, Hotelling [24] gave another definition to this method, based on maximum variance. In this thesis, we are going to adapt the maximum variance approach [25]. But first, we should briefly mention what eigenvectors and eigenvalues are.

Given an n x n square matrix A, an eigenvector v is a vector of size n x 1 that satisfies

$$Av = \lambda v$$

where λ is a scalar and is called an eigenvalue. The eigenvalues can be computed by solving the determinant of (A – λI) where I is a unit vector. With that in mind, we can proceed to PCA.

Given a set of observations x_n, n = 1 … N, our goal is to project these observations onto a lower dimensionality space, that maximizes the variance of the projected data. That means, that if each of our data vector x_n is a vector of size 1 x K, after the projection we wish to have observations of size 1 x M, where M<K. As to why we would want to do something like that, the reasoning is simple. Any observation can have redundant data. For example, a face image can contain background pixels. By reducing the original dimensionality, we hope to minimize the redundant information, so that our classifiers won't be affected as much by it. From a computational perspective, it is also quicker to process less data. That can make a huge difference when using more complex methods, like Support Vector Machines (SVM) [26].

For simplicity, let M=1. The direction of this space is a K-dimensional vector u_1 that we consider to also be unitary ($u_1 u_1^T$=1, T denotes transpose) without loss of generality. The projection of a data vector to this space is defined as $u_1^T x_n$ and the dimensionality of the data is (1 x K) x (K x 1) = 1 x 1 .

The variance is given by

$$\frac{1}{N}\sum_{1}^{N}\{u_1^T x_n - u_1^T x'\}^2 = u_1^T S u_1$$

where x' is the mean of the projected data and S is the covariance matrix defined as

$$S = \frac{1}{N}\sum_{1}^{N}(x_n - x')(x_n - x')^T$$

Notice that S should be a K x K square matrix. As we stated before, our goal is to maximize the variance $u_1^T S u_1$. This can be done utilizing Lagrange multipliers (just λ_1 in our case), while also taking into account the unitarity constraint we demanded earlier. The problem to be maximized can be formally described as

$$u_1^T S u_1 - \lambda_1(1 - u_1^T x_n)$$

By setting the derivative with respect to u_1 to zero we get

$$S u_1 = \lambda_1 u_1$$

which means that u_1 must be an eigenvector of S and obviously of size K x 1. By multiplying with u_1^T, the variance is then given by

$$u_1^T S u_1 = \lambda_1$$

So, in order to maximize the variance, λ_1 must be the largest eigenvalue of S and u_1 the corresponding eigenvector, which is called the first principal component. To increase the dimensionality of the subspace we need to find a new direction that is orthogonal to u_1(additional constraint $u_2^T u_1 = 0$). By induction it can be shown that in order to find M principal components, we just have to find the M eigenvectors that correspond to the M largest eigenvalues.

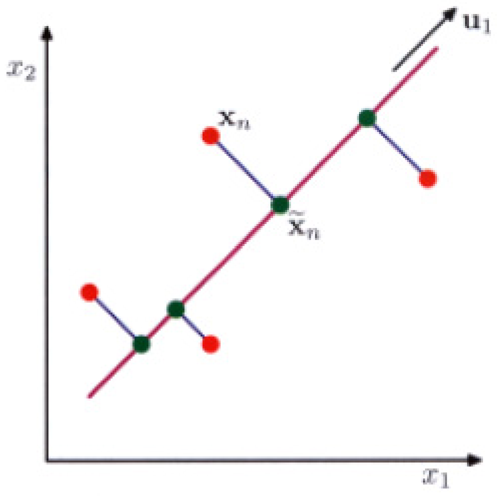


Figure 3-10: A visual example of PCA with one principal component [25]

At this point, we have understood the basics of PCA, but some points still need to be clarified. For example, this thesis is about images (2-D observations) but in the above explanation the observations are one-dimensional. Luckily, in order to apply PCA to a dataset of images we just have to concatenate all rows of an image, resulting in observations of size 1 x (K*N), assuming out images have a K*N resolution. This approach was also used in 1991 by Turk and Pentland [27] for face recognition, dubbing the method 'Eigenfaces'. To understand this name, we need to take a look at the eigenvectors of S. We have shown that they share the same dimensionality as the observations. In the case of images, if we reshape them from 1-D to 2-D (the reverse of what we did to the dataset in order to apply PCA) we get

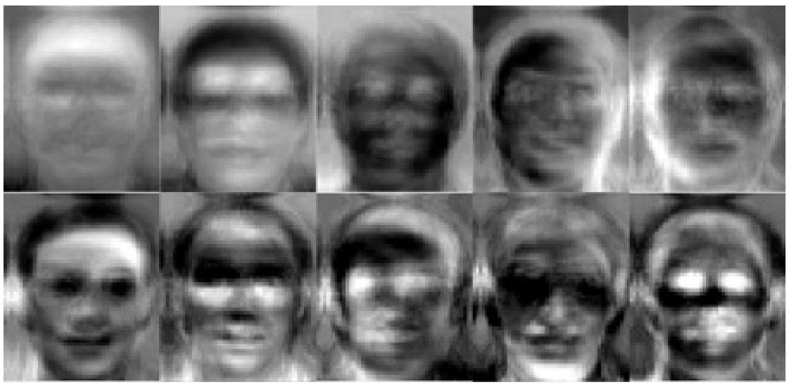


Figure 3-11: Eigenfaces from the ORL database (1-level DWT was applied first)

However, that is not the only way of applying PCA to 2D data. One alternative is presented in [28] but we will not go into detail, as it is beyond the scope of this thesis.

Another issue that arises, is how many principal components to select, i.e. how few and which directions are sufficient to project our data. Remember that the variance in each direction is equal to an eigenvalue. We sort the eigenvalue-eigenvector pairs in descending order, so that the first eigenvalues and their respective eigenvectors represent the most amount of variance. The standard as far as face recognition is concerned is to then select the first X number of eigenvectors and discard the rest. We assume that they do not offer meaningful –for classification purposes- variation to the data, like noise. An example where this is not the case is presented here [29]. We can select X by trial and error by trying an increasing amount of eigenvectors until the recognition rate plateaus or decreases. If that is not possible because of time or other constraints, [30] mentions two simple alternatives. The first, is to set a desired threshold e in total variance (typically over 90%), calculate

$$e = \frac{\lambda_\iota}{\sum \lambda_i}$$

and then choose X as the number of eigenvalues needed to surpass the threshold. The second is to choose a threshold s for the “stretch” of an eigenvalue, defined as

$$s = \frac{\lambda_\iota}{\lambda_1}$$

with λ_1 being the largest eigenvalue.

Sometimes, however, some eigenvalue-eigenvector pairs are discarded first. It has be shown that changes in illumination can contribute more to the total variance than changes among faces of different people [31]. Thus, it may be advisable to remove up to 5-6 pairs, depending on the dataset to account for variations in lighting conditions.

Before we close this section, we need to address one of the more serious questions. Can PCA always be used with the desired results? The answer is no. A visual example can help us see why:

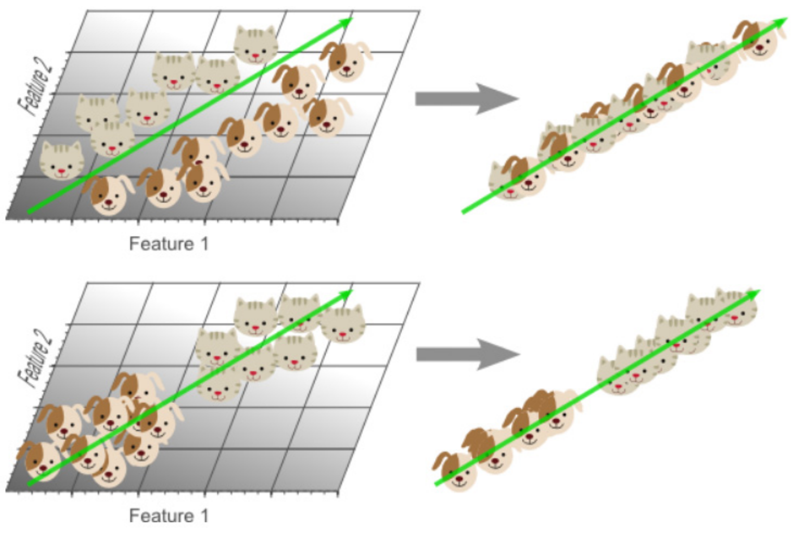


Figure 3-12: A cute example where PCA doesn't help [79]

Using the first projection, we can’t separate where the feature spaces of cats and dogs lie, even though before projecting, it was very easy to linearly distinguish them.

Despite the example showcased above not being very common, it is always important to not blindly apply methods to our data, without first understanding the problem at hand and what the end goal is.

3.3 Linear Discriminant Analysis (LDA)

While PCA might be a good method for dimensionality reduction, we must not forget that our end goal is actually classification. PCA doesn't use any kind of class information and ideally, we would like to take into account that different observations (face images in our case) come from different sources (people) when we try to find an appropriate reduced sub-space. It turns out that that there is such a method, Linear Discriminant Analysis (LDA), that originates from a 1936 paper by Fischer [32]. In 1996, it was applied in face images for face recognition [33] [34].

First, let's define the between-class variance as

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

and the within-class variance as

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in x_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

where c is the number of classes, μ_i is the mean image of class x_i and $|x_i|$ is the number of observations in class x_i. Note the similarities with the covariance matrix we mentioned in PCA.

If S_W is nonsingular the optimal projection W_{opt} is given by

$$W_{opt} = \arg\max_{\mathrm{W}} \frac{|W^T S_B W|}{|W^T S_B W|} = w_1, w_2 \dots w_m$$

where w_i, i=1…m is the set of the generalized eigenvectors of S_B and S_W defined as

$$S_B w_i = \lambda_1 S_W w_i$$

The need for a non-singular S_W is evident here, as we want to find its inverse to solve the above equation (singular matrices are not invertible). Unlike PCA, where the number of chosen eigenvectors is dependent on the dataset, a good upper bound for m is c-1 [33] so that we can keep S_W non-singular, which also is especially convenient when we have a small number of classes.

In face recognition, S_W can be singular (mostly due to the high dimensionality of the problem) so [33] proposed a different approach for W_{opt}. Instead of performing LDA on the original data, PCA is used first to reduce the dimensionality. The upper bound mentioned above still holds, fortunately. In this thesis, we choose to follow this approach.

Similarly to PCA, the eigenvectors of LDA have a distinct look dubbing the method Fisherfaces.

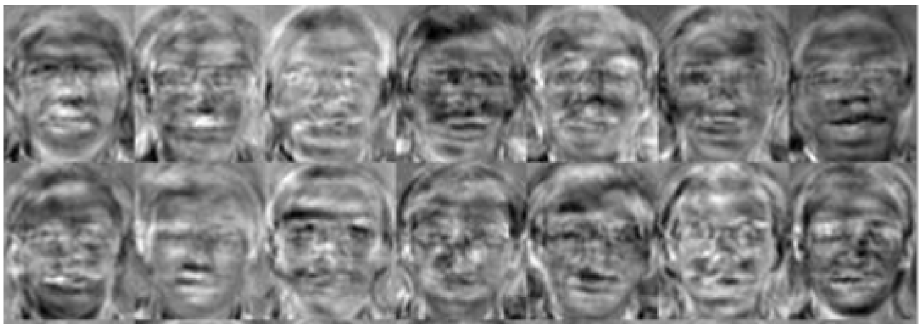


Figure 3-13: Fisherfaces from the ORL database

Lastly, we should mention that even though LDA should seemingly outperform PCA for classification purposes that is not always the case. In [35], it is shown that when the number of observations per class is small, PCA can outperform LDA.

3.4 Independent Component Analysis (ICA)

Independent component analysis aims to perform Blind Source Separation (BSS). An explanation can be given using the cocktail party problem. Let's imagine a room where two people are speaking simultaneously and are being recorded through two microphones in different locations. Each of the two recorded signals (s1, s2) is a weighted sum of the speech signals emitted by the two speakers. This can be expressed as

$$x_1 = a_{11}s1 + a_{12}s2$$

$$x_2 = a_{21}s1 + a_{22}s2$$

where $a_{11},\ a_{12},\ a_{21}, a_{22}$ are parameters that depend on the distance of the microphones from the speakers. We want to recover the two speech signals even if we don't know the values of the parameters. If we assume that s1 and s2 are statistically independent it turns out that the parameters can be estimated. This assumption is not as restrictive as it may seem [36]. ICA tackles the estimation of these parameters. To better describe the problem let's define the random vector x whose elements are the mixed signals $x_1 \ldots x_n$, the random vector s whose elements are the original signals and A, a matrix containing the parameters a_{ij}, called mixing matrix

$$x = As$$

After we estimate A and find its inverse W, the original estimated signals (called components) are given by

$$s' = Wx$$

As we mentioned above, our assumption is that the elements of s' will be independent. Formally, signals are statistically independent when [37]

$$f_{s'}(s') = \prod_i f_{s'_i}(s'_i)$$

where $f_{s'}$ is the pdf of s'. As sometimes there is no matrix W that fully satisfies the independence condition and it is difficult to maximize the independence condition above, ICA algorithms iteratively optimize a smooth function whose global optima occur when the vectors in s' are independent. The InfoMax algorithm [38] uses the entropy defined as

$$H(s') = -\int f_{s'}(s') log f_{s'}(s') ds'$$

The JADE algorithm [39] minimizes the kurtosis of $f_{s'}(s')$ while the FastICA algorithm [40] maximizes

$$J(y) \cong c[E\{G(y)\} - E\{G(v)\}]^2$$

where y is a random variable with zero mean and unit variance (white), G is a non-quadratic function, v is a Gaussian random variable and c is a positive constant. The official FastICA implementation (that we will be using) offers four choices for G. It also uses PCA to whiten the data. Thus, it is worth emphasizing that both LDA and ICA still use PCA for its desirable properties (and that's the reason we described it in more detail). For more information about the math behind ICA, one should check [36].

Can ICA be used on 2-D signals, like images? The answer is yes.

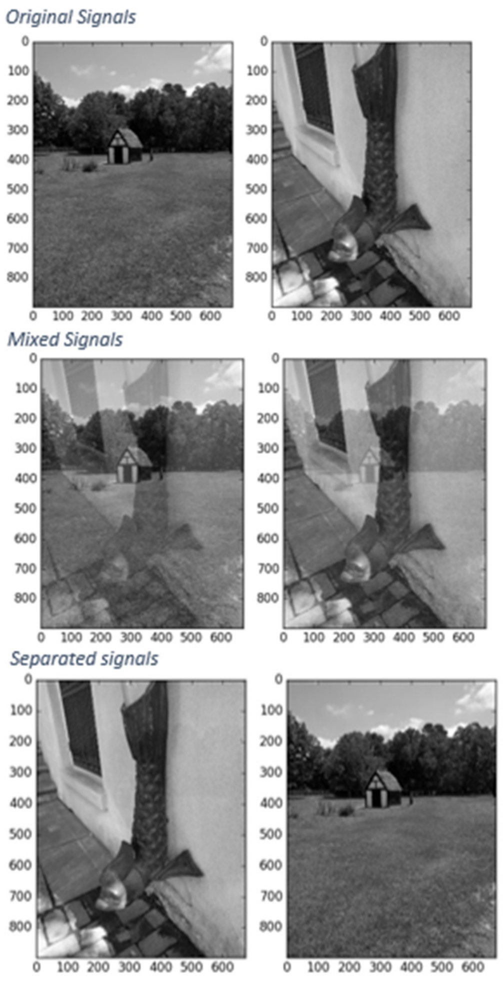


Figure 3-14: A 2-D case of ICA (JADE) [80]

But how can it be applied to face recognition? Recall that in PCA, we found a set of uncorrelated and orthogonal vectors and then projected our data onto them. ICA finds a set of independent vectors instead that can also be used for projection. Another major difference is that in PCA (and LDA), we had a way of ordering the eigenvectors (using the eigenvalues), but that doesn't apply to ICA. Instead, [41] uses a ranking metric based on between-class and within-class variance.

3.5 Classification

After applying any of the above methods on our data, we are left with a number of values per original image. These values are the features and the process of getting them is called feature extraction. We notice that we started working with 2-D images and their features were the pixels, that is, their features had an easy to interpret physical meaning. Feature extraction changes that, our features can no longer be interpreted that easily (if at all). Our goal, however, was never to just get a different interpretation, that could be achieved by using a transform like FT. What we want is to be able to distinguish between face identities (or classes, in the general case) using those features. That is called classification. More formally, classification is defined as "the action or process of classifying something according to shared qualities or characteristics"[1] .

Naturally, many algorithms have been created that handle classification. They can mostly be divided into two categories, supervised and unsupervised learning [42]. In supervised learning, a classifier first uses a-priori information about a known set of data (training) and then makes a decision about another set of data, based on that information. In face recognition, our data is feature vectors and the a-priori information is the class labels, i.e. the identity corresponding to each feature vector. A feature vector of an unknown identity is presented to the classifier and it decides in which class it fits better. When the class labels are not available, unsupervised learning takes place. The algorithm tries to cluster the feature vectors into groups based on similarities they share. The number of groups is determined by the algorithm and it corresponds to the predicted number of classes.

It is obvious that the problem of face recognition is solved through supervised learning. The methods available range from easy to implement and lightweight to really complex and computationally demanding. The majority of the bibliography examined for the methods up to this point, opts for a lightweight approach and more specifically k-NN, which we will describe in the next section.

If the bibliography favors simple solutions, why ever use the more complex ones? It's a matter of convenience and scope. When introducing or evaluating methods for feature extraction, the focus is on how the methods perform compared to others. Why waste time fine-tuning and running a heavyweight algorithm, when differences in performance can be shown more easily? Also, by avoiding fine-tuning, results are more reproducible.

3.6 k – Nearest Neighbors (k-NN)

As we mentioned above, k-NN is one of the simplest but at the same time most widespread classification algorithms. Given a set of training features and an unidentified probe feature vector, it can be described in three steps [42]:

- Identify the k closest training feature vectors to the probe vector, using a distance measure
- Find in which label most of those vectors belong
- Assign this label to the probe vector

[1] Google definition

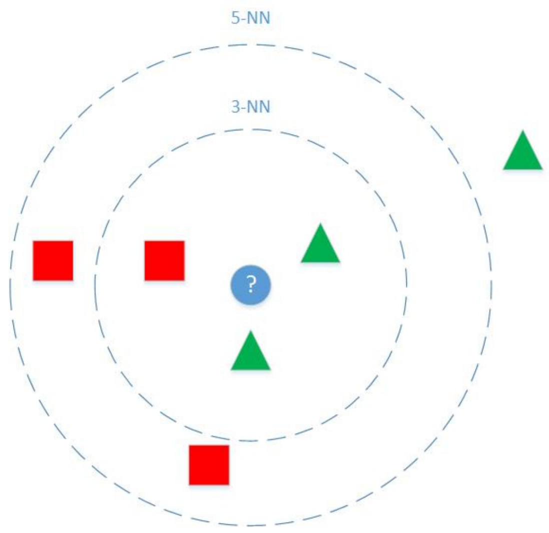


Figure 3-15: Visualization of k-NN

K is usually an even number to reduce the chance of a tie, because in those cases, the label choice is made at random. Some popular distances are:

Euclidean:

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

City block:

$$\sum_{i=1}^{n}|q_i - p_i|$$

Cosine (similarity):

$$\frac{\sum_{i=1}^{n} q_i p_I}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} p_i^2}}$$

Chebyshev:

$$\max_i(|p_i - q_i|)$$

Pearson Correlation of random variables (used in face recognition in [13]):

$$\rho(A, B) = \frac{covariance(A, B)}{\sigma_A \sigma_B}$$

In [30] a modified Mahalanobis family distance is proposed, especially for PCA-based face recognition

Mahalanobis (Yambor):

$$d(x,y) = -\sum_i \frac{1}{\sqrt{\lambda_i}} x_i y_i$$

where λ_i is the eigenvalue corresponding to the ith largest eigenvector. A reported drawback of this metric is that it underperforms for small numbers of eigenvectors.

Additionally, there are ways in which we can introduce further complexity into the algorithm, if we need to. For example, in weighted k-NN, the neighbors are given a weight so as to favor the closest ones to the probe, during the label assignment.

3.7 Evaluating a Classifier

There exist some metrics that help us quantify how well our classifier works. Accuracy (number of correct predictions over total test number), at first thought, seems to be the most important one. Isn't our goal to maximize the correct predictions? Unfortunately that is not enough. Take for example the case where a dataset has two classes, let's call them positive and negative. We can build a classifier and get a 70% accuracy. Depending on the task, that might be satisfactory. But, if the positive class takes the 80% of the dataset, we can change the classifier to a "dumb" one that makes no meaningful predictions, every class is classified as positive. The accuracy would then jump to 80% and the second classifier, although useless, would appear to be better. This is called the accuracy paradox. So, instead of just calculating the accuracy, we will also calculate two other metrics, recall and precision. First, let's define the confusion matrix.

	Predict 1	**Predict 2**	**Predict 3**	
Class 1	13	1	1	Total 1: 15
Class 2	1	14	0	Total 2: 15
Class 3	3	1	11	Total 3 : 15
	Predicted 1: 17	Predicted 2: 16	Predicted 3: 12	

The confusion matrix tells us in the main diagonal how many samples of each class were correctly labeled (true positives). In the rest of the cells in each row, it tells us how many samples of a class were given wrong predictions (false negatives). In the rest of the cells in each column, it tells us how many predictions of each class were wrong (false positives). For each class, the recall is calculated by

$$\frac{tp}{tp + fn} = \frac{tp}{Total}$$

and the precision by

$$\frac{tp}{tp + fp} = \frac{tp}{Predicted}$$

We have as many pairs of precision-recall as the number of classes in our classification problem. To simplify things a bit, because we can have a large number of classes, we take the mean of these pairs that we get one value for recall and one for precision. This is called “macro” averaging.

4 Deep Neural Networks (Deep Learning)

In the last few years, the term deep neural networks or deep learning has exploded in popularity. This can probably be attributed to the availability of more computational resources compared to the past and some very promising results in recognition problems, like the more than 10% decrease in error rate (compared to the second best algorithm at that time) in the ImageNet object recognition database in 2012 [43]. However, the foundations of shallow neural networks, or simply neural networks, were laid decades ago.

In 1957, Frank Rosenblatt [44] created the perceptron, an artificial neuron (named after the neurons in our brain). Its function was simple. Given a number of inputs, weights and a threshold, its output would be one if the summed weighted inputs were larger than the threshold, or zero if they were not. Alternatively, the output is one if the weighted input minus the threshold (now called bias) is larger than zero and zero if it is not.

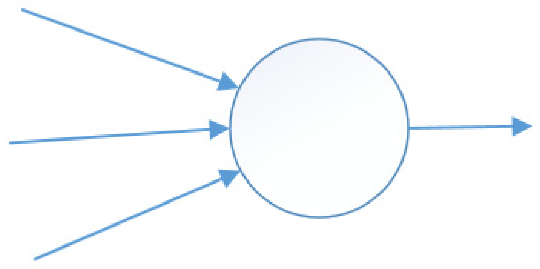


Figure 4-1: Graphical representation of a perceptron with three inputs [45]

A simple way to understand what perceptrons where supposed to do is this. Let's say you want to make a decision about going to a party. The output can be interpreted as going (one) or not going (zero). The inputs are factors affecting your decision, like the weather or the location. If the most important factor is the location, then that input will have a larger weight that the rest. A varying threshold would change the output accordingly, if it is big then most of the positive factors would need to be present for you to go. In that sense, we could say it corresponds to your general mood that day. As we will see, neurons like the perceptron can be layered to make harder decisions. We will focus on layers whose outputs always feed a next layer, not a previous one, forming networks that are called feed-forward. The "learning" part of neural networks, is finding appropriate weights and thresholds that give a desirable output. The "deep" part, refers to the number of layers. Modern deep neural networks usually have more than 10 layers, but the term technically refers to networks with two or more hidden layers, which means layers that are neither input nor output layers.

The need to learn, is the reason that eventually the perceptron was replaced. To allow learning, any small change in the weights, should produce a small change in the output. In a perceptron however, a small change can shift the output from the lowest (zero) to the highest (one) value. We need a neuron whose output is smoother. The sigmoid neuron with a bias b has output (also referred to as activation function) given by

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

Its shape should be something like the figure below

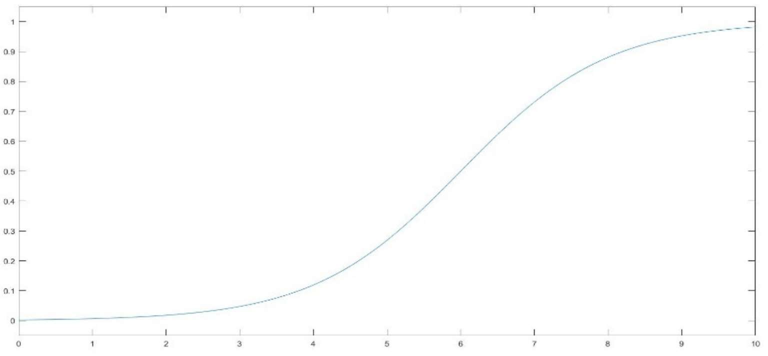


Figure 4-2: Sigmoid neuron output

Notice that it is a non-linear function. If it was linear, then the output of a neural network could be described by a linear transformation of its input. That limits the complexity and learning ability of the model. We can now take a closer look on learning, or training. Assume that we have somehow built a network that for example classifies digits.

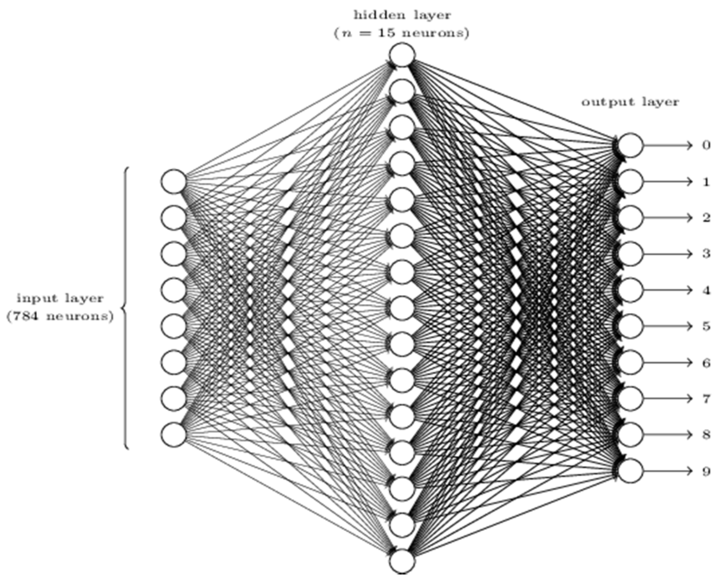


Figure 4-3: A network that can classify digits [45]

Notice how in the above network, every input neuron is connected to every neuron in the hidden layer. That type of layer is called fully connected. The output layer has ten neurons, one for each digit. When a digit (in the form of pixels) is fed through the network, we want the corresponding output to be one and the rest to be zero, ideally. First, we define a cost function like

$$C(w,b) = \frac{1}{2n}\sum_{x} \|y(x) - a\|^2$$

where a is the output of the network (expressed through weights and biases) and y(x) is the desired output for input x. Our goal is now to minimize this function. This is achieved by gradient descent. Assume that C was a function of only two variables, v1 and v2, instead of the many weights and biases. Then, C could look like this

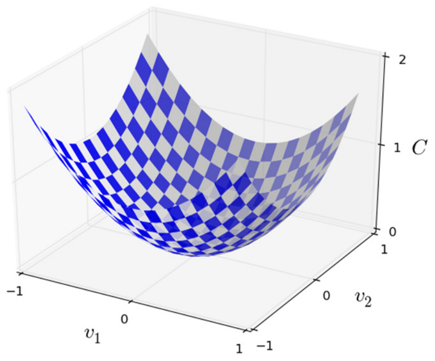


Figure 4-4: A cost function of two variables [45]

To find the global minimum we can imagine a ball rolling down the graph. The change in C for a change in v1 and v2, as the ball rolls would be

$$\Delta C \approx \frac{dC}{dv1}\Delta v1 + \frac{dC}{dv2}\Delta v2$$

If we manage to find changes in v1 and v2 so that the change in C is negative, then that means that the ball rolls down, towards the minimum. If we rewrite the above equation to

$$\Delta C \approx \nabla C.\Delta v$$

where ∇C is the gradient of C, then the change in direction Δv can be proven to be

$$\Delta v = -\eta \Delta C$$

where η is a small, user defined parameter called learning rate. In practice, because the number of weights is generally enormous and training requires many samples, instead of computing Δv for each training sample and then updating the weights, a variation is used. In stochastic gradient descent, the gradient is computed for a number of random samples, instead of just one. The update to the weights is performed by averaging over this sample set, called batch, instead for each sample individually. A common term in neural networks, backpropagation, is the way in which we compute the gradient. We will not go into detail about how it works, but one can check the resource most frequently referenced in this chapter, Michael Nielsen's introductory webpage to neural networks [45].

4.1 Improvements on the methodology

Sigmoid neurons are not perfect. Their output gets almost flat when it is near the minimum or the maximum. That means that the changes in the cost function will be very small during training, if we fall into in these regions. This can be solved by using a different cost function, called cross-entropy. It is defined by

$$C = -\frac{1}{n}\sum_{x}\sum_{j}\left[y_j ln a_j + (1 - y_j)\ln(1 - a_j)\right]$$

where j sums over all the output neurons. The reasons why this change is beneficial are not obvious, but we will not delve deeper.

We can also help the network avoid these regions during initialization. Before training starts, the weights and biases need to have some initial values. Instead of just picking any random values, using a Gaussian distribution with mean zero and standard deviation $1/\sqrt{n}$ where n is the number of inputs in a neuron

Another improvement on the cost function, is called regularization. It is an extra term that gets added and can look like this

$$\frac{\lambda}{2n}\sum_{w} w^2$$

where λ is a positive user defined regularization parameter. What this does, is make the network prefer smaller weights when training and it has been shown to improve the recognition rate by helping against overfitting. Overfitting is a term used in machine learning algorithms and describes the phenomenon where these algorithms either memorize patterns or get affected by noisy samples, instead of learning general characteristics of the data. That can lead to high success rates on the training but poor results in the test set.

4.2 Popular Hidden Layer Architectures

We mentioned that neurons can be layered in different ways. Over the years, specific layer designs have shown desirable properties and have taken the field of neural networks to the next level.

Convolutional layers, are the MVP of image processing. Given a 2-D image, convolutional layers mimic the convolution of a kernel filter across it, creating a filtered output called feature map. An n x n region of the image is connected with weights to a neuron in the convolutional layer. These weights and the bias are the same for the whole feature map and represent the kernel (parameter sharing). In practice, convolutional layers have many kernels/filters and compute a feature map for each one.

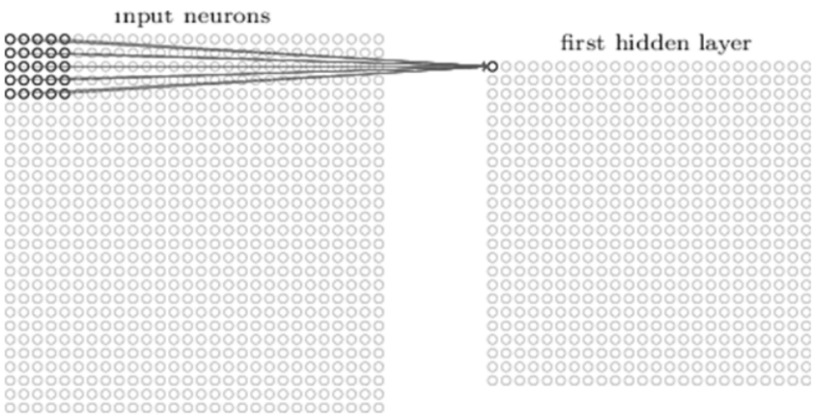


Figure 4-5: How a convolutional layer works [87]

Additionally, the kernels can be three-dimensional with each 2-D kernel part handling a different 2-D part of the input. The values calculated by the 2-D parts of a kernel are aggregated to produce the final kernel output. The example below shows a layer with two 3-D kernels acting upon a 3-D input (RGB channels of an image).

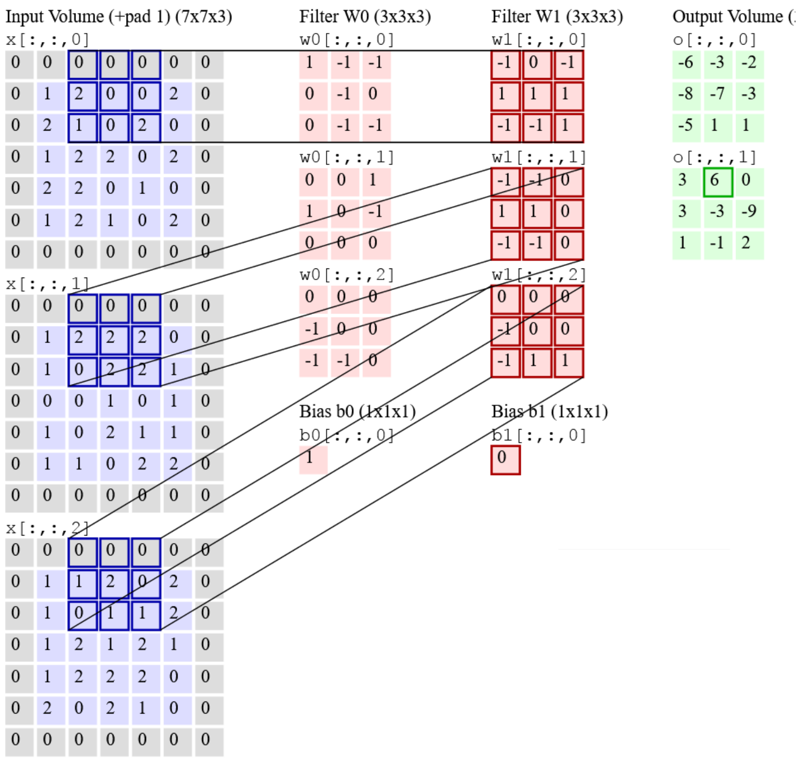


Figure 4-6: Convolution with 3-D filters [88]

Generally, a convolutional layer can be defined using four parameters. The number of filters, their size (n x n x m), the stride which defines the step of the “convolution” (one for classic convolution, higher values skip pixels) and the type of padding used (zero padding in the above example). The size of its output is dependent on those parameters.

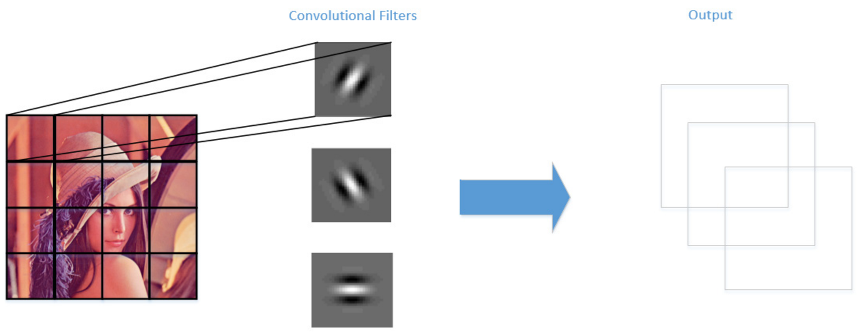


Figure 4-7: Block diagram of convolution

The networks that utilize them are called Convolutional Neural Networks (CNNs). The first successful convolutional network was LeNet-5 [46], proposed in 1998 and applied to the task of digit recognition.In that network, the convolutional layers don't have activation functions. They are followed by averaging subsampling, that reduces the dimensionality of the feature maps. In the same paper it is theorized that the combination of connecting nearby pixels (local receptive fields), parameter sharing and subsampling can help achieve some degree of shift, scale and distortion invariance.

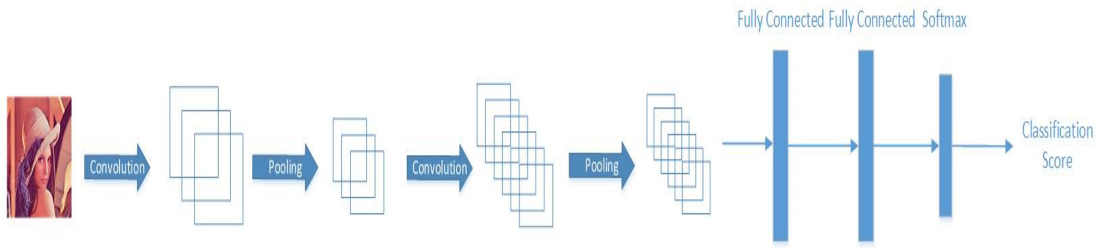


Figure 4-8: Typical modern convolutional architecture

It turns out that the weights of the first layer, when visualized, often do resemble real filters. This is an interesting property, as it is shared among most (if not all) deep convolutional networks, regardless of architecture and target task (e.g. object recognition, face recognition)

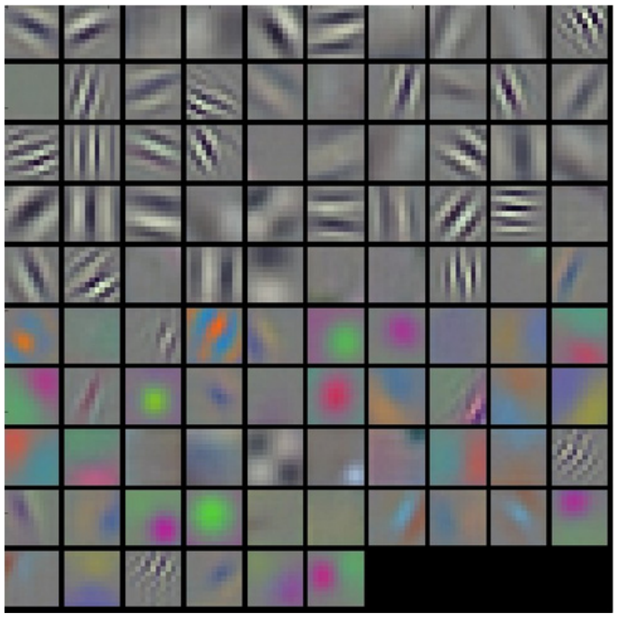


Figure 4-9:AlexNet convolution filters [88]

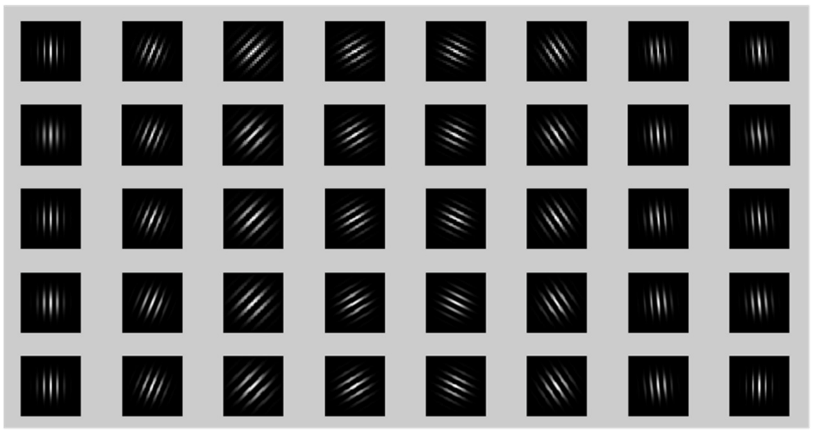


Figure 4-10: Gabor filters

Modern deep convolutional networks are used in conjunction with an activation function. This is handled by a rectification layer (ReLU) which is a layer comprised of neurons whose output is calculated by the function y=max (0,x). They are designed so as to not alter the input dimensionality. They are reported to also speed up training when used [33].

A layer that goes hand in hand with convolutional layers and performs a form of subsampling, is the pooling layer which generally performs max-pooling. It is connected in the same way as convolutional layers, but its outputs are the max values of an m x m window, instead of the average value like in LeNet5..

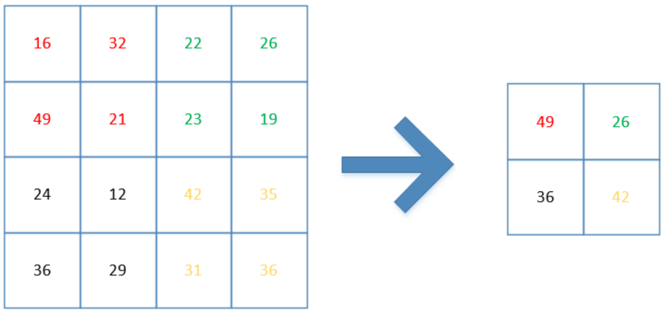


Figure 4-11: How a max-pool layer works

Its main goal is to reduce the high dimensionality of the data and help with invariance, but some researchers suggest that they may not be necessary [47]. Invariance is achieved by discarding the values where the filters' outputs were small, which means they didn't detect the trained pattern in that area. At the final level of pooling, many areas are discarded, keeping mostly the detected patterns, regardless of their original spatial position.

Dropout layers, are another method to guard against overfitting. During training, some neurons in a fully-connected layer, different each time, are temporarily disabled.

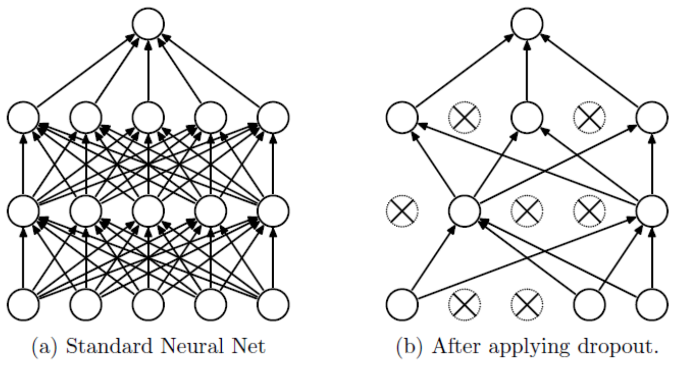


Figure 4-12: Example of dropout [48]

They can make the overall features learned more robust, as they reduce co-adaptation between neurons [48]. Each neuron is forced to learn “good” features by itself, without relying as much on others.

Last but not least, the Softmax layer seen in Figure 4.7 is often used as the output layer of a network. It is a fully connected layer, meaning that every input is connected to every neuron, i.e. the input to the activation function of each neuron can be described by $\sum w_i x_i - b$. Its size (number of neurons) is the same as the number of classes in our classification problem. The output values always sum to one (using the softmax function), so they can be viewed as the probabilities of an input belonging to a certain class. This means that classification in handled within the network, there is no need for external classifiers.

Typically, a modern convolutional neural network will have multiple blocks of a convolutional, ReLU and max-pooling layer connected in a series, followed by a smaller number of fully connected layers, the last of whom is a softmax.

Due to the fact that in deep networks a convolutional layer is almost always used with a ReLU activation, the term convolutional layers most of the time also includes the activation.

4.3 Training a Deep Neural Network

Let’s assume that we have constructed a deep neural network, it’s now time to train it. This can be a daunting task. Apart from the structural parameters, like the weights or the filter sizes, we also need to optimize the so-called hyperparameters of our network. These include the number of epochs, i.e. how many times the data set go through the network, the batch size, learning rate, momentum and possibly others.

Training may also involve data augmentation. The training set is augmented by adding cropped and/or rotated copies of the original images to boost generalization ability. A more advanced technique is triplet loss, where the last layer of the networks is trained using an “anchor” image, one positive and one negative example of classification [49].

The structural parameters are learned using the training set. Afterwards, using a smaller and different set than the training one, called validation set, the hyperparameters are tuned. This is done to combat overfitting.

This may not sound so bad. After all, there exist other complex techniques with hyperparameters. That may be true, but in the case of deep neural networks, the more challenging part can be training. State of the art architectures take many days to complete training on multiple high-end GPUs [49] [50]. That is because of the high number of parameters that need to be learned and the consequently large datasets required to do so. For example, [50] has more than 120 million parameters [51].

It is, therefore, apparent that training a state of the art deep network can’t be reasonably accomplished by the common user.

4.4 Designing a Deep Neural Network

Wait, shouldn’t this section precede the last one? We need to have a network design first in order to train it. That is correct. However, describing how difficult can it be to train a network helps us understand why it also difficult to design one.

We know how each layer works independently, more or less. So, designing a network using these layers shouldn’t be much of a problem, right? Unfortunately, it is. The field is still very new and under intensive study. While there exist networks that achieve record-setting results, there aren’t many insights provided about how or why they work better than others. They just do. In modern bibliography, architectures are presented without any mention on architectures that did not perform as well and possible explanations.

In [51] this problem is acknowledged. They report some inferior architectures they came up and described how they handled the design process. However, most of their work is based on trial and error. They began from a small convolutional network and kept adding neurons and layers until the results were unsatisfactory. It is safe to assume that a large part of the work on deep neural networks is heavily based on trial and error and of course, experience.

The above might sound a bit grim but there is a silver lining. As the popularity of CNNs rose, so did the interest to understand them. Visualization techniques such as t-distributed stochastic neighbor embedding (t-SNE) [52] and especially deconvolution (a reverse network that produces pixels from feature maps) [53] can help us visualize what a network has learned. They play will certainly keep playing an important role in deep learning.

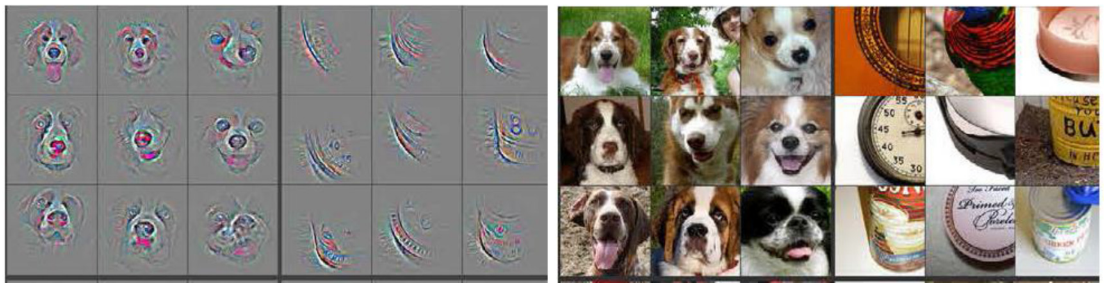


Figure 4-13: Deconvolution example [53]

Given the heavy computational costs of training mentioned in the previous section, designing a new deep neural network as a part of an undergraduate thesis seems to be very unapproachable.

Does that mean that we should leave deep learning to researchers for now? Fortunately, there is another approach to deep learning that was employed in this thesis, and it can be used by anyone.

4.5 Transfer Learning

Speaking broadly, transfer learning is the use (transfer) of resources from other tasks to improve prediction accuracy on a given task. Multitask learning [54] was an early form of transfer learning applied on shallow neural networks. The main idea behind it, is that training for additional similar tasks can boost the performance on our original task.

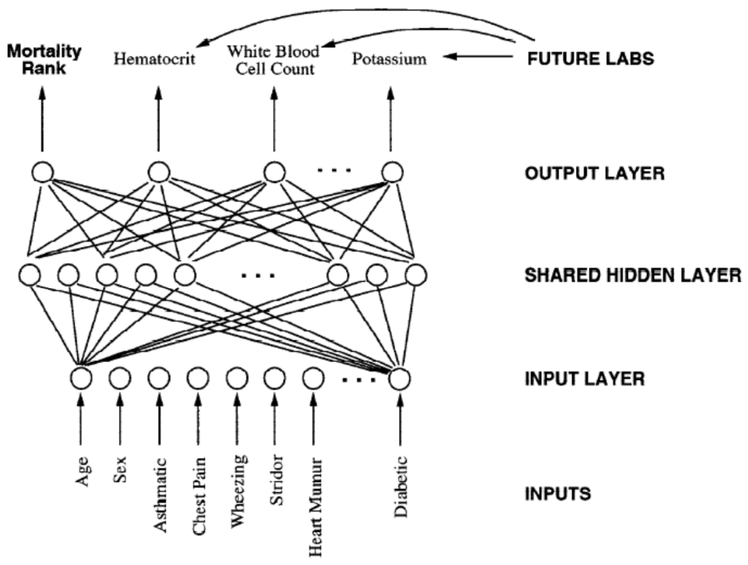


Figure 4-14: Multitask Learning example [54]

In the example above, the original goal is to predict the mortality rank of a patient based on his physical characteristics and symptoms. When training a bigger network for more related predictions, like the results of certain blood tests, however, it was found that the originally wanted prediction became more accurate.

In 2007 the idea of self-taught learning [55] was introduced. The authors theorized that "…many randomly downloaded images will contain basic visual patterns (such as edges) that are similar to those in images of elephants and rhinos. If, therefore, we can learn to recognize such patterns from the unlabeled data, these patterns can be used for the supervised learning task of interest, such as recognizing elephants and rhinos". They projected their original images with bases generated from an optimization problem on the foreign data, with promising results.

In 2008, multitask learning was applied to shallow CNNs [56].

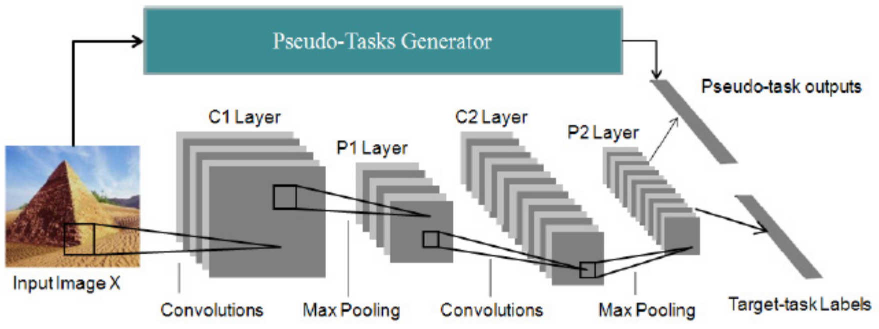


Figure 4-15: Multitask learning in shallow CNNs [56]

In this case, the related tasks are generated from our input images by selecting a random patch from the input and using it to filter the whole image.

It is obvious that there isn't just one way to implement transfer learning and the term doesn't refer to just one technique. So, what does transfer learning mean in the context of deep CNNs?

First of all, one could consider data augmentation as a form of transfer learning however the term generally refers to transferring (or more accurately, reusing) trained networks to other similar or dissimilar tasks. Soon after deep CNNs, rose into prominence, studies were made to test if and how well trained networks handle new tasks and datasets.

Naturally, some changes need to be made to the network. The last (softmax) layer generally can't be kept, as its output is seen just as probabilities for an input to belong to a class in the original problem. We can either add a few fully connected layers or just a softmax layer tailored to our data and retrain the network, or simply take the output of one of the previous fully connected layers and use it as features in a classifier (SVM seem to be very popular). Note that the outputs of convolutional layers are never used directly due to their number and dimensionality, they also tend to underperform [57]. For example, in the network we ended up using, the output of the last convolutional layer has 7 x 7 x 512 = 25088 neurons, while the fully connected layers that follows it, only 4096. Retraining only one or few layers is not only computationally easier than training a whole network, but also allows the use of deep learning on smaller datasets, as we need less data to prevent overfitting due to the smaller number of parameters. This is very important, as deep learning can be used in more practical or rare applications, where the number of available labelled data is limited. So, transfer learning makes deep learning more approachable, but how well does it work? Numerous papers have been published, testing state-of-the-art pre-trained networks on various tasks [53] [58] [59] [60]. Two factors are common among these publications. They all use networks pre-trained on object recognition and they all get amazing results. The choice of object recognition is not unexpected. It is a very general and difficult task and it is logical to think that a network trained on a more general task could transfer better. As we can see below, object recognition covers a wide range of objects and animals. The complexity of the task is evidently very high.

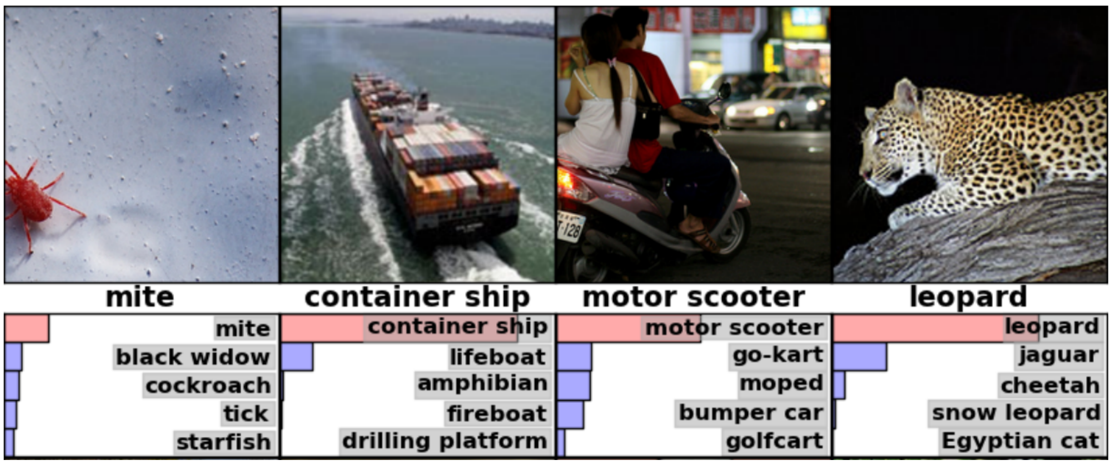


Figure 4-16: Typical object recognition images [61]

Other than the source task of the transferred network, there are not much else to worry about. Due to data augmentation, any network can be used, regardless of minor differences between its input size and our dataset. Some selected results are presented below.

Source Network	**Target Task**	**Previous state-of-the-art**	**Transfer Learning**
Overfeat **[62]** + augmentation	Scene classification	64%	69%
Overfeat + augmentation	Fine-grained recognition	56.8%	61.8%
AlexNet **[63]**	Fine-grained recognition	56.8%	58.75%
AlexNet	Object recognition	40.5%	65.7%

For more detailed results, one should read the papers mentioned above. Transferred networks can not only beat (or come close to) the previous state-of-the-art in the same domain but also on very different domains than the one the network was initially trained for. Of course, the closest the original dataset is to the new one, the better the results [57]. Unfortunately, comparisons among the networks used regarding their transferability cannot be made as the researchers used mostly different datasets and tricks (e.g. data augmentation). The same goes for the comparison between retrained layers vs no retraining.

A more in depth study on transfer learning was done in [64]. The researchers built two identical networks with 8 convolutional layers. The networks were trained on two random (A & B) but similar splits of an object recognition database. They then defined different transfer scenarios. In BnB, n layers are transferred from the network trained on B. These layers are locked (not trained) and the rest of the network is trained on split B. In BnB+, the whole network is retrained. AnA and AnA+ are the same but for split A. AnB, BnA, AnB+ and BnA+ test the transfer between different splits. Their results are summarized below.

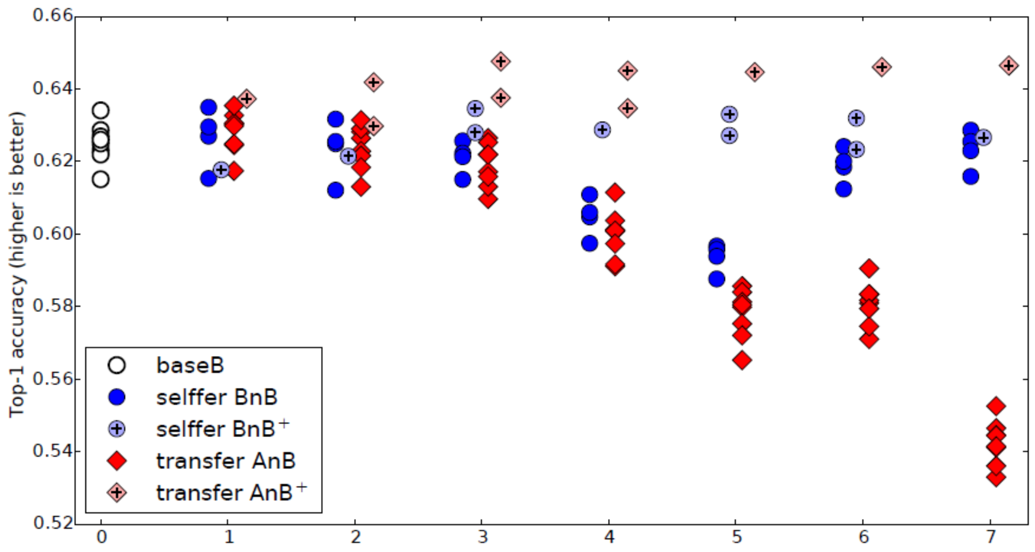


Figure 4-17: Transfer results [64]

We will put emphasis on two significant results. First, the drop in accuracy for n=4 and the rise for n=6, suggest that convolutional layers deeper in the network, may form co-adaptions, meaning that the feature maps they produce are more dependent on each other than in earlier layers and that they should be retrained or kept as a whole. Second, it appears that a network trained on a similar dataset and then trained again on the original dataset it was supposed to be trained on, achieves better accuracy. That implies that transfer learning is not only beneficial when there is a restriction of resources (computational or lack of enough data) but also to regular deep learning applications.

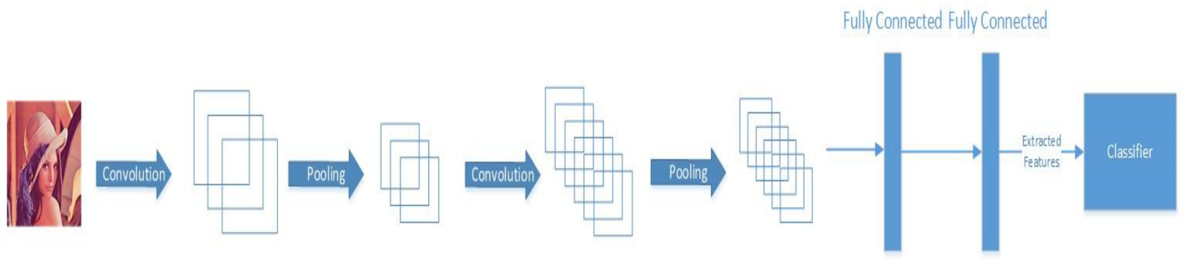
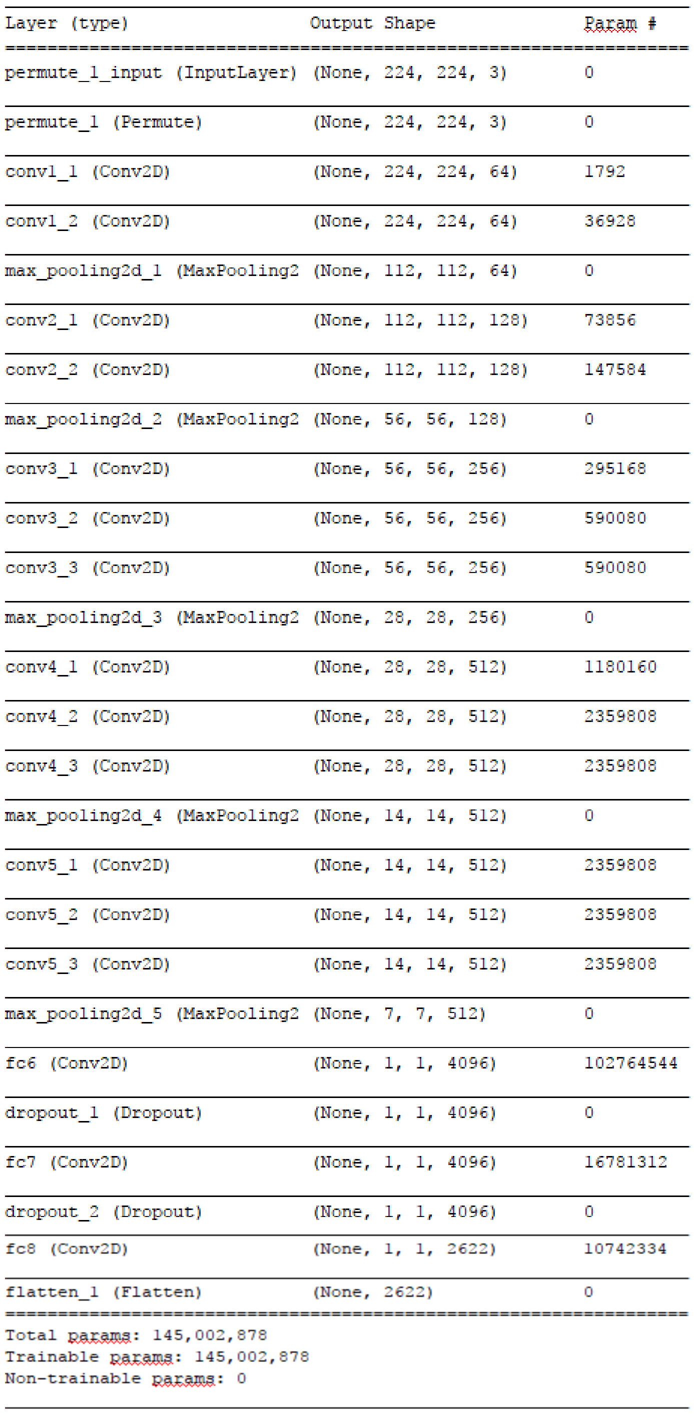


Figure 4-18: Transfer learning scheme

4.6 Visual Geometry Group - Face (VGG-Face)

VGG-Face is a deep convolutional neural network developed in the University of Oxford in 2015 [49]. It is comprised of 23 hidden layers (excluding Keras-only layers) and a softmax output layer. The Keras [65] breakdown of the model (minus the softmax layer) can be seen below. The activation function used in all layers is ReLU.

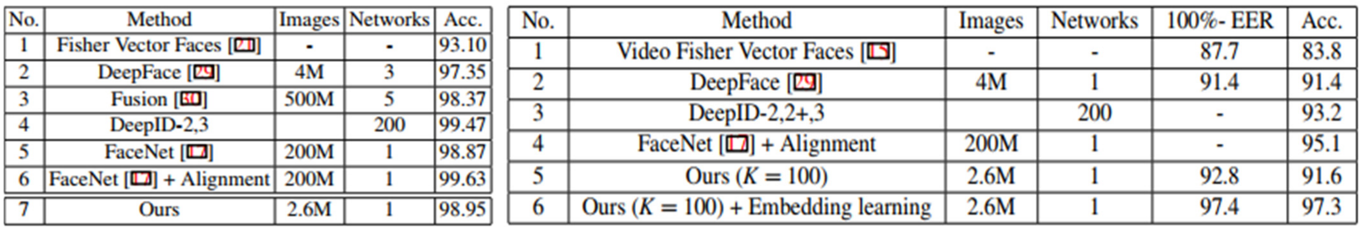


```
Layer (type)                 Output Shape              Param #
=================================================================
permute_1_input (InputLayer) (None, 224, 224, 3)       0
_________________________________________________________________
permute_1 (Permute)          (None, 224, 224, 3)       0
_________________________________________________________________
conv1_1 (Conv2D)             (None, 224, 224, 64)      1792
_________________________________________________________________
conv1_2 (Conv2D)             (None, 224, 224, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 112, 112, 64)      0
_________________________________________________________________
conv2_1 (Conv2D)             (None, 112, 112, 128)     73856
_________________________________________________________________
conv2_2 (Conv2D)             (None, 112, 112, 128)     147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 56, 56, 128)       0
_________________________________________________________________
conv3_1 (Conv2D)             (None, 56, 56, 256)       295168
_________________________________________________________________
conv3_2 (Conv2D)             (None, 56, 56, 256)       590080
_________________________________________________________________
conv3_3 (Conv2D)             (None, 56, 56, 256)       590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 28, 28, 256)       0
_________________________________________________________________
conv4_1 (Conv2D)             (None, 28, 28, 512)       1180160
_________________________________________________________________
conv4_2 (Conv2D)             (None, 28, 28, 512)       2359808
_________________________________________________________________
conv4_3 (Conv2D)             (None, 28, 28, 512)       2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 512)       0
_________________________________________________________________
conv5_1 (Conv2D)             (None, 14, 14, 512)       2359808
_________________________________________________________________
conv5_2 (Conv2D)             (None, 14, 14, 512)       2359808
_________________________________________________________________
conv5_3 (Conv2D)             (None, 14, 14, 512)       2359808
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
fc6 (Conv2D)                 (None, 1, 1, 4096)        102764544
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 1, 4096)        0
_________________________________________________________________
fc7 (Conv2D)                 (None, 1, 1, 4096)        16781312
_________________________________________________________________
dropout_2 (Dropout)          (None, 1, 1, 4096)        0
_________________________________________________________________
fc8 (Conv2D)                 (None, 1, 1, 2622)        10742334
_________________________________________________________________
flatten_1 (Flatten)          (None, 2622)              0
=================================================================
Total params: 145,002,878
Trainable params: 145,002,878
Non-trainable params: 0
_________________________________________________________________
```

Figure 4-19: Breakdown of the VGG-Face CNN

We note two important things. First, the network takes images of resolution 224 x 224 (50176 total pixels) extracts 2622 features per image. That's a considerable dimensionality reduction. Second, the network has more than 145 million parameters! That imposes considerable restrictions to training time and training set size. The researchers created a dataset of 2.6 million images and used data augmentation to further increase its size during training and implemented triplet loss. As far as hardware is concerned, 4 very high end GPUs were used (NVIDIA Titan Black).

At the time of its release, it managed very competitive performance on the LFW benchmark and outperformed every other method on the Youtube Faces Dataset [66]



No.	Method	Images	Networks	Acc.
1	Fisher Vector Faces [21]	-	-	93.10
2	DeepFace [29]	4M	3	97.35
3	Fusion [30]	500M	5	98.37
4	DeepID-2,3		200	99.47
5	FaceNet [17]	200M	1	98.87
6	FaceNet [17] + Alignment	200M	1	99.63
7	Ours	2.6M	1	98.95

No.	Method	Images	Networks	100%- EER	Acc.
1	Video Fisher Vector Faces [15]	-	-	87.7	83.8
2	DeepFace [29]	4M	1	91.4	91.4
3	DeepID-2,2+,3		200	-	93.2
4	FaceNet [17] + Alignment	200M	1	-	95.1
5	Ours ($K = 100$)	2.6M	1	92.8	91.6
6	Ours ($K = 100$) + Embedding learning	2.6M	1	97.4	97.3

Figure 4-20: VGG-Face results [49]

.

5 Proposed methodology and Technical Details

In the following sections, we will propose methodologies and implement the above algorithms for the task of face recognition, primarily on the ORL database. For transfer learning, Keras was used. Keras is a python library for deep learning and works as an intermediate for other very popular libraries like Tensorflow, CNTK [67] and Theano [68]. The code written with Keras is independent of the backend library, meaning that the same Keras code can be run regardless of chosen backend. Switching from running on the GPU to the CPU and vice versa is also very easy. As such, Keras is an ideal choice for prototyping. For the rest, the MATLAB environment was used with the appropriate toolboxes. It should be noted that the MATLAB toolbox for CNNs was also tested, but required more memory than the Keras implementation, making it impossible to run with our resources (8 GB of memory).

5.1 Classic methods

Based on the presented statistical methods, we propose to use combinations of wavelet decomposition, feature extraction, clustering of features and classification in the following form. First, DWT will be implemented up to a desirable level that is task specific, for feature extraction. DWT provides dimensionality reduction through subsampling and low-pass filtering improves the quality of the image by removing noise and improves the properties of the image histogram. As far as the wavelet choice is concerned, we will be using the db4 wavelet on the LL sub-band, as proposed in [13]. In each repetition, a range of chosen eigenvalues is used, in steps of 10. A statistical method is then applied to the decomposed features in the LL sub-band, to further reduce dimensionality, discard redundant information and reorganize the features. PCA provides an orthogonal basis that maximizes total variance, LDA maximizes between-class variance and minimizes within-class variance and ICA uses higher order statistics to achieve statistical independency of the basis. This process creates features of low dimensionality and appropriate for use in a classifier, which is trained using labeled data and then assigns labels to unlabeled data. We make the assumptions that the decomposed features perform better in classification purposes and that the statistical reorganization of these features results in a further boost in recognition accuracy

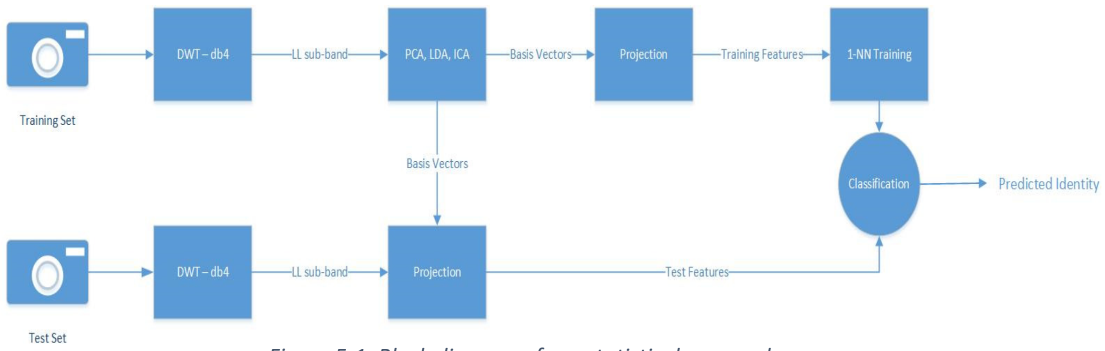


Figure 5-1: Block diagram of our statistical approach

5.2 Experimentation on Deep Learning

Instead of training the entire deep neural network, which requires the formation of a big dataset of images, we will experiment on the concept of transfer learning, which assumes that a network trained on such a big dataset covering the peculiarities of many application areas can also address the feature needs of a specific face recognition application. The chosen pre-trained network will be treated as a black box that provides the relevant features for our own image dataset. These features will be then used to train a k-NN classifier outside of the network. There is an interest to this approach, as it enables the use of deep learning architectures to fields were the size of available data is small (an example could be a rare animal or disease case) or in cases were the computational resources available do not suffice to support the taxing process of training, as the cost of just classification is many times smaller.

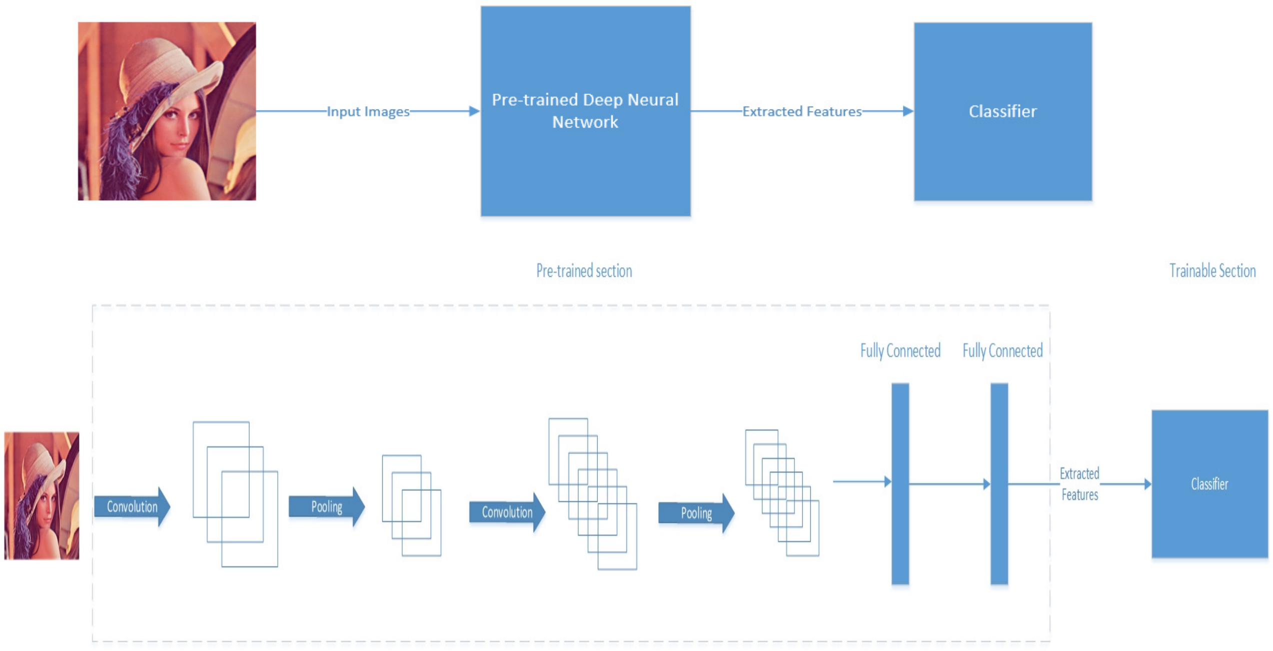


Figure 5-2: Transfer Learning approach

5.3 Validation

In every scenario, we choose a random 70/30 dataset split for training and testing. We do not impose a restriction on a minimum or maximum number of pictures per person. That means that there can be splits where a person is not present in the training or test set or is heavily under/overrepresented. The “experiments” are run 128 times and the mean accuracy, standard deviation (presented only for the top scores), recall and precision are calculated. The multiple runs and data splits are performed to ensure the generalizability of our results. 1-NN is used for classification, in all cases apart from transfer learning.

To summarize, our proposed methodology for feature extraction, using DWT and statistical methods is:

1. Split the dataset into testing and training subsets using a 70/30 random split
2. Apply DWT using the db4 wavelet function on both sets, to generate four level 1 wavelet bands per image
3. If needed, keep applying DWT on the highest level LL band
4. Discard all other bands apart from the last LL band

5. Use a statistical method on the training set to calculate a new basis for the data
6. Project all the data using an increasing number of basis vectors (steps of 10)
7. Train a k-NN classifier with the training data
8. Prompt the classifier with the test data
9. Calculate the accuracy, precision and recall
10. Repeat 128 times and calculate the standard deviation

In the case of transfer learning, the network replaces steps 2 through 6 in the proposed methodology with statistical methods implementing feature mapping in orthogonal or independent directions, as well as feature reduction.

As a baseline, the “naive” approach of not applying any transformation to our ORL images and just calculating the distance between the original pixels instead of features, yields an accuracy of 93.1% using Pearson Correlation, standard deviation of 0.02%, recall of 94% and precision of 94.5%, using 112 x 92 = 10304 features per image. These results are indicative of the database’s good characteristics (aligned faces, same background, same scale etc.). The use of transformations induces further improvement on the classification accuracy.

The next section indicates the classification accuracy that can be achieved by the various modules of processing operations proposed in this thesis.

6 Evaluation

6.1 PCA

First, PCA is applied without DWT. The distance measures mentioned in [30] [13] are used to check how much accuracy can be affected by the choice of distance metric. Recall and precision are only reported for the top performing metrics.

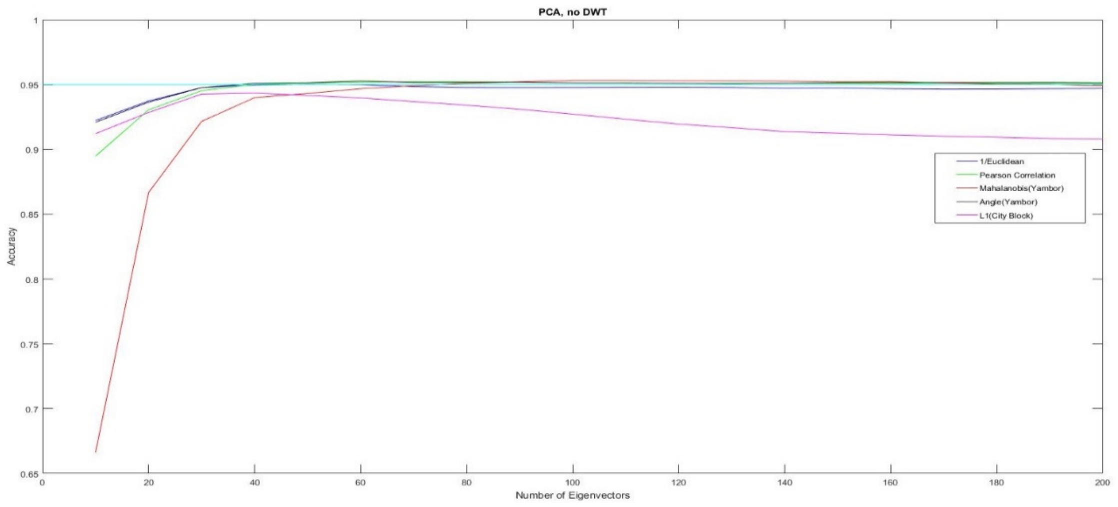


Figure 6-1: PCA, no DWT, accuracy

The cyan horizontal line is provided for comparison purposes. At a first glance, we are able to verify that Mahalanobis distance severely underperforms for small numbers of eigenvectors. This could be attributed to the fact that the first eigenvectors can be several times bigger than the rest. Using only large values in the Mahalanobis distance, shrinks the features and makes them more prone to noise. By taking a closer look, we see that the maximum accuracy is given by Mahalanobis distance, 95.33% at 100 eigenvectors/features. The second best accuracy is 95.2% at 80 features, given by Pearson Correlation. Those metrics not only score the highest accuracy but also low standard deviation.

At 100 features, the standard deviation of Mahalanobis distance is 2.20% and 2.13% for Pearson Correlation at 80.

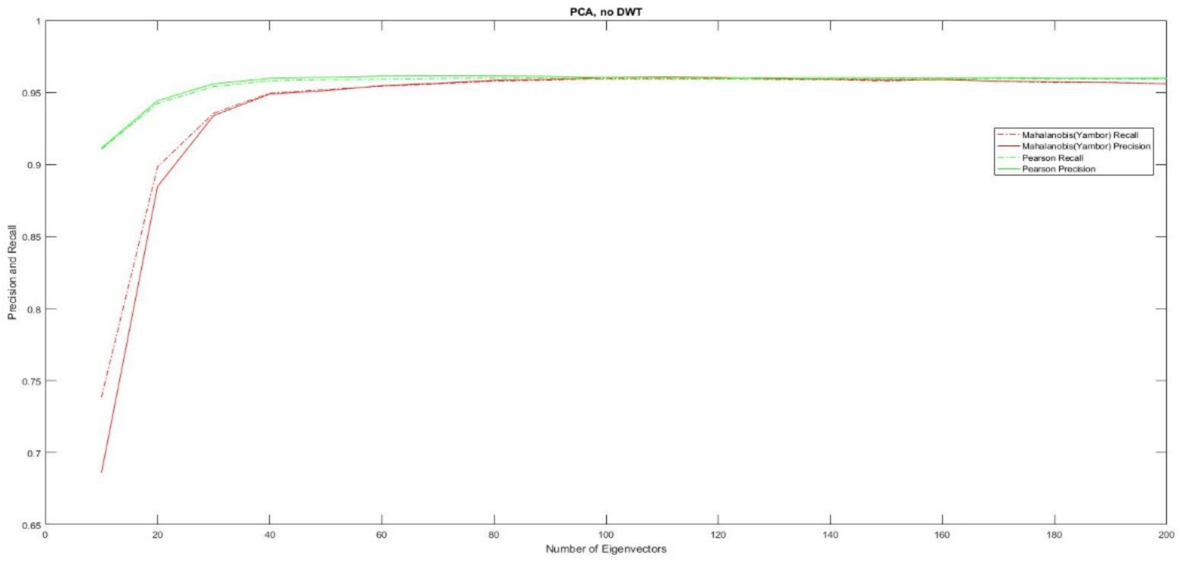
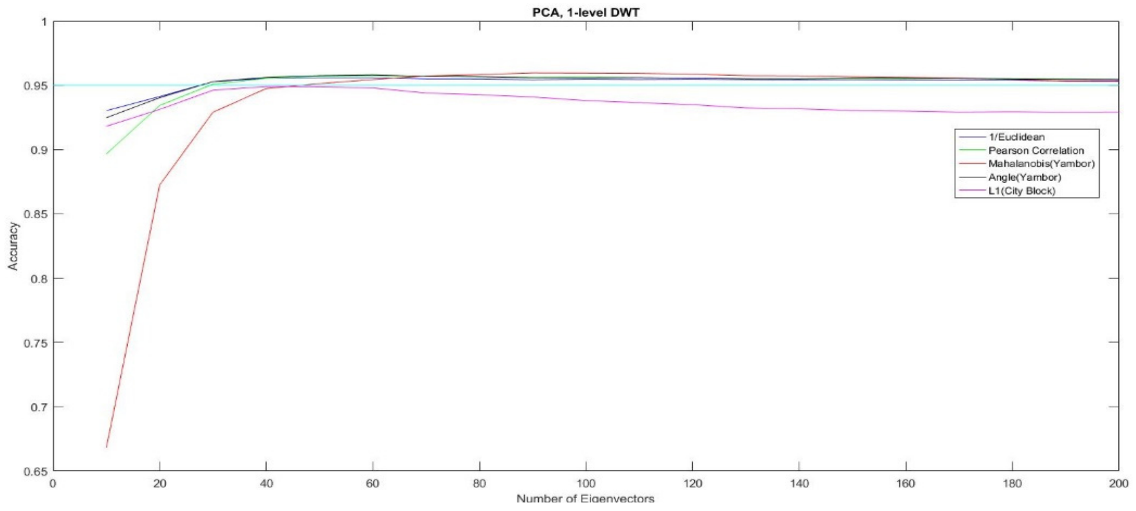
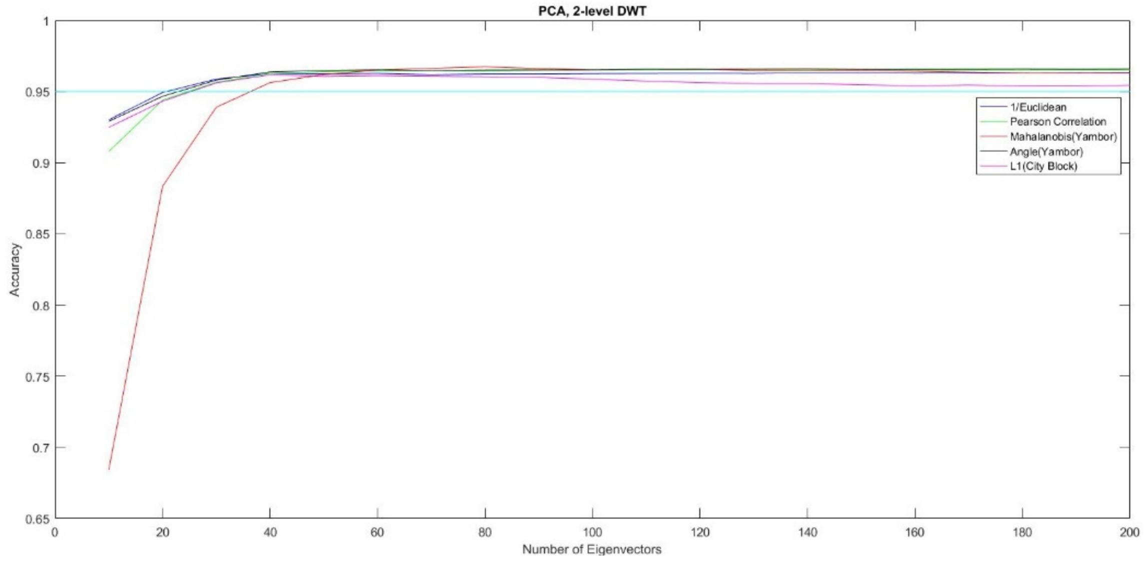
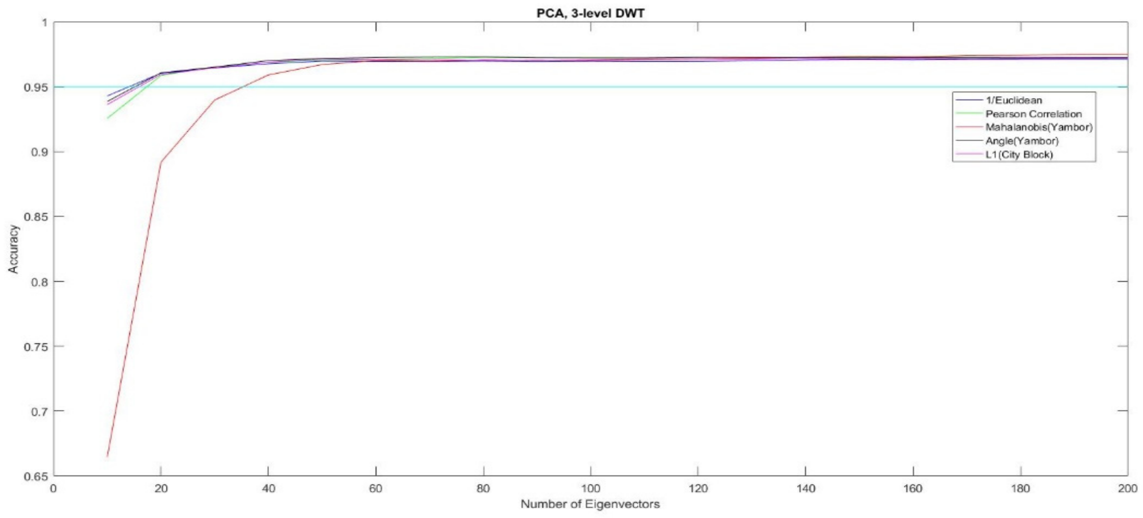


Figure 6-2: PCA, no DWT, recall and precision

Precision and recall also score highly, at 96.03% and 95.9% for Mahalanobis and 96.13% and 96% for Pearson Correlation.

What can be noted is that after a metric reaches its peak, the accuracy, precision and recall plateau, and they vary minimally for each following batch of 10 eigenvectors. That is why we omit results past the 200 eigenvector mark, as the plots are already not very easy to read. This doesn't come as a surprise. As we showed earlier, the first/highest eigenvalue-eigenvector pairs contribute the most variance in the data. The eigenvectors that follow offer lower and lower variance and thus affect classification in a small manner. The biggest deviation in accuracy between distances in the plateau region is at 100 eigenvectors where the lowest accuracy is 94.77% and the highest is 95.3%, if we ignore Citiblock distance.

Next, DWT is applied for levels one through four, using the db4 wavelet.


PCA, 1-level DWT
Accuracy
Number of Eigenvectors
1/Euclidean
Pearson Correlation
Mahalanobis(Yambor)
Angle(Yambor)
L1(City Block)

PCA, 2-level DWT
Accuracy
Number of Eigenvectors
1/Euclidean
Pearson Correlation
Mahalanobis(Yambor)
Angle(Yambor)
L1(City Block)

PCA, 3-level DWT
Accuracy
Number of Eigenvectors
1/Euclidean
Pearson Correlation
Mahalanobis(Yambor)
Angle(Yambor)
L1(City Block)

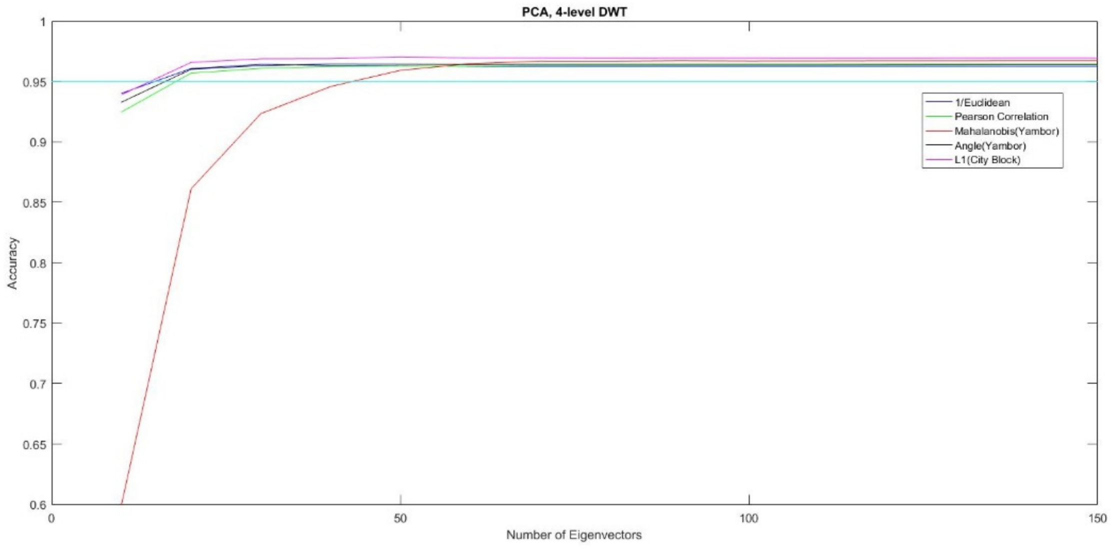
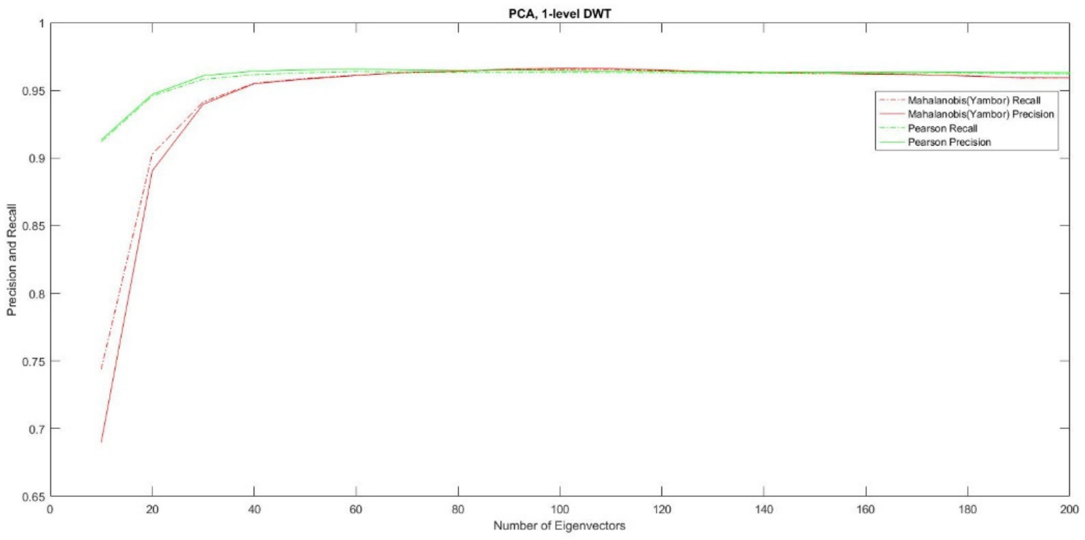


Figure 6-3:PCA, DWT levels 1-4, accuracy

DWT affected accuracy in a positive manner. Mahalanobis and Pearson Correlation still perform very well, with the exception of level 4, where the accuracy decreased from level 3. This does not agree with [13], where level 4 outperforms level 3. However, it can be explained by the fact that we use images with resolution 112 x 92 instead of 128 x 128 that they used. After a certain level, dependent on the image size, decimation can remove too much information, making the task of recognition harder. Maximum accuracy (97.3%) is achieved at 80 eigenvectors using Pearson Correlation or Angle distance and level 3 DWT. Mahalanobis distance follows with an accuracy of 97.07%. Another positive results is that the difference between the metrics, even if we consider Citiblock, is reduced to 0.32% at level 3 DWT.



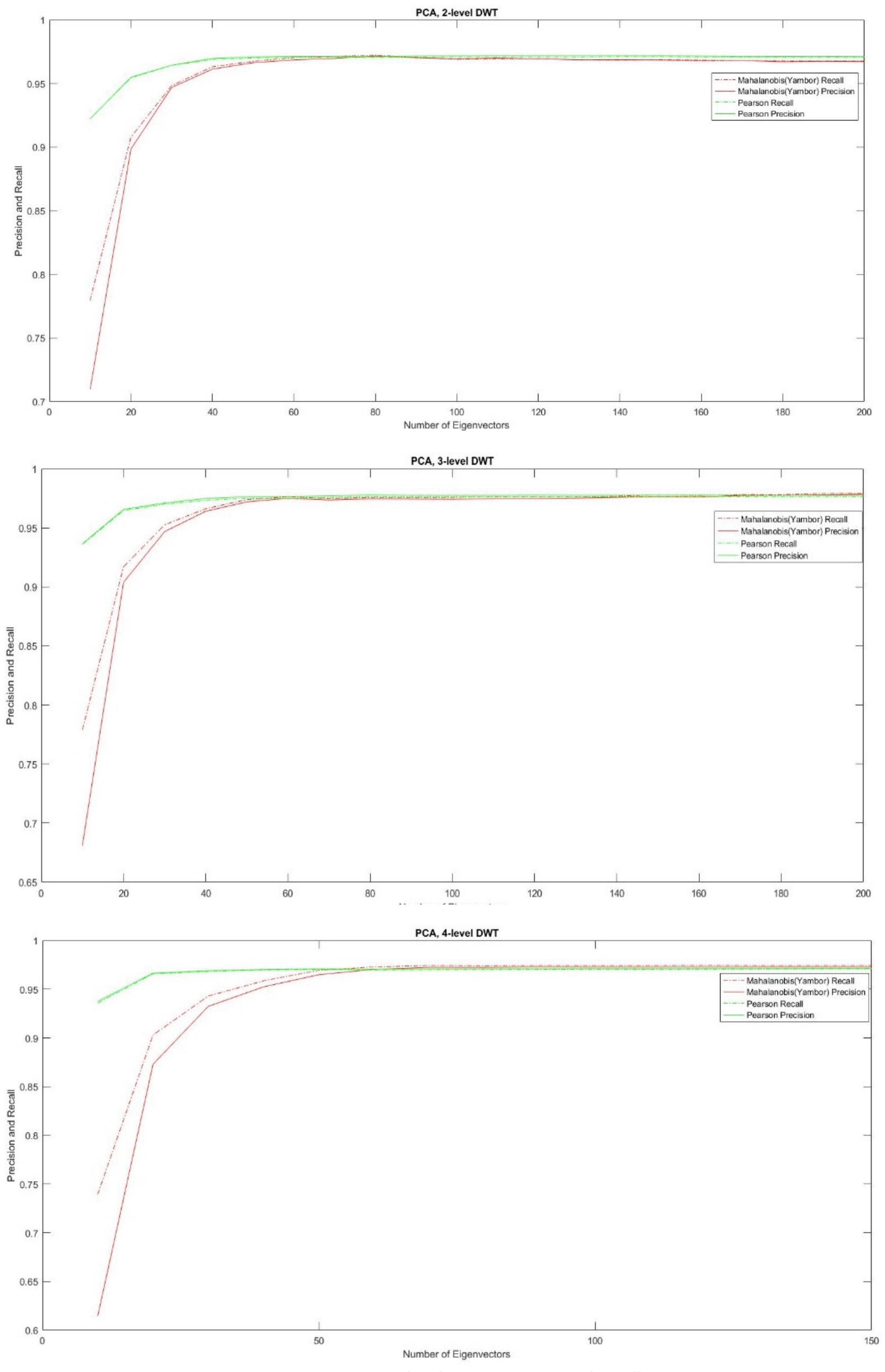


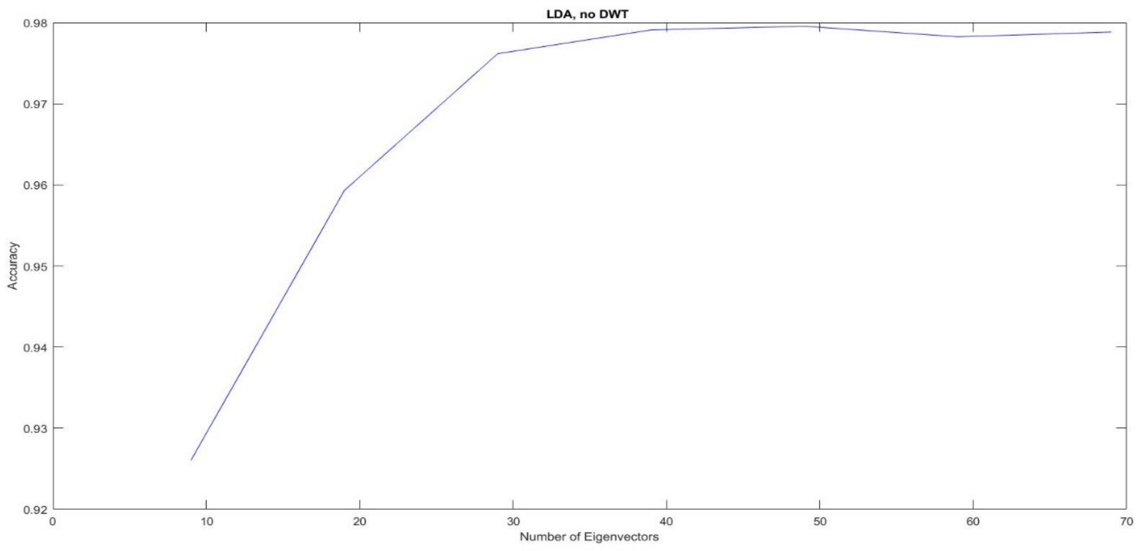
Figure 6-4: PCA, DWT levels 1-4, precision and recall

The boost in performance is also evident in the recall and precision metrics. We summarize the results in the table below.

Method	Naive	PCA	PCA, 1-DWT	PCA, 2-DWT	PCA, 3-DWT	PCA, 4-DWT
Distance	Pearson	Mahalanobis	Mahalanobis	Mahalanobis	Pearson	Citiblock
Features	10394	100	90	80	80	50
Accuracy	93.10%	95.30%	95.6%	96.74%	97.27%	97.04%
S. Dev.	2.42%	2.20%	2.00%	1.88%	1.80%	1.82%
Precision	94.50%	96.03%	96.58%	97.16%	97.80%	97.57%
Recall	94.00%	95.9%	96.53%	97.23%	97.66%	97.37%

6.2 PCA and LDA

Next, we will evaluate the combination of PCA and LDA. In our experiments, cosine distance outperformed all others. To avoid clutter, only the results using that distance will be presented. Due to the upper bound imposed by the desired non-singularity, the number of eigenvectors starts at nine (instead of ten) and we hope to find the maximum accuracy at c-1 = 39 eigenvectors. First, PCA and LDA are performed without the aid of DWT.



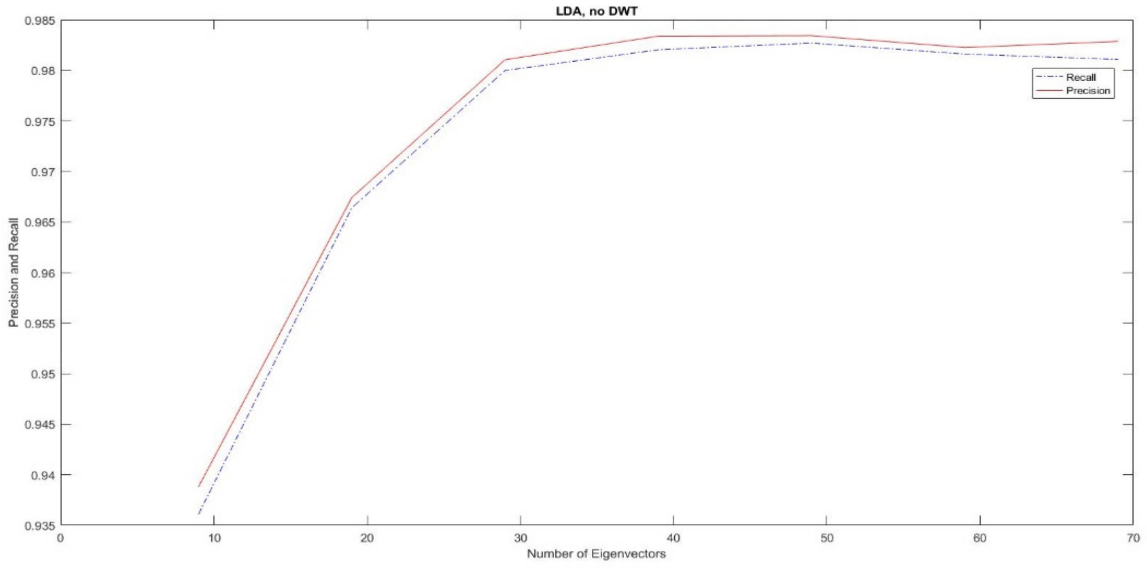
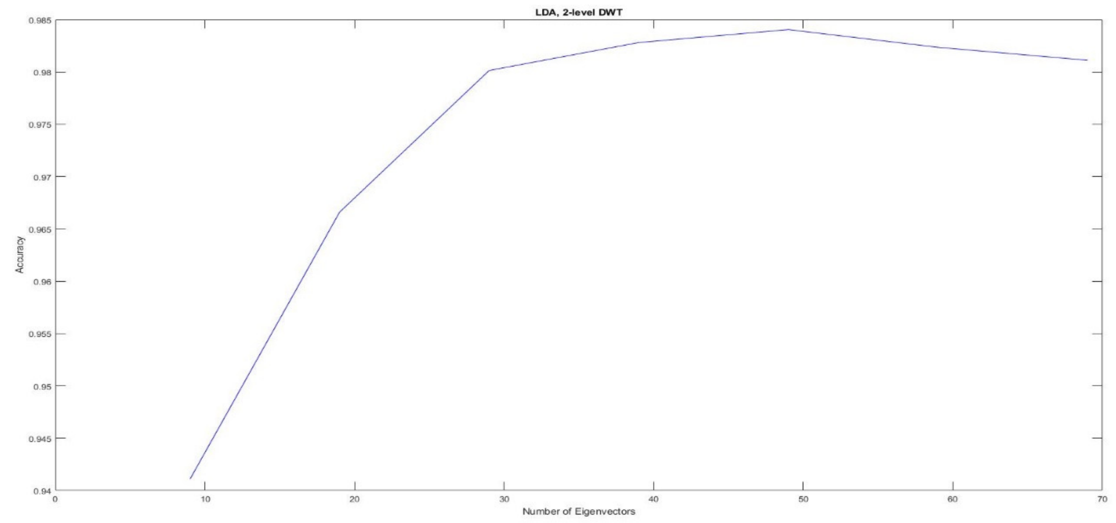
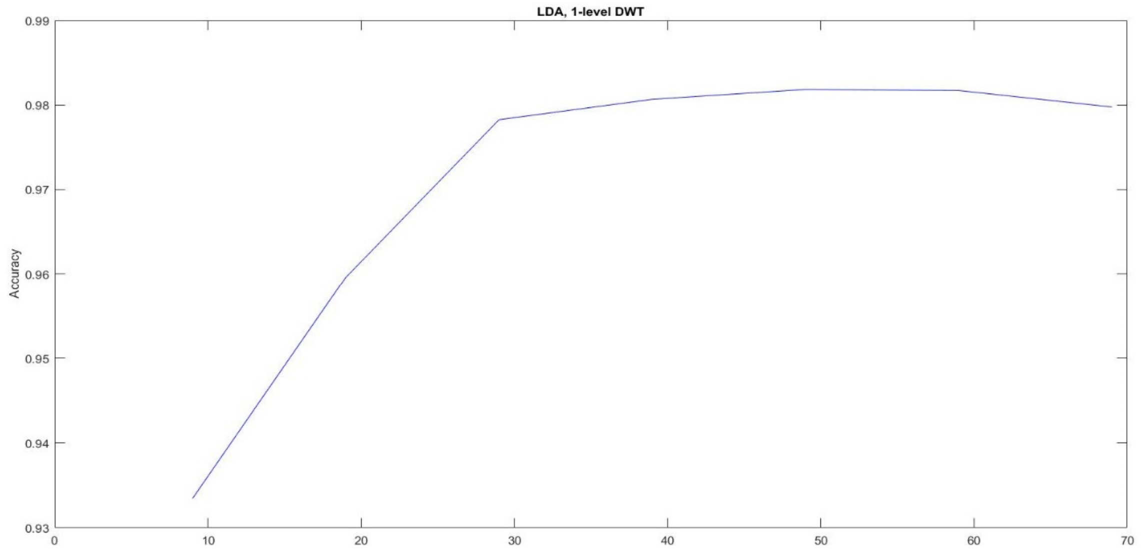


Figure 6-5: LDA, no DWT

It is obvious that LDA outperforms PCA in our case, even when no DWT is applied. At 39 eigenvectors, we reach maximum accuracy when compared to fewer eigenvectors, as expected. For eigenvalue choices greater than 39, the results should be ignored, as non-singularity doesn't stand. Next, the effect of DWT is going to be presented for levels 1-3 as we have already established that at level 4 there is a drop in performance due to the input size.





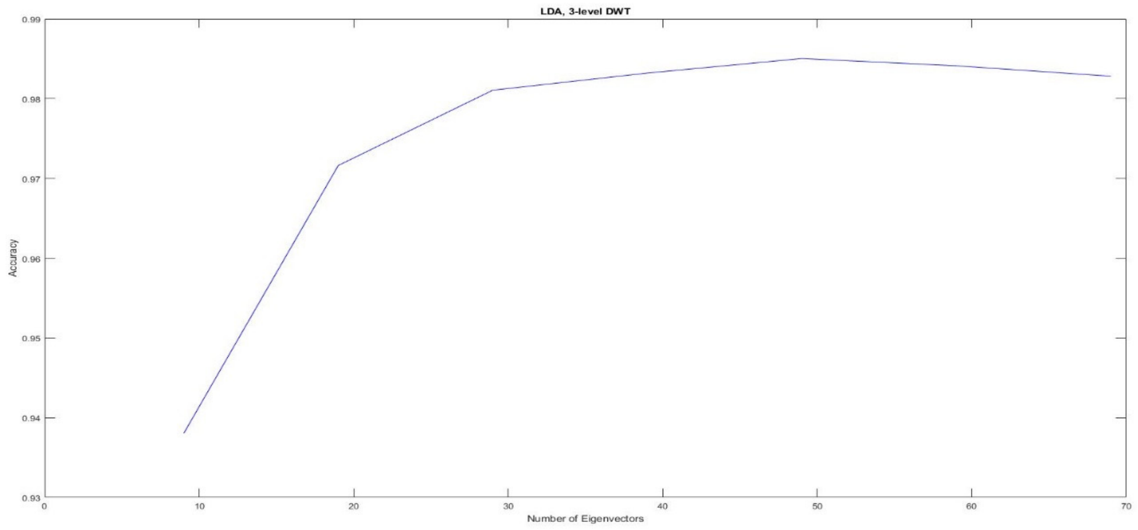
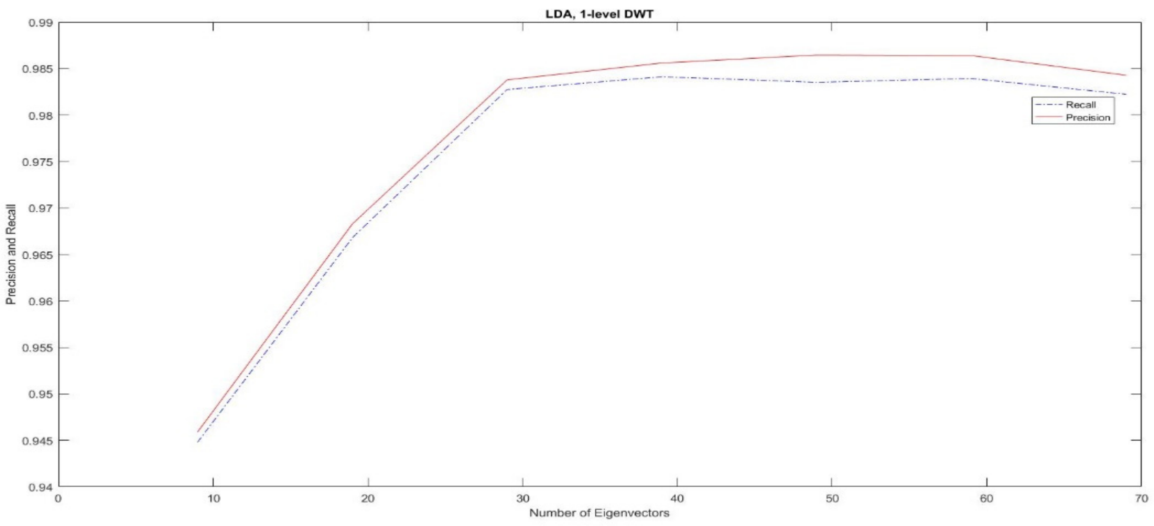
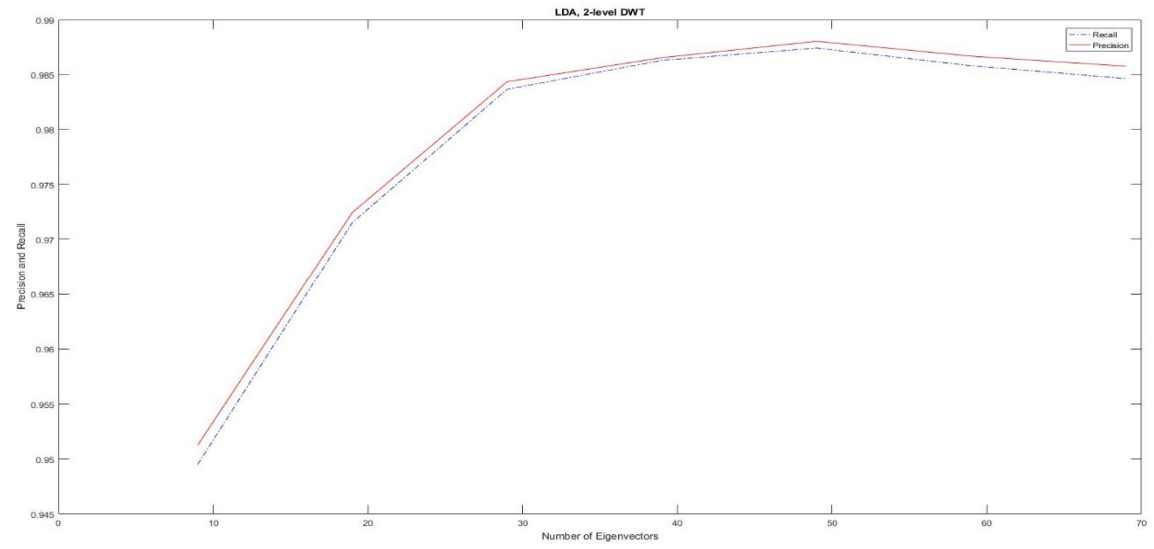


Figure 6-6: LDA, DWT, levels 1-3, accuracy





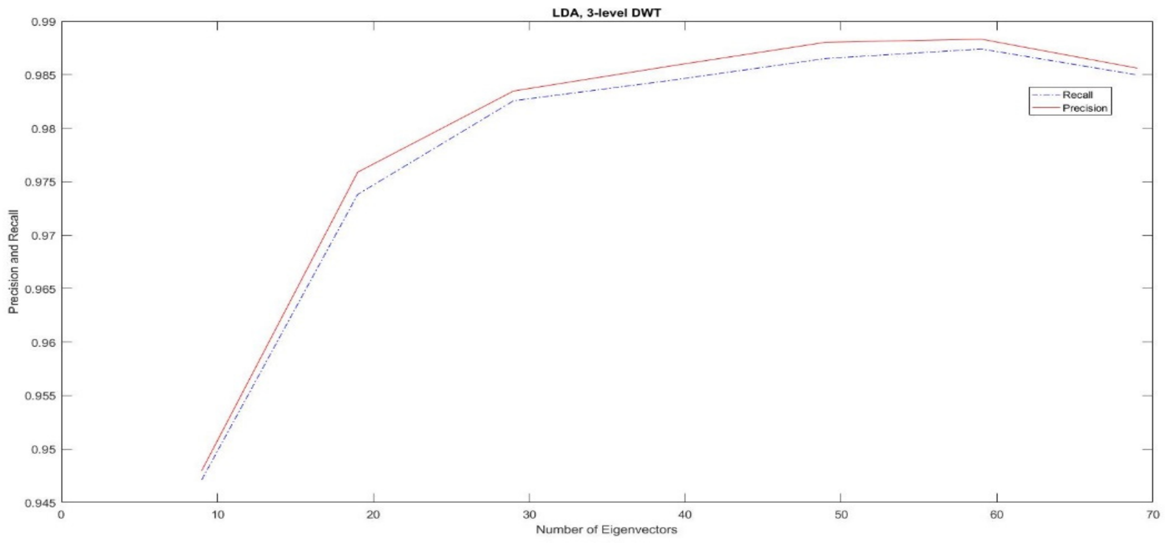


Figure 6-7: LDA, DWT levels 1-3, precision and recall

The results are summarized below.

Method	PCA, LDA	PCA, LDA, 1-DWT	PCA, LDA, 2-DWT	PCA,LDA, 3-DWT
Distance	Cosine	Cosine	Cosine	Cosine
Features	39	39	39	39
Accuracy	97.91%	98.07%	98.28%	98.32%
S. Dev.	1.62%	1.5%	1.49%	1.6%
Precision	98.34%	98.56%	98.66%	98.72%
Recall	98.02%	98.41%	98.60%	98.49%

Even though level 3 DWT seems to be the best in this case also, the difference between level 3 and level 2 is too small to make any assumptions.

Because the combination of PCA, LDA and DWT is the highest scoring statistical method we implemented, we would like to see how it performs against similar methods on the ORL dataset. The results from three papers will be compared to ours [70] (2016), [15] (2012) and [69] (2009). Because they favored a 50/50 dataset split, which understandably can cause a performance drop, we will rerun our experiment with the same split but without ensuring that each class will be represented by 5 images, the choice of images will be completely random.

Method	PCA, LDA, 3-DWT	LDA, 1-Haar DWT **[15]**	PCA, RBF SVM **[69]**	LPP **[70]**	OLPP **[70]**	OELDA **[70]**
Accuracy	96%	83%	98.50%	89.30%	90.40%	91.40%
S. Dev	2.20%	-	-	5.58%	5.85%	6.28%

Our method outperforms all methods in [70], however it scores significantly lower than PCA + SVM. Our 70/30 split result comes closer, but it still is not enough. It should be noted however that SVM is a computationally intensive classifier and can run many times slower than k-NN. Even though tests were made, we were not able to reproduce that result. To try and boost our accuracy we draw influence from [5] . In that paper, the researchers used higher order statistics from wavelet sub-bands to try and classify images that had been digitally tampered. Among other more complex ones, these statistics included variance, skewness and kurtosis. In our case, we took the variance and kurtosis of each level 2 and level 3 sub-band and appended them as extra features at the end of the DWT feature vector (before the application of PCA and LDA).

Method	PCA, LDA, DWT + statistics
Distance	Cosine
Features	39
Accuracy	98.70%
S. Dev.	1.50%
Precision	99.10%
Recall	99.00%

There is an improvement in error rate, from 1.78% to 1.30%, a 22.5% decrease. Also, precision and recall were boosted. The result might not be significant, but calls for further testing of higher order statistics mentioned in the paper.

6.3 PCA and ICA

ICA was also tested. As mentioned earlier, the FastICA implementation was used, with skewness as the G function. In this case, the bibliography agreed that cosine is the most appropriate distance metric [41] [37].

Method	PCA, ICA	PCA, ICA, 1-DWT	PCA, ICA, 2-DWT	PCA, ICA, 3-DWT	PCA, ICA, 4-DWT
Distance	Cosine	Cosine	Cosine	Cosine	Cosine
Features	40	50	60	110	70
Accuracy	94.17%	94.80%	95.57%	96.30%	96.07%
S. Dev.	2.71%	6.5%	7.4%%	3.5%	3.1%
Precision	95.32%	95.67%	96.49%	96.97%	96.66%
Recall	95.32%	95.60%	96.25%	96.81%	96.69%

Even though ICA is reported to achieve good results, in our experiments there is an obvious drop in performance. An earlier study [71] also finds that ICA is inferior to PCA and theorizes that preprocessing may be the culprit.

To compare our results with a more recent study [14], we again recalculate our accuracy using a 50/50 dataset split. Their experiments are very similar to ours, as they test the FastICA algorithm on a range of wavelets on different sub-bands. However they did not test the db4 wavelet. Their top results are reported below, along with ours.

Wavelet	db4	db3	sym8	rBio3.5
Distance	Cosine	Euclidean	Euclidean	Euclidean
Level	3	4	3	3
Accuracy	92.27%	84%	91%	91.50%

Our choice of db4 wavelet and cosine distance performs better than the sym8 and rBio3.5 wavelets with Euclidean distance. Because the difference in accuracy isn't very large it's not clear whether the increase in accuracy is due to the choice of wavelet and/or the distance metric. In the case of PCA, we saw that the difference between metrics was in the range of 0.30%-0.50% so the rrBio3.5 wavelet could be equal to db4 in terms of accuracy, if a different metric was used.

6.4 Transfer Learning

The VGG-Face network is loaded in Keras. The softmax layer is removed from the network. The last layer has 2622 neurons so that's the number of extracted features. Because the input is supposed to be 224 x 224 x 3 (RGB), our grayscale images are resized using interpolation and duplicated across each missing channel. Using 3-NN with Chebyshev as the distance metric, we achieve accuracy of 99.60%, standard deviation of 0.7%, recall of 99.69% and precision of 99.79%. Even with a 50-50 split, the achieved accuracy is 99.07% using 1-NN., higher than PCA with SVM. Because it is not clear how many layers to remove from the top (if any) apart from the softmax, we first performed tests with a varying number of removed layers. Removing only the softmax resulted in higher accuracy. The accuracy when taking features from the dropout or the previous fully connected layer drops to 99.25% and the number of features is almost doubled.

The above experiment doesn't really show the power of deep neural networks and transfer learning. First, we will use a network trained for object recognition. VGG-16 [72] is a network that won a 2014 object recognition benchmark. It is comprised of 16 ReLU activated layers, the first of which convolutional, topped by 3 fully connected and a softmax layer. Max-pooling is also present between some of the convolutional layers. We again remove the softmax layer and use 1-NN with Euclidean distance for classification. Even though it was never trained using face images, it achieves accuracy of 97.14%, standard deviation of 1.6%, recall of 97.25% and precision of 96.97%. The decrease when compared to VGG-Face is expected but could become even smaller when using data augmentation for example. Next, we will present some results using the Georgia Tech database. Because of the differences in scale, the images don't share the same size so the naïve approach is to use interpolation again. We interpolate them to 224 x 224 and for every iteration we randomly choose a subset of 10 (out of 15) pictures of each subject and then use a 70/30 training/testing split. Also, because the ORL database was in grayscale, we will be using a grayscale version of the Georgia Tech database, even though it is originally available in RGB. The results are summarized below.

Method	PCA, 3-DWT	PCA, LDA, 3-DWT	PCA, ICA, 3-DWT
Distance	Pearson	Cosine	Cosine
Features	80	49	50
Accuracy	75.00%	81.03%	69.17%
S. Dev.	3.61%	3.78%	3.59%
Precision	75.64%	82.58%	68.90%
Recall	77.33%	82.55%	71.85%

There is an obvious drop in performance even though we kept the methods, the split and the number of images per person the same. This can be attributed to the following factors. First, there are more subjects present in the Georgia Tech database and as such more ways for a classifier to make a mistake. Second, there are noticeable variations in the backgrounds. Third, the interpolation process is not the suggested way to handle images of different scales, there are other methods not examined in this thesis (e.g. SIFT [73]). The values added to the images can influence their statistical properties and affect related methods.

However, deep CNNs are trained to recognize patterns anywhere in an image and are also generally more powerful. We expect that things that hinder statistical methods, are not going to affect deep networks that much. Indeed, VGG-Face achieves accuracy of 99.82%, standard deviation of 0.3%, recall of 99.83% and precision of 99.79%, using a weighted 1-NN classifier with Euclidean distance. What seems bizarre is that even though the Georgia Tech dataset is more difficult to work with, VGG-Face achieves higher accuracy, recall and precision. This could be attributed to the fact the images in the ORL database need to be interpolated more due to their smaller size.

7 Conclusions and Loopholes in Face Recognition Systems with Machine Learning

In this work we tackled the problem of face recognition and more specifically the task of feature extraction. Two distinct approaches were tested, classic statistical dimensionality reduction with the added twist of wavelet decomposition and state-of-the-art transfer learning. By choosing to study both the past and the present of face recognition, valuable insight has been gained about the field and recognition problems in general. Regarding our results, we were able to show that the db4 wavelet can outperform other recently used wavelets and more studies should include it in their research, as none of the papers used for comparisons utilized it, and also that LDA combined with PCA and 3-level DWT can outperform other statistical approaches. However, utilizing a deep CNN provided astounding results proving why most recent advancements in image recognition fields revolve around deep learning.

However, since this thesis is about a security related topic, not mentioning any attack vectors would be a mistake. In this section, we will talk about known and possible vulnerabilities that any face recognition system designer must have in mind.

Let's start with thinking how a face recognition system would work in practice. An individual would walk up to a camera that would take a picture of their face which would then be preprocessed (aligned, possibly converted to grayscale etc.) and fed to a face recognition algorithm like the ones shown above. What could go wrong? No need to think about crazy scenarios from movies! The system mentioned above, comprised of complex algorithms, can be fooled by a kid with a printer. All someone needs to do is to print a picture with the victim's face clearly visible and the system will be fooled. In the age of social media, finding an appropriate picture is certainly an easy task. This is not mere speculation. Thankfully (for the sake of this example, not its customers) Samsung decided to ignore the need for a secure implementation and made the headlines earlier in 2017, when its flagship and very expensive smartphone, got fooled by a piece of paper [10]. What is more disconcerting is that they added an extra layer of security, that required users to blink, but completely missed the point again. An attacker just needs two pieces of paper now instead of one! One could argue that biometrics on everyday mobile devices are just a gimmick. Or that the limited resources they have can limit how secure the implementations can be. It is true that a mobile phone for example, is expected to be very fast. Any delays can decrease user satisfaction and of course, sales. However, that cannot excuse badly implemented security. Any security feature must be implemented with the users' data privacy and integrity as the top priority. If a feature cannot ensure those two, then it should not be implemented at all. Security is not just a buzzword. An unsuspecting user can suffer dire consequences when a feature advertised as "secure", is nothing but. The leak of personal and professional data is a common phenomenon. Fortunately, there is relevant research on how to secure face recognition. One example is a recent (2014) paper [74] that utilizes motion analysis to detect attacks as the one mentioned above.

Something a bit more general concerns how deep neural networks actually recognize images. In a 2015 paper [75], researchers were able to create artificial images using evolutionary algorithms or gradient ascent that were mostly unrecognizable by humans but popular neural networks identified them with high certainty.

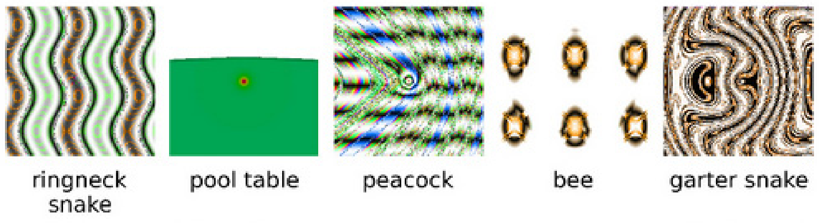


Figure 7-1: Artificial images and their recognized labels [75]

A year earlier, researchers managed to make deep networks to misclassify almost identical images

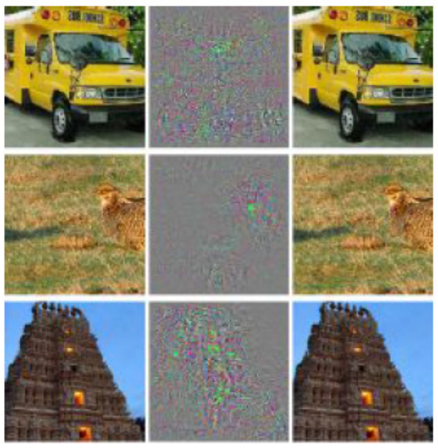


Figure 7-2: Correctly identified(left), incorrectly identified(right) and the amplified difference between the two(center) [91]

These studies could possibly help us achieve a better understanding of neural networks in the future, but also helps us keep in mind that any technique, no matter how powerful can have blind spots. It is not certain whether something similar will or can be used in face (or any biometric) recognition but the possibility should not be ignored easily.

8 Future Work

As this thesis covers a wide spectrum, there are many avenues for future work. First and foremost, the effects of the db4 wavelet should be further explored in more datasets and methods and certainly not be ignored in related studies, as well as the higher order statistics generated with it. An extension of the wavelet transform, the curvelet transform [76] could also be considered. Regarding transfer learning, the transferability of features should be tested on more obscure and limited datasets, possibly in the biomedical field for diagnostic purposes. As far as network visualization is concerned, DeepVis [77] is a library that allows easy visualization of neurons' outputs at runtime. This tool can provide insights to how networks learn and expand our understanding of them.

8.1 Proposed Associations

To further expand upon our previous work and derive useful links among the classic and modern approaches, we attempt to associate the black box operations within a deep convolutional neural network with classic feature extraction, dimensionality reduction and classification. In this way we can explain and justify the layers of a network based on statistical analysis methodologies. In addition, we can also exploit the wavelet transform within the network, an approach that was recently presented in [78], a work that proceeded in parallel to this thesis.

First, we will explore deeper the concept of 2-D DWT. We saw that in the 2-D case DWT is computed with a filter bank comprised of two different filters, a low-pass filter ϕ and a high-pass filter ψ. This structure is derived from a filter property called separability. A separable filter f(x,y) can be written as the product of two filters, i.e. f(x,y) = h(x)k(y). In the case of DWT we can define four filters, one for each sub-band, as

$$f_{LL}(x,y) = \varphi(x)\varphi(y)$$

$$f_{LH}(x,y) = \varphi(x)\psi(y)$$

$$f_{HL}(x,y) = \psi(x)\varphi(y)$$

$$f_{HH}(x,y) = \psi(x)\psi(y)$$

The above filters are 2-D kernels, like the filters used in a convolutional layer. Then, these filters are convolved across the input image for the derivation of the DWT, in a way that looks identical to the use of convolutional layers in convolutional neural networks. After subsampling, the generated values are the wavelet coefficient sub—bands at decomposition level one. This can be repeated until the desirable level of decomposition is reached.

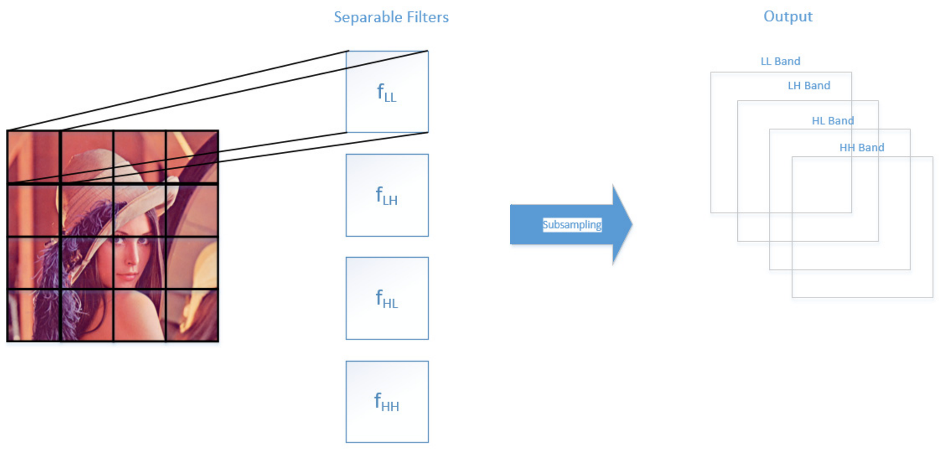


Figure 8-1: Block diagram of 2-D DWT

In our methodology, only the LL band is kept. By using f_{LL} filters of multiple wavelets we directly associate the 2-D DWT in the form of a convolutional and a pooling layer and vice-versa.

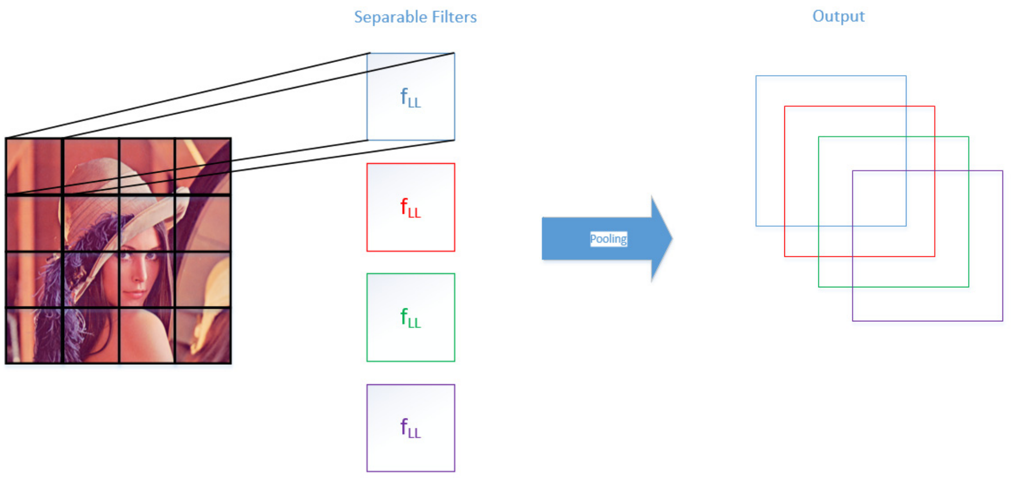


Figure 8-2: Proposed association (DWT –Convolutional layer)

Each generated feature map is the LL sub-band of a DWT using different filters. This association describes one of the most popular deep neural network layer architectures and raises the question whether such associations are not just conceptual but also practical. In the parallel work of [78], Gabor filters were used to successfully decrease the time and energy in the training step of a deep convolutional neural network. Given this result and the associations provided above, the use of other separable filters, that have been extensively employed in 2-D DWT, could be examined as future work, as there is a rich selection of them in wavelet related studies.

We will now attempt to justify the use and the need of the last building block of convolutional neural networks, the fully-connected layer. To briefly reiterate, a fully-connected layer in the context of deep CNNs, combines and reduces the dimensionality of its input data and can be used with any activation function (e.g. ReLU, Softmax) or even extended (e.g. dropout).

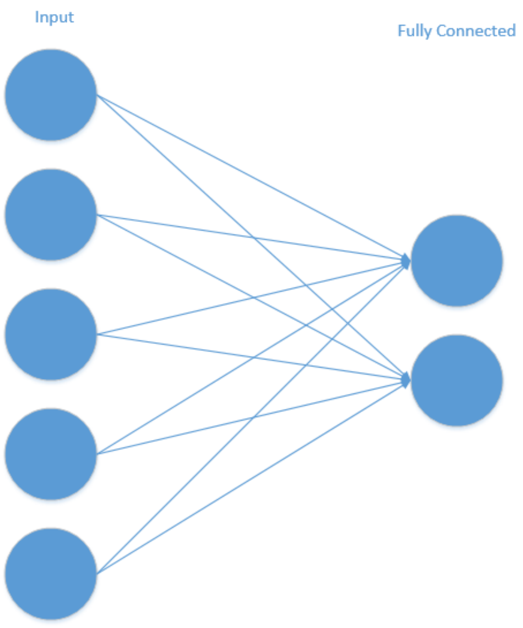


Figure 8-3:A fully-connected layer

Every input value is multiplied with a weight value for every neuron in the fully-connected layer and the output is calculated using the neuron bias and the chosen activation function. Mathematically, the multiplication step can be seen as the projection of the 1 x N input with the N x M weights, where M≤N. In that sense, the weights can be seen as basis vectors for the fully-connected subspace.

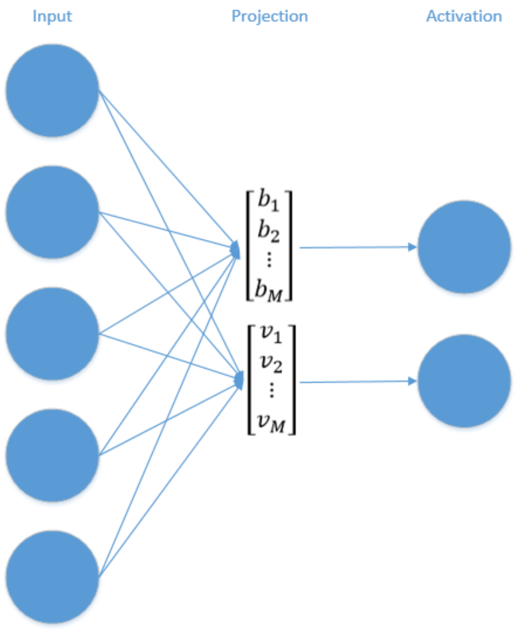

Input
Projection
Activation
$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix}$
$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{bmatrix}$

Figure 8-4: Proposed association (Fully-connected layer - Subspace projection)

As more than one fully-connected layers are usually used, the deeper layers perform an even further dimensionality reduction up to the last one, the Softmax layer, which has the size of the classes present in the data. We propose that this multiple-step projection to a predetermined dimension (the number of classes) can be associated with the statistical reduction and reorganization of the data, achieved using the subspace projection methods presented in this thesis, PCA, LDA, and ICA. The first stages can be associated with projections to orthogonal spaces, while the late steps achieve further decomposition onto independent feature subspaces.

Thus, through our joint study of the state-of-the-art, the statistical foundations and DWT, we are able to provide associations and justifications for the structures comprising deep convolutional neural networks, using already established and well documented concepts.

Bibliography

[1] B. Violino, “Biometric security is on the rise | CSO Online,” 4 March 2015. [Online]. Available: http://www.csoonline.com/article/2891475/identity-access/biometric-security-is-on-the-rise.html. [Accessed 27 July 2017].

[2] “National Law Enforcment Museum Insider,” [Online]. Available: http://www.nleomf.org/museum/news/newsletters/online-insider/november-2011/bertillon-system-criminal-identification.html. [Accessed 27 July 2017].

[3] O. Solon, “Facial recognition database used by FBI is out of control, House committee hears,” The Guardian, 17 March 2017. [Online]. Available: https://www.theguardian.com/technology/2017/mar/27/us-facial-recognition-database-fbi-drivers-licenses-passports. [Accessed 17 August 2017].

[4] B. Usevitch, “A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000,” *IEEE Signal Processing Magazine,* 2001.

[5] H. Farid and S. Lyu, "Higher-order Wavelet Statistics and their Application to Digital Forensics," in *Conference on Computer Vision and Pattern Recognition*, Madison, 2003.

[6] "Kairos," [Online]. Available: https://www.kairos.com/.

[7] "Google Vision API," [Online]. Available: https://cloud.google.com/vision/.

[8] "Amazon Rekognition," [Online]. Available: https://aws.amazon.com/rekognition/.

[9] “How does Facebook suggest tags?,” [Online]. Available: https://www.facebook.com/help/122175507864081. [Accessed 10 Auhust 2017].

[10] R. Amadeo, “Galaxy S8 face recognition already defeated with a simple picture | Arstechnica,” 31 3 2017. [Online]. Available: https://arstechnica.com/gadgets/2017/03/video-shows-galaxy-s8-face-recognition-can-be-defeated-with-a-picture/. [Accessed 5 June 2017].

[11] J. Vincent, “MasterCard unveils 'selfie' security checks, says heartbeat authentication could follow,” 23 February 2016. [Online]. Available: https://www.theverge.com/2016/2/23/11098540/mastercard-facial-recognition-heartbeat-security. [Accessed 11 August 2017].

[12] J. Wakefield, "Comedy club charges per laugh with facial recognition," 9 October 2014. [Online]. Available: http://www.bbc.com/news/technology-29551380. [Accessed 11 August 2017].

[13] G. Feng, P. Yuen and D. Dai, “Human face recognition using PCA on wavelet subband,” *Journal of Electronic Imaging,* 2000.

[14] K. Kinage and S. Bhirud, "Face recognition based on Independent Component Analysis on wavelet subband," in *International Conference on Computer Science and Information Technology*, Chengdu, 2010.

[15] P. Marasamy and S. Sumathi, "Automatic recognition and analysis of human faces and facial expression by LDA using wavelet transform," in *International Conference on Computer Communication and Informatics*, Coimbatore, 2012.

[16] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *IEEE Conference on Computer Vision and Pattern Recognition* , 2014.

[17] Y. Sun, X. Wang and X. Tang, “Deep Learning Face Representation from Predicting 10,000 Classes,” in *Computer Vision and Pattern Recognition (CVPR)*, Columbus, 2014.

[18] A. L. Cambridge, “The Database of Faces,” [Online]. Available: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. [Accessed 4 June 2017].

[19] G. Huang and E. Learned-Miller, “Labeled Faces in the Wild: Updates and New Reporting Procedures,” University of Massachusetts, Amherst, 2014.

[20] A. Nefian and M. Hayes, "Maximum likelihood training of the embedded HMM for face detection," in *IEEE International Conference on Image Processing*, Vancouver, 2000.

[21] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Transactions on Information Theory,* vol. 36, no. 5, pp. 961-1005, 1990.

[22] I. Daubechies, 10 Lectures on Wavelets, SIAM, 1992.

[23] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine,* 1901.

[24] H. Hotetling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology,* 1933.

[25] C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[26] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning,* vol. 20, no. 3, pp. 273-297, 1995.

[27] A. Pentland and M. Turk, “Eigenfaces for recognition,” *Journal of Cognitive Neuroscience,* vol. 3, no. 1, pp. 71-86, 1991.

[28] J. Yang, D. Zhang, A. Frangi and J. Yang, “Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 26, no. 1, pp. 131-137, 2004.

[29] I. Jolliffe, "A Note on the Use of Principal Components in Regression," *Journal of the Royal Statistical Society,* vol. 31, no. 3, pp. 300-303, 1982.

[30] W. Yambor, B. Draper and J. Beveridge, Empirical Evaluation Methods in Computer Vision (chapter), World Scientific, 2002.

[31] Y. Moses, Y. Adini and S. Ullman, “The Problem of Compensating for Changes in Illumination Direction,” in *European Conference on Computer Vision*, Stockholm, 1994.

[32] R. Fisher, “The Use of Multiple Measurements in Taxonomic Problems,” *Annals of Eugenics,* 1936.

[33] P. Belhumeur, J. Hespanha and D. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,” in *European Conference on Computer Vision*, Cambridge, 1996.

[34] K. Etemad and R. Chellapa, “Discriminant analysis for recognition of human face images,” in *International Conference on Audio- and Video-Based Biometric Person Authentication*, Rye Brook, 1997.

[35] A. Martinez and A. Kak, “PCA versus LDA,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, no. 2, pp. 228-233, 2001.

[36] E. O. A. Hyvärinen, “Independent Component Analysis: A Tutorial,” April 1999. [Online]. Available: http://cis.legacy.ics.tkk.fi/aapo/papers/IJCNN99_tutorialweb/. [Accessed May 2017].

[37] B. Draper, K. Baek, M. Bartlett and J. Beveridge, “Recognizing faces with PCA and ICA,” *Computer Vision and Image Understanding,* vol. 91, no. 1-2, pp. 115-137, 2003.

[38] A. Bell and T. Sejnowski, “An Information-Maximization Approach to Blind Separation and Blind Deconvolution,” *Neural Computation,* vol. 7, no. 6, pp. 1129-1159, 1995.

[39] J. Cardoso, “High-Order Contrasts for Independent Component Analysis,” *Neural Computation,* vol. 11, no. 1, pp. 157-192, 1999.

[40] A. Hyvärinen, “Fast and Robust Fixed-Point Algorithms for Independent Component Analysis,” *IEEE Transactions on Neural Networks,* vol. 10, no. 3, pp. 626-634, 1999.

[41] M. Bartlett and J. Movellan, “Face Recognition by Independent Component Analysis,” *IEEE Transactions on Neural Networks,* vol. 13, no. 6, pp. 1450-1464, 2002.

[42] S. Theodoridis and K. Koutroumbas, Pattern Recognition, Academic Press, 2009.

[43] A. Krizhevsky, I. Sutskever and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems* , Lake Tahoe, 2012.

[44] F. Rosenblatt, “The Perceptron--a perceiving and recognizing automaton,” Cornell Aeronautical Laboratory, 1957.

[45] M. Nielsen, "Neural Networks and Deep Learning," [Online]. Available: http://neuralnetworksanddeeplearning.com/. [Accessed 6 June 2017].

[46] Y. LeCun, L. Bottu, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[47] J. Springenberg, A. Dosovitskiy, T. Brox and M. Riedmiller, “Striving for Simpicity: The All Convolutional Net,” in *ICLR*, 2015.

[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[49] O. Parkhi, A. Vedaldi and A. Zisserman, “Deep Face Recognition,” in *British Machine Vision Conference*, Swansea, 2015.

[50] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, 2014.

[51] G. Hu, Y. Yang, D. Yi, J. Kittler, W. Christmas, S. Li and T. Hospedades, “When Face Recognition Meets with Deep Learning: An Evaluation of Convolutional Neural Networks for Face Recognition,” in *IEEE International Conference on Computer Vision Workshop (ICCVW)*, Santiago, 2015.

[52] L. Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research,* 2008.

[53] M. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *Computer Vision – ECCV 2014*, Zurich, 2014.

[54] R. Caruana, "Multitask Learning," *Machine Learning,* vol. 28, no. 1, pp. 41-75, 1997.

[55] R. Raina, A. Battle, H. Lee, B. Packer and A. Ng, “Self-taught learning: transfer learning from unlabeled data,” in *International conference on Machine learning*, Corvalis, 2007.

[56] A. Ahmed, K. Yu, W. Xu, Y. Gong and E. Xing, "Training Hierarchical Feed-Forward Visual Recognition Models Using Transfer Learning from Pseudo-Tasks," in *European Conference on Computer Vision*, Marseille, 2008.

[57] H. Azizpour, A. Razavian, J. Sullivan, A. Maki and S. Carlsson, "Factors of Transferability for a Generic ConvNet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 38, no. 9, pp. 1790-1802, 2015.

[58] A. Razavian, H. Azizpour, J. Sullivan and S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Columbus, 2014.

[59] J. Donahue, Y. Jia, J. H. O. Vinyals, N. Zhang, E. Tzeng and T. Darrell, "DeCAF: a deep convolutional activation feature for generic visual recognition," in *International Conference on Machine Learning*, 2014.

[60] M. Oquab, L. Bottou, I. Laptev and J. Sivic, “Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, 2014.

[61] "Tensorflow," [Online]. Available: https://www.tensorflow.org/.

[62] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,” arXiv e-print, 2014.

[63] A. Krizhevsky, I. Sutskever and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Neural Information Processing Systems (NIPS)*, Lake Tahoe, 2012.

[64] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, “How transferable are features in deep neural networks?,” in *International Conference on Neural Information Processing Systems*, Montreal, 2014.

[65] “Keras Documentation,” [Online]. Available: https://keras.io/.

[66] L. Wolf, T. Hassner and I. Maoz, "Face Recognition in Unconstrained Videos with Matched Background Similarity," in *IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, 2011.

[67] Microsoft, “An Introduction to Computational Networks and the Computational Network Toolkit,” 2014.

[68] T. D. Team, “Theano: A Python framework for fast computation of mathematical expressions”.

[69] O. Faruqe and A. Hasan, "Face recognition using PCA and SVM," in *International Conference on Anti-counterfeiting, Security, and Identification in Communication*, Hong-Kong, 2009.

[70] C. Lin, B. Wang, X. Fan, Y. Ma and H. Liu, "Orthogonal enhanced linear discriminant analysis for face recognition," *IET Biometrics,* vol. 5, no. 2, pp. 100-110, 2016.

[71] K. Baek, B. Draper, J. Beveridge and K. She, "PCA vs. ICA: A comparison on the FERET data set," in *4th International Conference on Computer Vision*, 2002.

[72] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 2014.

[73] D. Lowe, “Object Recognition from Local Scale-Invariant Features,” *Proceedings of the International Conference on Computer Vision,* vol. 2, no. 2, p. 1150, 1999.

[74] A. Anjos, M. Chakka and S. Marcel, "Motion-based counter-measures to photo attacks in face recognition," *IET Biometrics,* vol. 3, no. 3, pp. 147-158, 2014.

[75] A. Nguyen and J. C. J. Yosinski, “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, 2015.

[76] J. Ma and G. Plonka, “The Curvelet Transform,” *IEEE Signal Processing Magazine,* March 2010.

[77] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs and H. Lipson, “Understanding Neural Networks Through Deep Visualization,” in *ICML Deep Learning Workshop 2015*, Lille, 2015.

[78] S. Sarwar, P. Panda and K. Roy, “Gabor Filter Assisted Energy Efficient Fast Learning Convolutional Neural Networks,” in *IEEE/ACM International Symposium on Low Power Electronics and Design*, 2017.

[79] V. Spruyt, “Computer Vision for Dummies,” [Online]. Available: http://www.visiondummy.com/2014/05/feature-extraction-using-pca/. [Accessed 6 May 2017].

[80] Novarank, “Wikipedia,” 13 November 2016. [Online]. Available: https://commons.wikimedia.org/wiki/File:BSS-example.png. [Accessed 22 May 2017].

[81] R. Polikar. [Online]. Available: http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html. [Accessed 27 May 2017].

[82] R. Wang, “Fast Wavelet Transform (FWT) and Filter Bank,” 16 December 2008. [Online]. Available: http://fourier.eng.hmc.edu/e161/lectures/wavelets/node7.html. [Accessed 28 May 2017].

[83] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, Third Edition, Pearson, 2010.

[84] N. Instruments, "2D Signal Processing (Advanced Signal Processing Toolkit)," [Online]. Available: http://zone.ni.com/reference/en-XX/help/371419D-01/lvasptconcepts/wa_dwt_2d/. [Accessed 28 May 2017].

[85] A. Ajanki, “Wikipedia,” 28 May 2007. [Online]. Available: https://en.wikipedia.org/wiki/File:KnnClassification.svg. [Accessed 2 June 2017].

[86] "Wavelet Browser," [Online]. Available: http://wavelets.pybytes.com/wavelet/db4/. [Accessed 3 June 2017].

[87] A. Desphande, “A Beginners Guide to Understanding Convolutional Neural Networks,” 20 July 2016. [Online]. Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/. [Accessed 8 June 2017].

[88] “Stanford University CS231n: Convolutional Neural Networks for Visual Recognition,” [Online]. Available: http://cs231n.stanford.edu/. [Accessed 8 June 2017].

[89] A. Saxena, "Convolutional Neural Networks (CNNs): An Illustrated Explanation," 29 June 2016. [Online]. Available: http://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/. [Accessed 8 June 2017].

[90] L. Wolf, T. Hassner and Y. Taigman, “Effective Unconstrained Face Recognition by Combining Multiple Descriptors and Learned Background Statistics,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 2010.

[91] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, “Intriguing properties of neural networks,” in *International Conference on Learning Representations*, 2014.

The bibliography used is a mix of online resources, new and older papers. Most online resources are either figures, commercial products and articles in the press, however, some of them were used to describe theoretical foundations. We were able to do that after scrutiny and cross-evaluation and by only considering reputable sources (e.g. Stanford University course). For the classic methods, we opted for older but well established bibliography but all comparisons were made using recent publications.