UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Recent advances in face detection and face recognition algorithms on still images

Diploma Thesis

Efstathios Tsitsopoulos

Supervisor: Georgios Thanos

Volos 2020

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Recent advances in face detection and face recognition algorithms on still images

Diploma Thesis

Efstathios Tsitsopoulos

Supervisor: Georgios Thanos

Volos 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τελευταίες ερευνητικές εξελίξεις στην περιοχή εντοπισμού προσώπου και αναγνώρισης προσώπου σε εικόνα

Διπλωματική Εργασία

Ευστάθιος Τσιτσόπουλος

Επιβλέπων/πουσα: Γεώργιος Θάνος

Βόλος 2020

Approved by the Examination Committee:

Supervisor **Georgios Thanos**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Member **Aspasia Daskalopoulou**
Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Hariklia Tsalapata**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Date of approval: 20-9-2020

Acknowledgements

I would like to thank my family and friends for their support.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Efstathios Tsitsopoulos
15-9-2020

Abstract

Face detection and recognition are two distinct but intuitively similar and challenging Computer Vision problems. Through the years and the variety of challenges a number of different approaches to either detect, recognize or systems that do both, were introduced. In this thesis, literature concerning the face detection and recognition field were examined as well as code from different detection and recognition algorithms. The purpose of this thesis is to provide a solid algorithmic explanation of some approaches, so as to give the reader a view of the different methods used, spanning from the birth of the field to the most recent advancements. The face detection and recognition algorithms to be addressed are Viola - Jones, LBP based detection, MTCNN and Eigenfaces, Fisherfaces respectively.

Περίληψη

Ο εντοπισμός και η αναγνώριση προσώπου είναι δύο ξεχωριστά αλλά φαινομενικά παρόμοια προβλήματα του πεδίου της Όρασης Υπολογιστών. Μέσα από το πέρασμα των χρόνων και τις διάφορες προκλήσεις, αρκετές προσεγγίσεις με σκοπό τον εντοπισμό, την αναγνώριση ή συστήματα που κάνουν και τα δύο έχουν εμφανιστεί. Σε αυτή τη διπλωματική, τόσο βιβλιογραφία όσο και κώδικας από διάφορους αλγορίθμους εντοπισμού και αναγνώρισης εξετάστηκαν. Ο σκοπός είναι να εξηγήσει στον αναγνώστη, από αλγοριθμική ματιά κυρίως, κάποιες προσεγγίσεις και να του δώσει μία σκοπιά της πληθώρας των προσεγγίσεων που χρησιμοποιήθηκαν, από την εμφάνιση του κλάδου μέχρι και τις πιο σύγχρονες εξελίξεις. Οι αλγόριθμοι που θα εξεταστούν είναι οι Viola - Jones, η αναγνώριση με βάση τα LBP και ο MTCNN οσο αναφορά το κομμάτι του εντοπισμού και οι Eigenfaces, Fisherfaces οσο αναφορά το κομμάτι της αναγνώρισης.

Table of contents

List of figures

Chapter 1

Introduction

Face detection and recognition is one of the fastest growing technology fields. In a more and more digital growing world, security of data has never been more important. Spanning from security on mobile phones, buildings and services using face detection and verification [11][12] to pedestrian, hotels, events and airlines security surveillance[13][14][15] face detection and recognition methods are becoming state of the art for security systems. Beyond security, face detection and recognition is used for a great number of applications some of which are emotion and facial expression analysis[16][17], advertising, human and robot interaction[18] and commercial cameras auto-focus technique. It has even been used in healthcare, from detecting diseases based on face features to useful applications for blind people.

Face detection and recognition is a topic that was first introduced in the 1960s mainly for security purposes. It was later expanded and improved with a variety of different approaches like Linear Algebra, Machine Learning techniques like Support Vector Machine or boosting and finally in the modern days with the use of Neural Networks. Through the years many algorithms and techniques have been proposed and many of them today produce exceptional results. Nonetheless it still is an extremely active research field today, due to its applications usages that are of great importance and allow little to no error.

This thesis will present some of the most significant and distinct approaches that have been taken and have led the field of face detection and recognition in the present state. The aim is to give the reader basic algorithmic knowledge of the most innovative for their time algorithms, as well as how Machine Learning, with great focus on Deep Learning have revolutionized the field and piloted to a detection accuracy that sometimes prevails over the

human capabilities. It also aims to make a clear separation of the detection and recognition terms and for this reason detection and recognition algorithms will be presented individually.

1.1 Structure of the thesis

The rest of the thesis will be outlined as follows. In Chapter 2, the boosting based Viola - Jones face detection algorithm will be explained followed by the face detection technique with Local Binary Patterns - LBP with the use of Support Vector Machines. In Chapter 3 the dimension reduction method Principal Component Analysis - PCA will firstly be explained and then the similar face recognition algorithms Eigenfaces and Fisherfaces. In section 4, a basic explanation of Neural Networks will be provided along with a brief history of their development. Furthermore Convolutional Neural Networks architecture will be demonstrated and each layer of the architecture will be presented. Finally, the Face detection deep learning algorithm Multi - task Cascaded Convolutional Neural Networks will be explained. The final chapter is the summary, along with thoughts of how this research could be expanded in the future accounting the most prominent for the future technologies.

Chapter 2

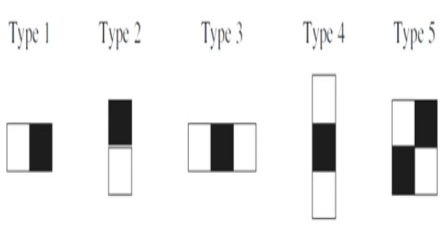
Face Detection

Face detection is a special case of the Computer Vision problem of object-class detection. The goal of any facial detection system is: Given an arbitrary image, determine whether it contains a human face or not. Often, face detection is combined with face localization. Face localization is the task of not only finding whether or not the image contains a face, but in the case it does, it tries to find the location that the face resides. This is achieved by utilizing Artificial Intelligence methods. However, this task is associated with a number of challenges. A human face in an image can significantly vary depending on the lighting conditions and the angle of the image, the pose and facial expression of the face, as well as the partial occlusion of the face by other objects. These challenges proved to be a great obstacle in the development of a face detection system, that provides adequate performance in non controlled environments. Face detection is of great importance since it is the building block of any face recognition, verification or tracking algorithm. In the following sections two important face detection algorithms will be presented and explained.

2.1 Viola-Jones

Viola-Jones is the first framework used for real time face detection applications. It was proposed in 2001 by Paul Viola and Michael Jones[19]. The basic idea was to train the computer to detect faces, based on the intensity difference on peoples' faces using Haar-like features (see Figure 2.1). Haar features are square shape functions proposed by Alfred Haar in 1909, and were used to extract features from an image. Haar features are placed upon a grey-scale image, where the intensity of each pixel range from 0 to 255 (or 0 to 1 when normalized)

and return a single integer, which is equal to the sum of the intensities of all the pixels under the black area minus the sum of the intensities of the pixels under the black area. Each Haar feature is placed on a detection window (authors used 24 by 24 pixels), and is shifted, scaled, rotated and inverted until all possible features are calculated. The number of these features are more than 180.000, that need to be calculated for each 24 by 24 pixel window. To reduce the enormous amount of computations needed, the authors presented the technologies of the Integral Image, a customized AdaBoost algorithm, and an attentional cascading algorithm for classifiers.



(α□) The 5 Haar features originally used by Viola and Jones.

(β□) Haar feature used on an image to detect the nose. Commonly the nose pixels are brighter than the pixels on the cheek area.

Figure 2.1: Simple example of Haar features operating on an image

2.1.1 Integral Image

With a normal image, the number of the pixel values that we would need to sum up, in order to compute a $x*$y feature would be $x*$y . After we shift this feature, lets say for simplicity by 1 pixel, we would need $(x+1)*$y more and so on, even though we had made computations for most of the covered area in the previous iteration. To reduce this number of repetitive computations, we use the Integral Image. The Integral Image or Summed Area Table is produced by the original Image I (i,j), and it has a very unique characteristic. Each I (i', j') represents the sum of all the pixel values which are left i $\leq$ i' or below it J $\leq$ j', and as so it is derived by the mathematical equation shown below.

$$I'(x', y') = \sum_{x' \leq x, y' \leq y} i(x, y) \tag{2.1}$$

4	1	2	2
0	4	1	3
3	1	0	4
2	1	3	2

(α□) Example of an original 4x4 pixel Image

4	5	7	9
4	9	12	17
7	13	16	25
9	16	22	33

(β□) The Integral Image produced.

Figure 2.2: Original and Integral Image values

The advantage we gain from the construction of the Integral Image, is that we are now able to calculate the sum of intensities under a rectangular area, regardless of size in constant time [20], using only 4 values and 3 addition operations [19]. For any rectangle area where the top left corner is (x1,y1) and bottom right corner is (x2,y2) the sum S is equal to:

$$S = I'(x2, y2) + \text{I'(x1-1,y1-1) - I'(x2,y1-1) - I'(x1-1,y2)} \tag{2.2}$$

2.1.2 Adaboost and feature classification

Even with the use of the Integral Image though, it is not possible nor efficient for a real-time face detector to use all 180.000 haar-features for each 24x24 pixel sub-window. It can easily be deducted that some of these features are more suitable to detect a face than others. To find these more suited features, the authors used a slightly modified version of a machine learning algorithm known as Adaboost[21]. Every feature can be used to produce a classification function $h(x)$, which for any input provides an output of 1 if it classifies the input as a face, or a 0 if not. It is defined as follows:

$$h(x) = \begin{cases} 1, & pf(x) < p\theta \\ 0, & otherwise \end{cases}$$

where f is the value of the Haar feature, p is the polarity (whether the input is labeled as face or not) and θ is the threshold value which is set manually and may differ depending the implementation. This function is known as a weak classifier, since no feature alone can produce a good enough classifier, Adaboost is used to choose the best of these classifiers and make a linear combination of them, thus constructing a strong classifier H(x).

$$H(x) = a_1h_1 + a_2h_2 + a_3h_3 + ... + a_nh_n$$

In order to test and find the best performing features using Adaboost, there are needed thousands of sample images both positive and negative. These sample images are initialized with a weight representing the importance of its right classification. Each of the 180.000 features then proceed in succession to classify all of the images and the classifier with the lowest error rate is picked and added to the strong classifier. Then the weights are updated, with the weights of the wrong classified images having their weights increased, since in the next iteration it is important to find another classifier which is able to correctly classify these images. The procedure ends when the number of weak classifiers, which is set manually, is met. It is to be noted, that this is a very heavy computational process that could take many hours or even days, depending on the number of weak classifiers chosen, to be completed.

2.1.3 Cascading classifiers

A strong classifier can produce very high detection rates with less than 0.5% of the original 180.000 features of a 24x24 pixel image. However, scanning repeatedly thousands of sub-windows, of which the strong majority do not contain faces is still not computationally efficient. In order to increase the performance, a heuristic algorithm is used in conjunction with Adaboost to construct a cascade classifier[22] and reject non-face sub-windows using only a handful of features. For example, a totally white sub-window does not need to be tested with hundreds of features to determine that it has no intensity variations whatsoever, and hence is impossible to contain a face. The goal is to build a series of stages, where each stage contains an increasingly number of features. Each sub-window will begin to be tested

by every stage individually and will proceed to the next stage if and only if the previous one has a positive output. In essence every stage is a test of progressive difficulty.

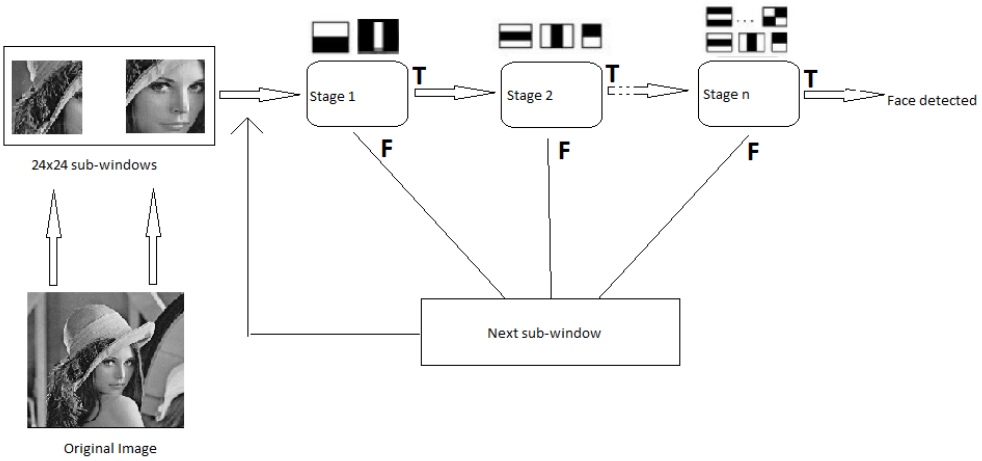


Figure 2.3: Cascading classifiers representation

To construct each stage, a f_i false positive rate threshold and a t_i true positive rate threshold are set, and Adaboost keeps adding classifiers to the strong classifier of each stage until these rates are met. The algorithm stops adding stages and is terminated when a F_t target false positive rate, set manually, is satisfied. f_i is maximum at the early stages and declining, while t_i is minimum and increasing, since the goal in early stages is to find face candidates and not to verify with certainty that it is indeed a face. It should also be pointed out, that to further decrease the false positive rate, a technique with overlapping rectangles is used. In order for a sub-area of the Image to be classified as face, there are needed several successful overlapping sub-windows.

Stages creation

- Given training sample images(x1,y1),...,(x_n, y_n), where $y_i = 0, 1$ for negative and positive examples respectively.

-Initialize F_t, f_i, t_i and the sample weights $w_i = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negative and positive samples.

While $F_t \leq F_{overall}$

Add stage with updated f_i and t_i

While $f_i \leq f_i current$ and $t_i \leq t_i current$ (or a fixed number of weak classifiers is met)

Normalize the weights so that w_t is probability distribution:

$w_i = \frac{w_i}{\sum_{j=1}^{n} w_j}$ (Already normalized in the first iteration)

For each feature ϕ, train a classifier h_ϕ, calculate its error rate e_ϕ and pick the one with the lowest$-> h_b$. $e_\phi = \sum_i w_i |h_\phi(x_i) - y_i|$

Update the weights: $w_i = w_i \beta_b^{1-e_i}$ where $\beta_b = \frac{e_b}{1-e_b}$ and e_i is 0 if classified correctly by h_b or 1 otherwise.

Add to the strong classifier $H(x) += a_b h_b$ where $a_b = \log \frac{1}{\beta_b}$

Figure 2.4: Pseudo code for the stages creation. Create a stage and train strong classifiers until the requirements the user set are met.

Improvements

Viola-Jones is a high performance algorithm accompanied with high detection rates[19], that made a great breakthrough in the face detection and recognition area. However, despite marking almost perfect results in full frontal images in different lighting conditions it displays some major weaknesses. It performs poorly on angled images, has lower detection rate on black skinned faces, the training phase is substantially time-consuming and has a high designing complexity. In the last decades, many improvements on both 3 stages have been

suggested to increase its performance. To reference a few, an extended number of features was proposed[23][24] that showed improved results compared to the original, a modified Adaboost algorithm[25] that displayed increased performance and a better cascading method that used the Support Vector Machine algorithm[26].

2.2 Local Binary Pattern - LBP

In this section I will describe the Local Binary Pattern operator or LBP[27], its different applications and briefly the classifying techniques used by different algorithms for face detection. The core idea is to calculate a LBP code, which will be explained thoroughly below, for each pixel image and create a histogram, each bin containing the appearances of each code, which will be used a texture descriptor. LBP compensates for a number of disadvantages that the Haar-features presented. Due to the overwhelming number of Haar-features, the training stage is considerable time-consuming and although the features in the first stages of the cascade are very efficient, in the last stages the computations needed to reach the wanted detection rate are substantial. LBP features are significantly less in number in any scale of image, simpler to calculate and with more discriminative power than Haar-features, hence both the training time and the classification require less time to complete. Furthermore LBP is also robust in illumination and pose changes since it does not rely on the differences between pixel intensities. Instead it encodes local or global spatial structures and micro-patterns and was originally used to describe textures[28][27]. After Ojala et al. showed its discriminative power and the almost perfect results they got for texture classification, it was then used widely in face detection and recognition, considering that a face and the facial features can be represented as micro-texture patterns.

2.2.1 LBP operator

The LBP operator labels each pixel of a given greyscale image, with a binary number known as the LBP code. To achieve that, it selects a 3x3 neighborhood and proceeds to make binary comparisons between the pixel intensities of the center (x_c, y_c) and value g_c with each of the 8 pixel intensity values g_i around it, replacing g_i with 1 if its value is greater than this of g_c or 0 otherwise. The result is an 8bit binary number which encodes the local texture(see Figure 2.5), and is mathematically derived by the equation below:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} s(g_i - g_c)2^p \tag{2.3}$$

where P is the total pixel number of the neighborhood, 9 in this case, and s is a sign function

$$s(x) = \begin{cases} 1, & x > 0 \\ 0, & otherwise \end{cases}$$

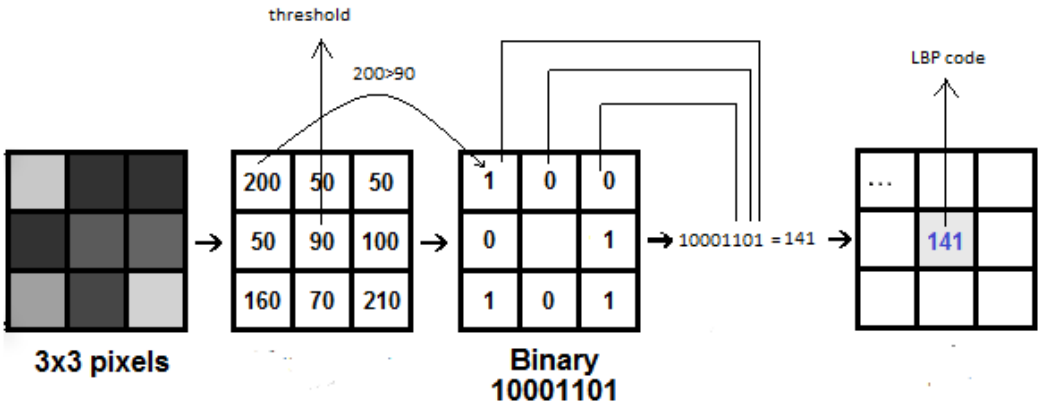


Figure 2.5: LBP original operator on 3x3 neighborhood and the corresponding LBP code

It is to be noted that the binary number appearing, is usually read clockwise starting from the top left cell, but it is not mandatory and some scientific papers read it counter-clockwise. These codes can indicate the occurrence of curved areas or flat areas, edges(see Figure) etc.

MB-LBP and LBP extensions

Despite the efficiency of the LBP, a problem of descriptive power emerges when it comes to larger scaled images due to the locality of a 3x3 operator. In order to achieve that Ojala et al.[28] came up with the $LBP_{(P,R)}$. It is essentially the same operator, but instead of operating on a 3x3 (8 pixel neighborhood), it can be used in any even spaced P pixel neighborhood

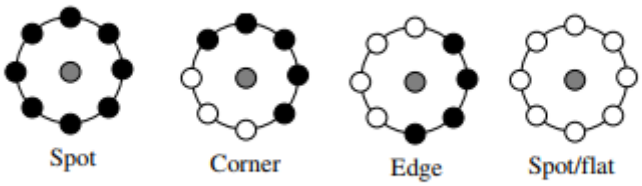


Figure 2.6: Representation of a neighborhood, where black corresponds to 1 and white to 0.

within a R radius around the center pixel. This can be used in any scale of image and neighborhood, thus solving the limitations of the original LBP operator.

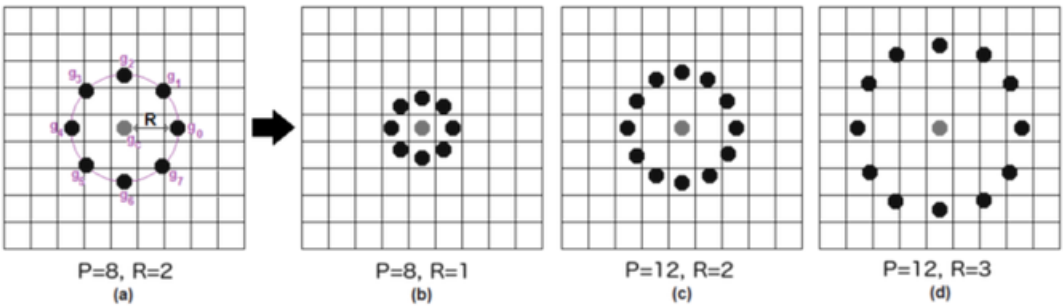


Figure 2.7: Different combinations of P,R LBP

It is definitely worth mentioning, that this limitation was also solved with the Census Transformation by Fröba and Ernst[29] and with the widely used Multi Block-LBP/MB-LBP, which Zhang et al[30] applied on face detection and can be briefly explained as a LBP for neighborhoods instead of pixels. To deal with the pixels near the boundaries of the image where the neighborhood is off the boundaries and thus non-existent, a number of techniques like zero padding or mirroring can be used, however the most usual path is to ignore them since they have little to none influence on the output, resulting at a smaller in height and width by R pixels LBP image.

Rotational Invariant LBP

In order to make the LBP efficient and robust in detecting textures in rotated images, Ojala et al31 attempted to modify the original LBP, and demonstrated the $LBP^{ri}_{P,R}$ which is defined by the equation below:

$$LBP^{ri}_{P,R} = \min_{i=0}^{P-1} ROR(LBP_{P,R}, i) \tag{2.4}$$

where $ROR(x, i)$ is a circular bit wise shift of x, i times. Equation (2.4), says that after the extraction of the binary LBP code, a number of bit-wise shifts occur in order to take the smallest possible binary number. This new $LBP^{ri}_{P,R}$ (ri denoting rotational invariant) reduces the number of different codes and thus giving even more simplicity while maintaining the benefits of the original LBP and being rotational invariant. However, results show that it has its limitations, and was found to output unstable results in numerous occasions.

Uniform Patterns

Another discovery that Ojala et al[28] made, was that the LBP texture descriptors have uneven discriminative power. They realized that in images containing texture, the percentage of LBP codes with two or less bit-wise transitions was more than 80%, even though the number of LBP codes with more than 2 bit-wise transitions is significantly more, regardless the scale of the neighborhood. For example, in a neighborhood of $P = 8$, the number of possible codes is $2^P = 256$, and the number of values containing two or less bit-wise transitions are 58. These kind of patterns are known fundamental or uniform patterns. In their experiements Ojala et al. tried to construct the histogram with 59 bins instead the original 256. 58 were used for each uniform LBP code $LBP^{u2}_{P,R}$, u2 denoting uniform with at most 2 bitwise transitions, and 1 for all other patterns. The results they had were as good if not better to using all 256 bins for the texture description. Similar results were demonstrated in a face description and recognition application by Ahonen et al. [32] in 2006.

2.2.2 Histograms and Classification

LBP and histogram construction

As stated in the beginning, the LBP codes are displayed in a frequency histogram to form a kind of texture descriptor. For any given image the first step is to calculate for each pixel the LBP code and thus obtain an LBP image as seen in Figure 2.8, and construct the corresponding histogram.

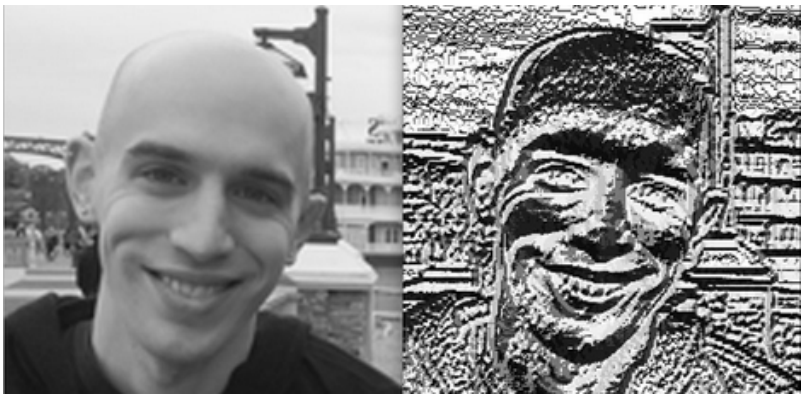


Figure 2.8: Original Image and the output Image with $LBP_{8,1}$ operation. Image taken from [1]

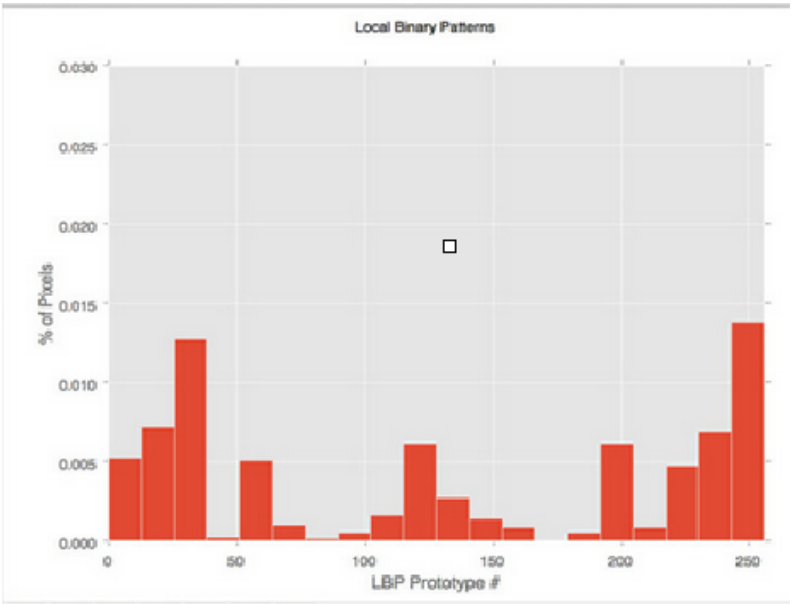


Figure 2.9: Histogram with 256 bins of figure 2.8. (Both uniform and non-uniform patterns are treated equally)

All the micro-patterns of the image are now encoded in the histogram and thus the histogram works as a descriptor. However describing a texture with such a histogram is easier than a face, which is a product of many different micro-patterns. In order to describe the face with more accuracy using LBP operators, Ahonen et al.[33] introduced a different and better representation of the face with great results. They divided the image in M equally sized regions, calculated the LBP histograms for each region and then proceeded to concatenate them in one big frequency histogram 2.10. So now, not only global information is accessible through the histogram, but also information on pixel level through the LBP codes, and in regional level. Specifically the regional level information can be very useful, especially in

face recognition, since the eyes, the nose and ears and other distinct features can now be described through a regional histogram. This technique proved to have strict limitations when used in lower resolution images. Ahonen et al. modified their initial approach 2 years later to successfully detect faces and in lower resolution images [32]. It is to be noted that they used uniform LBP patterns in both of their works referenced above.

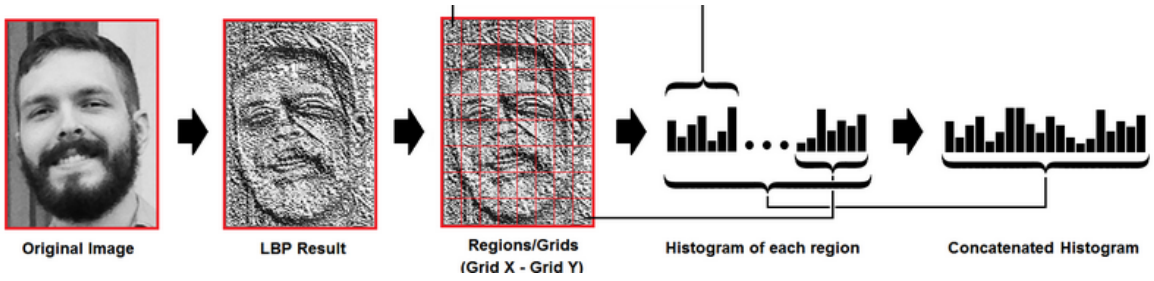


Figure 2.10: Regional and global Histograms concatenated. Image taken from [2]

Classification

There are numerous different algorithms used to classify images using the information gained by the histograms. In this section two of the most used by the scientific community will be presented. In this point it is important to emphasize that similar to the Viola - Jones algorithm, a sub-window is used to scan every input image. The size of the sub-window which indicates the region which is going to be classified, the shifting step of the sub-window and the possibility of overlapping sub-windows are up the user to set, just like in Viola - Jones. Down sampling can also be used to make it easier to locate faces on different scaled images. These variables can affect performance greatly and should be set carefully.

Chi Square: In this approach it is first needed to build a face and a non-face model description. Face and non-face images are used as training sample. From each such image the concatenated histogram is calculated and labeled as face or non-face manually. Then a mean value histogram is obtained by the faces labeled histograms and one from the non faces labeled histograms, which will be used as models. Chi square test is used because it is very simple in its calculation and design, and even though the classifications results are moderate, it indicates the power of the LBP. Chi square is widely used in statistics and very efficient in histogram comparisons. It is essentially a function that compares two histograms and outputs

a single number indicating the difference between the two. It was used by Ahonen et al.[32] mainly for the recognition part and Laura Sánchez López [34] who made improvements in the Ahonen et al. algorithm, and is derived by the below equation:

$$X^2(S, M) = \sum_i \frac{(S_i - M_i)^2}{|S_i + M_i|} \tag{2.5}$$

where i is the feature vector corresponding to the concatenated histogram, whose size varies depending on implementation, S_i is the part of the sample input image to be examined and is determined by the sub - window, and M_i is the counterpart which it will be compared to. The sample is compared to both Models, and depending on the output is classified as face or non - face. Equally to the Viola - Jones algorithm multiple successful sub - windows are needed in an area, for it to be classified as face. There are two different implementations of the chi squared classification, which are explained with great detail in [34][35].

Support Vector Machines - SVM:

Support Vector Machines is a robust prediction method based on statistically learning theory developed by Vladimir Vapnik[36][37]. Due to its successful application on different object detection projects, Ahonen et al. first decided to use it as their classification method. In this section, a brief explanation of the basic idea behind SVM will be provided, with further information in the literature[37][34][38[39]. Given positive and negative input samples placed on D dimensional space, SVM will try to to build a $D-1$ hyperplane to separate the positive and negative samples. When a new sample is to be tested, it is classified depending on its relative position to the hyperplane. In the figure below 2.11 the $|d_-| + |d_+|$ distance is called margin, where d_- is the maximum distance from the nearest positive sample and d_+ the maximum distance from the nearest negative sample. The line H_0 is placed so to maximize the distance between both the negative and positive samples. However any parallel line to H_0 and between the margin could also work as a separation line. The samples which are on H_1 and H_2 lines are called support vectors since they have a major role on the construction of the separation line/boundary line, which can be represented by the following equation:

$$H_0 = w^T x + b \tag{2.6}$$

where w^T are the parameters of the plane, so it is a constant in this example since there are only 2 dimensions, x is a point and b is called the bias and is calculated in order to minimize the error and maximize the distance from the support vectors. If we consider that our samples are labeled as face or non face or $y_i = -1, 1$ then the H_1 and H_2 lines can be defined as $H_1 : -1 = w^T x + b$ and $H_2 : 1 = w^T x + b$. Given a new sample and testing it on the separation line equations, it is now possible to determine based on the sign of the output number whether it is below or above H_0 and classify it accordingly.

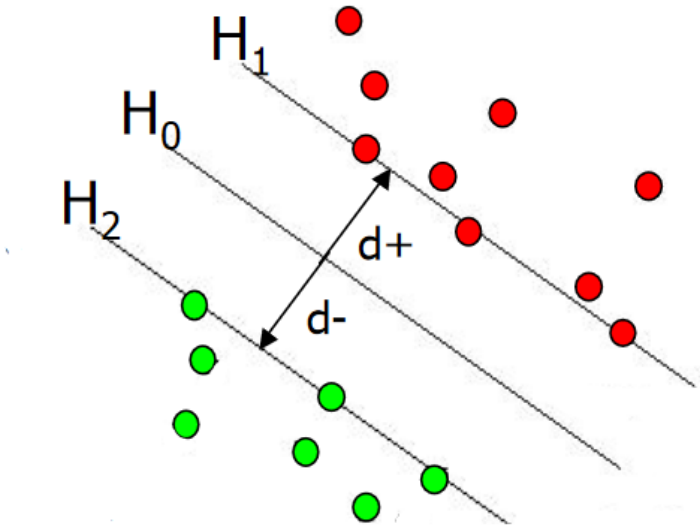


Figure 2.11: Green and red dots are positive and negative samples respectively. $H_0 = wx + b$ is the separation line.

Nonetheless face detection is not a linear problem, and more is needed to be done to apply SVM. In order to apply SVM, Hadid et al. , used something called the Kernel trick. Kernel functions(see Figure 2.12) are used to increase the dimensionality of the samples and find a hyperplane in higher dimensions that best separates these samples.

Kernel	Equation
Linear	$K(x,y) = x \cdot y$
Sigmoid	$K(x,y) = \tanh(ax.y + b)$
Polynomial	$K(x,y) = (1 + x \cdot y)^d$
KMOD	$K(x,y) = a\left[\exp\left(\frac{\gamma}{\|\|x-y\|\|^2+\sigma^2}\right) - 1\right]$
RBF	$K(x,y) = \exp(-a\|\|x-y\|\|^2)$
Exponential RBF	$K(x,y) = \exp(-a\|\|x-y\|\|)$

Figure 2.12: Different Kernel functions. Image taken from [3]

In this case, Hadid et al. used a second degree polynomial kernel function. Every LBP that was extracted in the first phase, is fed to the SVM and mapped in a high dimensional space and so a separating hyperplane is constructed. Given a new image a scanning sub-window is shifted across the image, extracting regional LBP_x features and classifying them based on the function below:

$$F(LBP_x) = Sign(\sum_{i=1}^{l} a_i y_i K(LBP_x, LPB_{ti}) + b) \tag{2.7}$$

where l is the total number of samples, LBP_{ti} is the LBP representation of the current i sample, b is the bias, y_i is 1 or -1 depending on whether LPB_{ti} is labeled as a face or not and a_i are the parameters of the SVM classifier and are calculated by:

$$\max(\sum_{i=1}^{l} a_i - 1/2 \sum_{i,j=1}^{l} a_i a_j y_i y_j K(LBP_{ti}, LBP_{tj})) \tag{2.8}$$

$$Subject\ \ to\ \ \sum_{i=1}^{l} a_i y_i = 0, \qquad 0 \leq a_i \leq C$$

Equation 2.8 is a quadratic programming problem and is standard procedure when using polynomial Kernel functions, while C is empirically computed and set and is called cost of the constraints violation. In essence depending on its value, the allowance of error increases or decreases inversely proportional to the complexity of the model. For more information look up the literature concerning SVM proposed above.

Chapter 3

Face Recognition

Face recognition is the task of matching a detected face from an input image to a face from a database of faces. Face recognition systems are in reality a combination of systems. The first stage of the system consists of a face detection and localization algorithm, which is used to locate the face in the image. Then, they process these faces to prepare them as input for the face recognition stage and finally analyze the face's pattern, geometry or features, depending on the algorithm, and matches it to a face from a face database. It is important to note, that face recognition has two main variations: a) face verification and b) face identification. Face verification is a one to one search and is of course a faster procedure. The system will compare the detected face to a single stored face and verify or not whether the detected one is the same as the stored one. Instead, a face identification system will search a database of many faces to identify a face detected in an input image. In the following sections, two popular and groundbreaking for their time algorithms will be explained.

3.1 Eigenfaces

Even though face recognition was first introduced by Wilson Bledsoe in the 1960s and was later addressed by a number of people, it wasn't until 1991 when Turk and Pentland[4] expanded on the idea of Eigenfaces proposed by Sirovich and Kirby[40] that the feasibility of an automated facial recognition system was indicated. In this section an explanation of the Eigenfaces algorithm will be provided, as well as the basic idea of PCA or Principal Component Analysis[41][42] since it is the core of the Eigenfaces algorithm. PCA is a dimensionality reduction method that given a large number of data samples in a high D dimensional space,

will seek to map them in a new multi-axis coordinate system of D' dimensionality, where $D' < D$ without losing the data structure and the information it may contain. This D' dimensional sub-space is formed by n linearly independent vectors $v_1, v_2, ..., v_n$, which are the basis of this sub-space. These vectors have very distinct characteristics, which will be explained below, and are called eigenvectors. Using this approach Turk and Pentland tried, given a sample of face images, to create a low dimension face sub-space spanned by eigenvectors, or as they are known eigenfaces.

3.1.1 Covariance Matrix, Eigenvectors and Eigenvalues

Given a data set of n samples with m seemingly independent attributes, it is only natural that these samples will be mapped in a m dimensional space.

$$A_{n,m} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vec{X_1} & \vec{X_2} & \cdots & \vec{X_m} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

where $\vec{X_1}, \vec{X_2}, ..., \vec{X_m}$ are $nx1$ vectors representing the attributes, and A is the matrix form of the data.

PCA's purpose is to find "hidden" correlations between the samples' attributes, that cannot be deducted intuitively, and use them to construct new m' independent vectors $[u_1,\ u_2,\ ...,\ u_{m'}]$ that span a sub-space in the m dimensional space, where the samples can be mapped without losing any significant information. These vector will be the eigenvectors or Principal Components. In order to calculate the Principal Components a matrix called covariance matrix is used, which contains numerical values depicting variances and covariances between the attributes. The covariance of the attributes x_1 and x_2 $Cov(x_1, x_2)$ measures how these variables change together or in other words move in the same direction. If the covariance is positive, it indicates that when one variable increases the other will also increase, and the inverse if it is negative. These values derive from the dot product of the A matrix and its transpose A^T, after a subtraction of the mean value μ from each attribute has been done. This

subtraction, centers the attributes around the origin and achieves a new mean value of 0. The mathematical interpretation:

$$\mu = \frac{\sum_{i=1}^{m} \vec{X}_i}{M} \tag{3.1}$$

where μ is the mean value

$$\Sigma = \frac{AA^T}{M} \tag{3.2}$$

where Σ is the covariance matrix (often denoted as C),

$$A = \{\vec{X}'_1, \vec{X}'_2, ..., \vec{X}'_m\} \text{ and } \vec{X}'_i = \vec{X}_i - \vec{\mu}$$

In the figure 3.1 below, three different examples of covariance can be seen. If the vectors forming the axis x, y are considered as the attributes and the black circles as the samples, it is easy to understand the concept of covariance. In the first example the covariance is negative since with the increase of the x attribute the y attribute decreases. They move in opposite directions. Similarly, the right example has a positive covariance, while the covariance in the middle examlpe is close to 0. It is to be noted that the covariance value does not indicate the ”strength” of change.

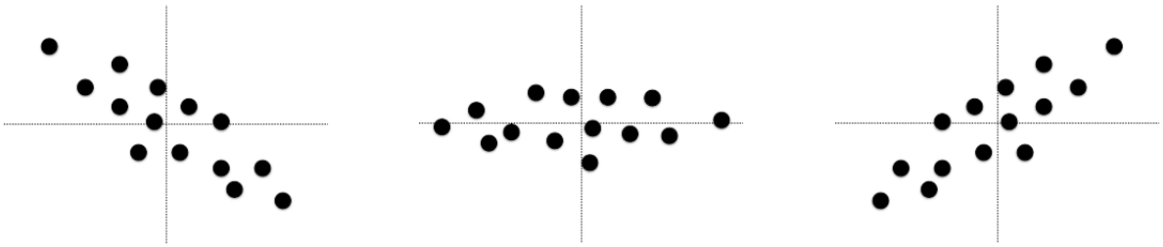


Figure 3.1: Covariance examples

In these spaces, if any random vector is multiplied by the respective covariance matrix Σ multiple times, a remarkable fact can be observed. The slopes of these random vectors will eventually converge into a value, and the vector with this slope value is the vector, or even better dimension, with the maximum variance. Maximum variance indicates the direction

where the data samples are most spread out. Any excessive multiplications with the covariance matrix mark no change on the slope of the vector, but only on its scale. This vector with maximum variance is the first eigenvector or Principal Component. The reason that variance along the eigenvector, which mathematically is the average of the squared distances of the projections of the data samples from the origin, is because it is a metric which indicates that the relative positions of the data and thus information is preserved.

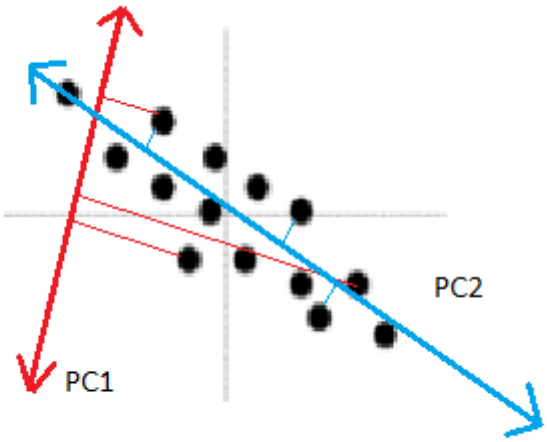


Figure 3.2: Visualizing Variance among two Principal Components

In the figure 3.2 below, it is easy to observe that $PC2$ is obviously on the direction that the data spread the most. The projections of the samples are minimized while keeping their relative positions. In contrast to $PC1$ where the projections of the samples lose their relative positions and information is lost, making $PC1$ a far "worse" eigenvector. In a real application the goal is to find the k, a number set manually, eigenvectors with the highest eigenvalues. Eigenvalues show the variance of the corresponding eigenvector. There are a number of ways to compute eigenvectors and eigenvalues, however in this thesis the most generic one will be presented. Given the $A_{n,m}$ matrix form of the data samples and its covariance matrix Σ, it is possible that m eigenvectors exist. To compute these $u_1, u_2, ..., u_m$ eigenvectors and their respective eigenvalues $\lambda_1, \lambda_2, ..., \lambda_m$ the following procedure is followed:

The eigenvalues are computed by the equation below:

$$\mathbf{det}(\Sigma - \lambda I) = 0 \tag{3.3}$$

For each eigenvalue λ_i found, calculate an eigenvector u_i :

$$\Sigma\vec{u_i} = \lambda_i\vec{u_i} \tag{3.4}$$

It should be mentioned that the eigenvectors should have unit length $\hat{u_i} = \frac{\vec{u_i}}{||u_i||}$ .After the eigenvectors are computed, the k eigenvectors with the highest eigenvalues and thus the highest variance are chosen as the new k-axis lower dimensional system where the data samples are projected. In a simplistic way it can be said that PCA builds new k attributes that summarize the original m ones, attaining a simpler environment where the data can be observed, losing as little information as possible. It linearly transforms the m dimensional data, in a linear combination of k eigenvectors. Each sample x_i can be represented as a sum of the mean μ and a weighted sum of the eigenvectors.

$$\vec{x_i} = \mu + w_1\vec{u_1} + w_2\vec{u_2} + ... + w_k\vec{u_k} \tag{3.5}$$

These weights are a unique combination for each unique sample and can be seen as a form of ID for this particular sample. This is an idea that led to the use of PCA and the eigenvectors in the field of face recognition.

3.1.2 Application to Face Recognition

In this section a step by step analysis of the eigenfaces algorithm developed by Turk and Pentland[4] will be presented. Given a set of M grey-scaled, centered and NxN pixel scaled training images $I_1, I_2, ..., I_M$, the first step is to transform each one to a N^2x1 dimensional vector $\vec{\Gamma_i}$. This is achieved simply by inserting each pixel of the image I_i into the vector $\vec{\Gamma_i}$ as a row (see 3.3). In succession, the mean face vector $\vec{\Psi}$ is computed

$$\vec{\Psi} = \frac{\sum_{i=0}^{M}\vec{\Gamma_i}}{M}$$

and subtracted from each vector $\vec{\Gamma_i}$.

$$\vec{\Phi_i} = \vec{\Gamma_i} - \vec{\Psi}$$

These averaged vectors form a matrix $A = \{\vec{\Phi_1}, \vec{\Phi_2}, ..., \vec{\Phi_M}\}$ from which the covariance

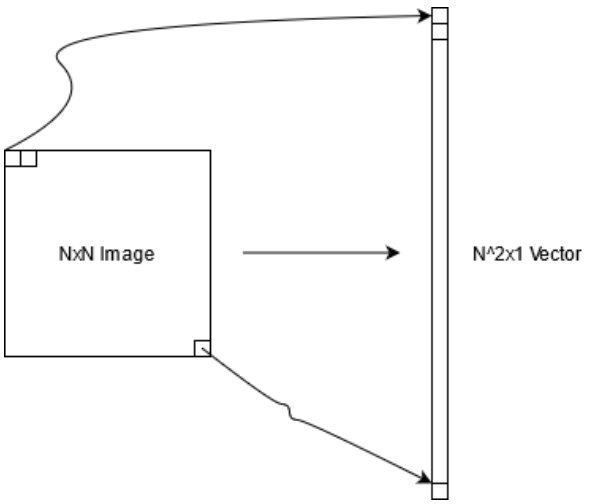


Figure 3.3: Image transform into vector

matrix Σ and thus the eigenvectors and eigenvalues will be computed. In essence $\vec{\Phi_i}$ contain the most prominent deviations from the mean, or in other words the dimensions that the samples differ most from the mean.

However a covariance matrix $\Sigma = \frac{\sum_{i=0}^{M} \vec{\Phi_i}\vec{\Phi_i^T}}{M} = AA^T$ of N^2xN^2 dimensions, since A is of N^2xM and A^T is MxN^2, would contain up to N^2 eigenvalues and eigenvectors of N^2x1 dimensionality. The goal is to find k eigenfaces or eigenvectors, which will make possible to represent all samples from the data set in a low dimensional space. Doing that from such a high dimensional covariance matrix is significantly inefficient. For example, if a training set consisted of 100 images of 512x512 pixel resolution there would be needed $k < 100$ eigenfaces to build a face-space. Even so, the computation of 512x512 = 262.144 eigenvectors and eigenvalues from a 262.144x262.144 covariance matrix would firstly be needed. To avoid this problem, a covariance matrix of reduced dimensionality $\Sigma' = A^T A$ can be used. It is known from linear algebra that these two covariance matrices, Σ and Σ' , have the same highest eigenvalues and eigenvectors. Σ' is a MxM matrix, since A^T is MxN^2 and A is N^2xM, that contains up to M eigenvectors of Mx1 dimensionality. As stated, these M eigenvectors are also contained in Σ covariance matrix and are the ones with the highest eigenvalues. Consequently $k < M << N^2$ eigenvectors can be calculated from Σ' and then mapped to the original N^2x1 dimensionality using linear algebra. Let $[u_1,\ u_2,\ ...,\ u_{N^2}]$ be the N^2x1 eigenvectors of $\Sigma = AA^T$ and $[v_1,\ v_2,\ ...,\ v_M]$ the Mx1 eigenvectors of $\Sigma' = A^T A$:

$$A^T A\vec{v_i} = \lambda_i\vec{v_i} <=> AA^T A\vec{v_i} = \lambda_i A\vec{v_i} <=> \Sigma Av_i = \lambda_i A\vec{v_i}$$

And according to equation 3.4 Av_i is eigenvector of Σ, thus $u_i = Av_i$. Therefore, after the calculation of the k v_i eigenvectors, they are mapped into the original dimensionality according to $u_i = Av_i$. These eigenfaces span the face-space.

Following the creation of the face-space is the the projection of the original samples so to be expressed as a linear combination of the eigenfaces. The weights ω_i corresponding to each projection to an eigenface are derived by the equation below

Figure 3.4: The training set and mean face of [4]

Figure 3.5: The corresponding eigenfaces-[4]

$$w_i = \vec{u_i}^T(\vec{\Gamma_i} - \vec{\Psi})$$

and are stored in a vector $\Omega_i^T = \{\omega_1, \omega_2, ..., \omega_k\}$. Thus the original image can be represented as a weighted sum (see 3.5).

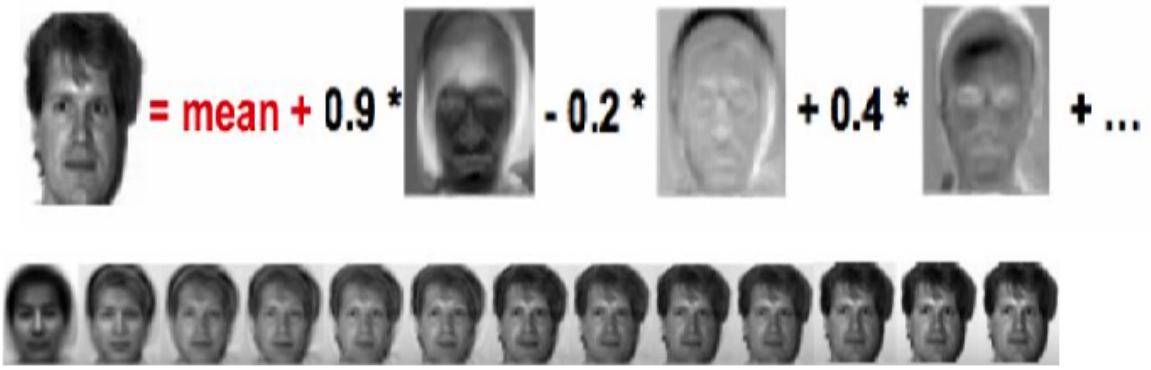


Figure 3.6: Original Image expressed as linear combination of eigenfaces. Each Image comes from an addition to the sum of a weighted eigenface. Image taken from [5]]

Detection and Recognition

Having the face-space and the weight vectors of the training samples computed, the only question to be answered is how a new test image will be classified and identified. A test image I_{test} is transformed into vector form, then projected into the k dimensional face-space and has its corresponding weight vector $\vec{\Omega_{test}}^T$ computed. Then a simple method of classification, the Euclidean Distance[4] is used to determine whether the test image is a face, and if yes what is its identity. First the distance ϵ_δ of the image and the face-space is computed. This distance equals to the distance of the averaged test image $\Phi_{test} = \Gamma_{test} - \Psi$ and its projection to face space $\Phi_{proj} = \sum_{i=0}^{k} \omega_i u_i$.

$$\epsilon_\delta = ||\phi_{test} - \Phi_{proj}||$$

Then the distances ϵ_k between the weight vector of the test image and the weight vectors of all samples, called now face classes, are calculated in succession and the minimum of these distances is kept.

$$\epsilon_k = ||\Omega_{test} - \Omega_k||$$

If ϵ_δ is lower than a manually set threshold θ_δ and ϵ_k is lower than an also manually set threshold θ_k then the image is classified as a face and is identified as the person whose corresponding weight vector was Ω_k (Not to be mistaken with the last kth eigenvector. It could be any eigenvector from the k). If θ_k is over the threshold but $\epsilon_\delta < \theta_\delta$, then the person is classified as an unknown face. If $\epsilon_k > \theta_k \; and \; \epsilon_\delta > \theta_\delta$ then it is classified as a non person. In the occasion that an image is not classified as a face but is near a face class, it is many times considered as a false-positive. These threshold play a major role on the performance of the algorithm. Tweaking the threshold is always a trade off between accuracy and false-positive detections[43].

Limitations

Although eigenfaces was a major breakthrough in the field of face recognition, it is now vastly outperformed by many systems and is very rarely used in real life automated systems.

The biggest drawback is that it is very scale dependant. Its performance and accuracy rapidly drop when a large scaled image is presented. Furthermore each test image needs to be pre-processed, scaled and has its background removed or processed. It is also demonstrating drop on performance and accuracy in significant illumination and pose changes.

3.2 Fisherfaces

Fisherfaces was an algorithm introduced by Belhumeur et. al. at 1997[44]. The authors were greatly influenced by the Eigenfaces algorithm that had been introduced some years before. Similar to the Eigenfaces method, they also used linear projection to project the high dimensional face image into a low-dimensional sub-space, that is called face-space, and used the projection's distance from already stored data to identify the person. However, they did it by using both PCA, exactly as it was used in the Eigenfaces algorithm, and a method heavily used in statistics called Fisher's Linear Discriminant analysis - FLD[45]. The main idea behind was that, even though Eigenfaces was an excellent algorithm to represent a face into a sum of Principal Components, its detection was greatly reliant on the person's facial expressions and the image's lighting conditions. Two images of the same person but with different facial expression and lightning could vary more than the picture of two different people under the same conditions. Since Eigenfaces sole purpose was to maximize variance, it took into account the lighting or facial variations of the same person in the construction of the Principal Components. As a result, Eigenfaces proved to be an excellent image reconstruction method, but its discriminative power was not consistent. In the following section the Fisherfaces algorithm will be explained.

3.2.1 Fisher's Linear Discriminant analysis - FLD

The major difference between the two dimensionality reduction methods, FLD and PCA, is that FLD is utilizing the concept of classes. PCA viewed each image sample individually, while FLD discriminates the samples by assigning a class to each image. Images of the same person, even in different conditions will be assigned under the same class. The first step in FLD is to calculate the scatter matrices, similar to the covariance matrix calculation step in PCA. There are two different kinds of scatter matrices: The interclass or Between class scatter matrix and the intraclass or Within scatter matrix. Between class scatter matrix S_B shows how

much the mean of each class μ_j varies from the total mean, while Within scatter matrix S_W shows how much the samples x_i of a class c vary from their mean μ_j.

The mean of a class is:

$$\mu_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_i \tag{3.6}$$

where N_j is the total number of samples that belong in that class. The total mean is defined as:

$$\mu_j = \frac{1}{\sum_{i=1}^{c} N_i} \sum_{j=1}^{c} N_j \mu_j \; = \; \frac{1}{N} \sum_{i=1}^{N} x_i \tag{3.7}$$

where c is the total number of classes. The Between class scatter matrix:

$$S_B = \sum_{j=1}^{c} N_j (\mu_j - \mu)(\mu_j - \mu)^T \tag{3.8}$$

The Between class scatter matrix is very similar to the covariance matrix which is defined in 3.2. In fact, it is just an unnormalized version of the covariance matrix, or better the covariance matrix is a scaled interclass matrix. The Within class scatter matrix:

$$S_W = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (x_i - \mu_j)(x_i - \mu_j)^T \tag{3.9}$$

FLD, in similarity with PCA, will try to maximize the interclass scatter with the difference, that it will keep intraclass scatter minimized. In other words, it will attempt to find the sub-space, where the projections of the samples will have maximized scatter, as long as their within class scatter is minimized. An intuitive example of two classes in a two dimensional space can be seen in the figure 3.7 below:

It can be seen that the variance is higher when using PCA, however the within class scatter is not taken into account. FLD will try to maximize the Between class scatter, in this

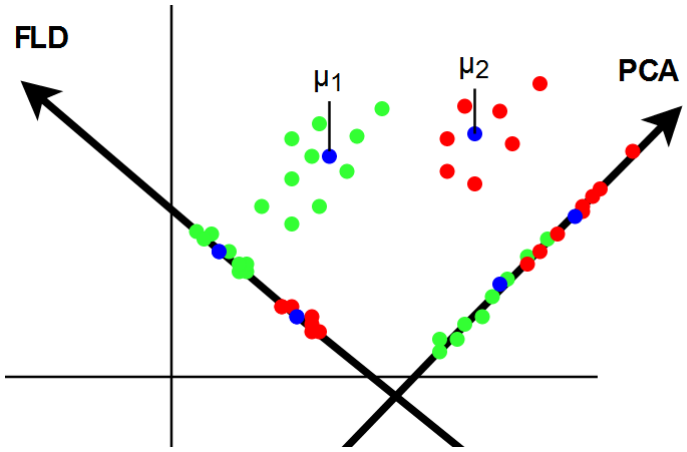


Figure 3.7: A two dimensional example of 2 classes. The green and red points are the samples of each class while blue points represent their mean value. It can be seen the the variance with the PCA algorithm is greater, however the in class scatter is much smaller in the projection with the FLD.

case $\mu_{1,p} - \mu_{2,p}$ while keeping the Within class scatter minimum. Given that x_i is a high dimensional sample and $y_i = W^T x_i$ is the projection to the sub-space then the projected samples' Within class scatter of class j will be equal to:

$$
\begin{aligned}
&= \sum_{i=1}^{N_j}((W^T(x_i - \mu_j))(W^T(x_i - \mu_j))^T \\
&= \sum_{i=1}^{N_j} W^T(x_i - \mu_j)W(x_i - \mu_j)^T \\
&= W^T \sum_{i=1}^{N_j}(x_i - \mu_j)(x_i - \mu_j)^T W \\
&= W^T S_J W
\end{aligned}
$$

In a similar manner all the classes' Withing scatter are calculated and the total Within class scatter matrix will be their sum and equal to $W^T S_W W$. Following the same process to find the Between class matrix for the projected means, where $\mu_{i,p} = W^T \mu_i$, it is found that it is equal to $W^T S_B W$. The ratio of $\frac{|W^T S_B W|}{|W^T S_W W|}$ is called the fisher criterion and it is this ratio that needs to be maximized. So the optimal projection $W_{opt} = [w_1, w_2, \ldots, w_l]$ is derived by:

$$W_{opt} = \underset{W}{\operatorname{argmax}} \frac{|W^T S_B W|}{|W^T S_W W|} \tag{3.10}$$

Then, the derivative of 3.10 with respect to W is taken and solved for $\frac{dW_{opt}}{dW} = 0$, which leads to the generalized eigenvalue problem:

$$S_B W_i = \lambda S_W W_i \tag{3.11}$$

where W_i are the eigenvector corresponding to the highest eigenvalues. It is to be noted that there are at most $c-1$ eigenvalues and eigenvectors. If S_W is invertible then the classical eigenvalue problem can be derived by multiplying 3.11 with S_W^{-1}.

$$S_W^{-1} S_B W_i = \lambda W_i \tag{3.12}$$

However, a singularity problem occurs. S_W is a singular matrix and thus can not be inverted. This occurs, because in face recognition problems the features of the images, which in this case are the pixels outnumber the samples by a large margin. This leads to a very large number of projection combinations and one will almost always find a projection where the Withing class scatter will be zero and thus the ratio will become seemingly infinite.

To solve this problem the authors used PCA in order to interpret the images with a small number of new compact features, as it was explained in the previous section, hence reducing the dimensionality. Then they proceeded to use FLD to reduce the dimensionality even more, while keeping the Within class scatter minimized. As a result, the final W_{opt} projection is defined as:

$$W_{opt} = W_{fld} W_{PCA} \tag{3.13}$$

where

$$W_{pca} = \underset{W}{\operatorname{argmax}} |W^T S_T W|, \; W_fld = \underset{W}{\operatorname{argmax}} \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|} \tag{3.14}$$

An example of Fisherfaces can be seen in the figure below. It can be seen how Fisherfaces differ from Eigenfaces in the sense that they retain discriminative class features like a beard, hair, glasses or even facial expressions.

Figure 3.8: An example of 4 Fisherfaces taken from a Dataset of 100 classes. Image taken from [6]

Improvements and Limitations

It has been shown both by the original paper[44] and other comparative tests that followed [46][47][48] that the Fisherfaces algorithm is both faster and more accurate than Eigenfaces. It it robust to lighting and facial expression changes in a high degree. Improved versions of this algorithm also followed. Some of them are the Laplacian faces[46], the improved-LDA face recognition[49] and the Fuzzy-Fisherfaces[50]. These algorithms achieved low error rates and in an adequate processing time, however they fell short in comparison with the deep learning algorithms that were introduced some years later.

Chapter 4

Convolutional Neural Networks

After the great success of Viola and Jones in 2001, machine learning algorithms became the standard for face detection and recognition problems. Each subsequent year new machine learning algorithms were introduced or improvements on already existing ones were made improving the detection rates. Despite the progress many limitations, some of which were mentioned in previous chapters, remained unresolved in the following years. Nevertheless with the beginning of the new decade a particular subfield of machine learning eliminated all its competition and dominated every field in object detection and image classification, demonstrating very exciting results. This subfield was deep learning, which are algorithms that utilize the technology of neural networks. More specifically in object detection and image classification, and thus face detection and recognition, a special kind of neural networks, called convolutional neural networks - CNN is used. In this chapter a brief introduction to neural networks and a more thorough explanation of CNN will be presented, as well as a face detection and a face recognition algorithm using these networks will be explained.

4.1 Basic idea of Convolutional Neural Networks

Convolutional neural networks met a huge surge of popularity in Computer Vision in the early 2010s. However the conception of the basic idea was made much earlier. This idea derived from the study of biological vision made by Hubel and Wiesel in 1959[51]. They experimented on cat brains with hope of understanding how their visual system worked, and which visual stimuli activated the neuron cells in their primary visual cortex area. They discovered that activation of different cells of distinct complexities. The cells of least complexity

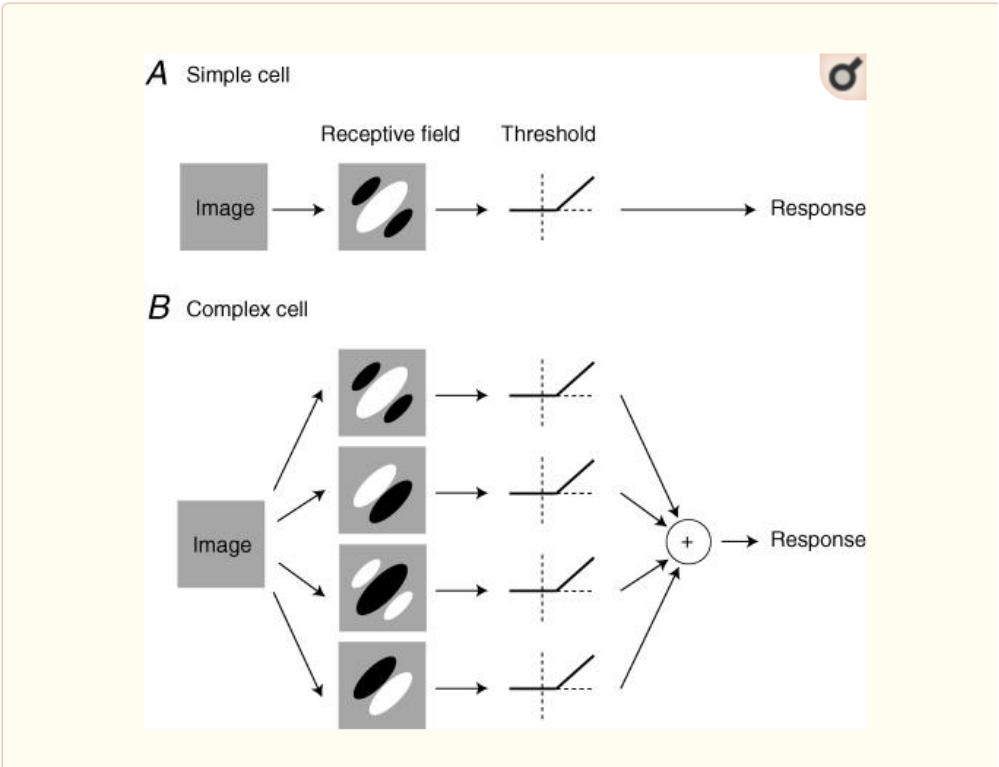


Figure 4.1: Model of Movshon&Tolhurst [7]. Image taken from [8]

called simple cells were of great importance and were responsive to edges and lines of different orientation. Simple cells have distinct receptive fields, meaning that the position of object on the image is very crucial on whether they will activate or not. In contrast more complex cells were spatial invariant and always activated. This observation led to the proposal that complex cells actually use the outputs of multiple simple cells and thus the system can be characterized as hierarchical. In the following years many experiments were conducted, and a mathematical analysis of what is happening was investigated[7]. A simplistic depiction of this analysis can be seen in figure 4.1.

These discoveries were the motivation for Fukushima[52] to build a neural network, simple to complex hierarchical model, which consisted of S-cells - Simple cells and C-cells - complex cells. These cells of course were mathematical operations aiming to recognise numbers. However this research was greatly constrained due to the low computational power of computers at the time. Nonetheless his work was the one to inspire the creation of the first modern Convolutional Neural Network made by Lecun et. al.[9]. This CNN would become the foundation for all CNN used in Computer Vision later and its architecture can be seen in figure 4.2. Details upon the technical details of CNNs will be presented later.

The purpose of this CNN was to recognize handwritten number patterns in order to sort

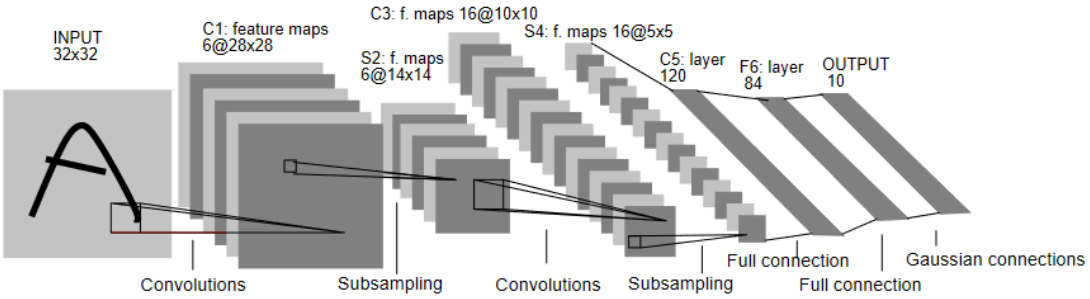


Figure 4.2: The architecture used by Lecun et. al. [9]

mail automatically at the post office. The basic idea of detecting simple features like edges, and combining them to more complex features remained the same.

For more than a decade research and updates were done on CNNs, but machine learning algorithms remained more feasible and more accurate for most problems in the field of Computer Vision. Deep learning algorithms' performance is heavily depended on the training set. They need incredible amounts of Data and computational power to be efficient. With the increased interest on the field of Computer Vision, Data sets with labeled images became more and more necessary. This led to the creation of large Data sets like FRGC, CIFAR-10, FERET and ImageNet. ImageNet was completed in 2009 by Fei-Fei-Li and a research team and it was by far the largest dataset at that time, containing more than 14 million hand-annotated pictures. At the same time Graphics processing units boosted the computational power by a large margin. All these conditions led to the creation of AlexNet, a CNN architecture created by Alex Krizhevsky et al. [53] which proceeded to mark by far the lowest error rate at the ImageNet Image classification challenge in 2012. Since then CNN architectures dominated in object and face detection and recognition demonstrating good performance and excellent detection rates, becoming the golden standard in Computer Vision.

4.2 Introduction to Neural Networks.

As stated above, CNN are a special type of neural networks. A neural network is essentially a series of mathematical operations that given a fixed size input vector of n variables $[x_1, x_2, ..., x_n]$, outputs a vector of k numbers. The main processing unit of a neural network are the nodes, which are also called neurons and are mostly packed in layers. Each neuron is connected to all other neurons of a neighboring layer, but do not communicate with neurons

of the same layer. These connections are called synapses and have a weight w_i assigned. The neuron internal value is calculated by the summation of the weight*feature factors plus a bias b_j. This value will be filtered by an activation function ϕ that is positioned after each neuron, then multiplied by the corresponding weight w_i of the connection line and passed forward as input into the next neuron.

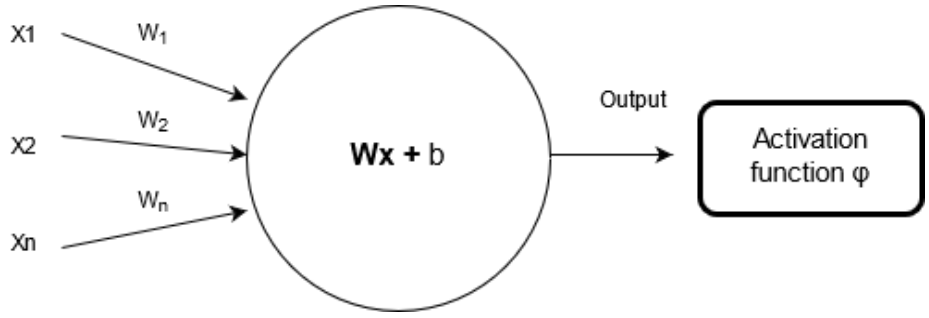


Figure 4.3: A simple neuron in a neural network

The value of such a neuron cell can be derived by the following equation: $\phi(\sum_1^n W_n x_n + b) = \phi(\vec{W}\vec{x} + b)$ where n is changing depending on the layer.

A simple neural network consists of 3 type of layers: the input layers where input is mapped, the hidden layer where processing is done and the output layer[54]. A simple neural network architecture can be seen in figure 4.4.

Loss function

Similar to every learning system the most critical part in neural networks is the training process. Thousands or even millions in more complex systems of labeled training samples are needed to train a network. Each label for any training sample contains the desired output y_i for this particular sample and thus the error is ϵ is $\epsilon = y_i' - y_i$ where y_i' is the output of the sample. In order to determine how well the system is fitting the data the Loss L is calculated using a loss function L_i [55][56]. Loss over a set of N samples is defined as:

$$L = \frac{1}{N} \sum_{i}^{N} L_i(f(X_i, W), y_i) \tag{4.1}$$

where X_i is ith sample and f is the existing model.

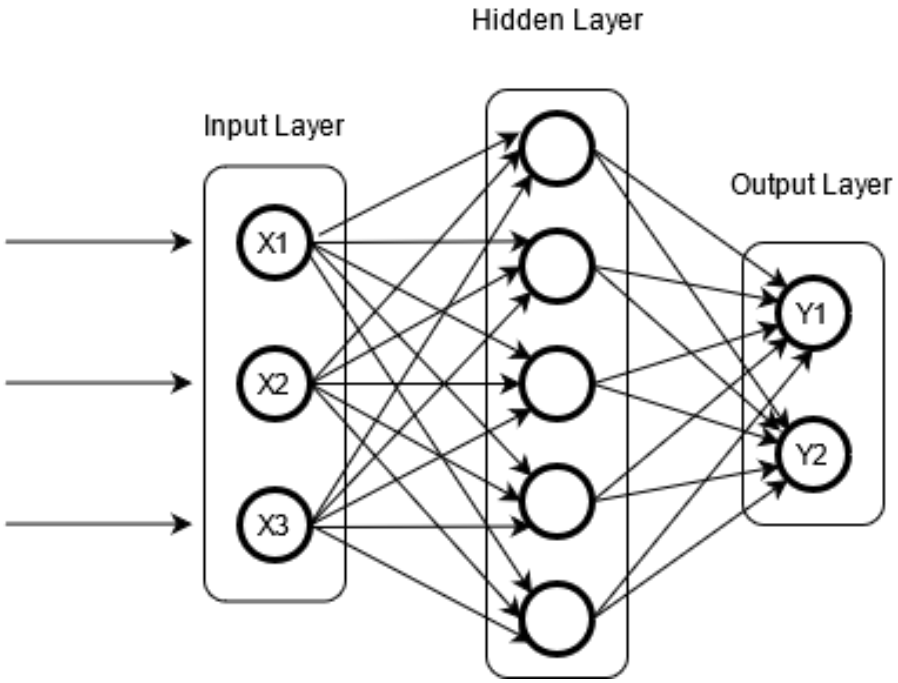
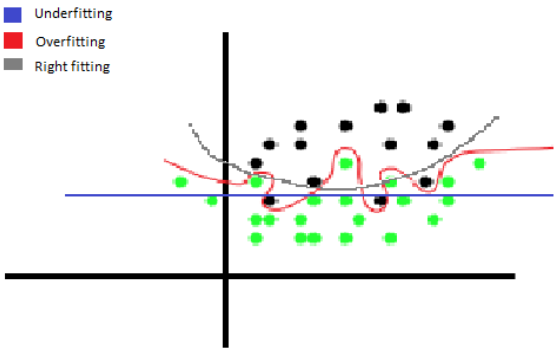


Figure 4.4: A 2-layered artificial neural network

The goal is to minimize the Loss by adjusting the Weights and bias. Since the input layer has no tunable variables it is excluded from the layer count and a network like 4.4 is considered a 2-layer network. In literature these kind of networks are referred as ANN - Artificial Neural Networks or MLP - Multilinear Perceptron. Mathematically, minimizing the Loss means to adjust the variables of the multi-variable and differentiable loss function in order to reach the global minimum. In essence it is becomes an optimization problem, which is often solved in neural networks by using the technique of gradient descent. Gradient descent is an iterative process where in each iteration calculates the partial derivatives of Li with respect to each variable w_i, b_l and adjusts these variables to the direction of the steepest descent. This process begins from the output layer and continues backwards to each layer. For this reason it is called backpropagation, in contrast to the forward passing or forward propagation of values in the first stage of the system. It is important to mention that most real applications add a term to the Loss, called regularization term, in order to avoid overfitting.

Overfitting[57] occurs when in the process of minimizing the Loss, the variables are adjusted to such extent to the details and even the noise of the training samples that its performance when testing new test subjects drops. Regularization attempts to prevent that by reducing the variance of the samples and penalizing the complexity of the system. It is in essence a trade off between the fitting of training samples and performance. Common regularization techniques are L1, L2, Dropout and Max norm regularization.

Figure 4.5: Overfitting problem displayed by red



Backpropagation

As stated above, the process that neural networks use to adjust the weights is called backpropagation. The architecture of these networks makes this process very efficient, since it utilizes the chain rule for the partial derivatives calculation and is able to reuse computed values for subsequent steps while having a fairly simple algorithmic design. The most common algorithm used is Gradient descent - GD, Stochastic Gradient descent - SGD and Minibatch Stochastic Gradient - MSGD. All these algorithms uses the same basic operation which will be explained below. Suppose the following part of figure 4.4

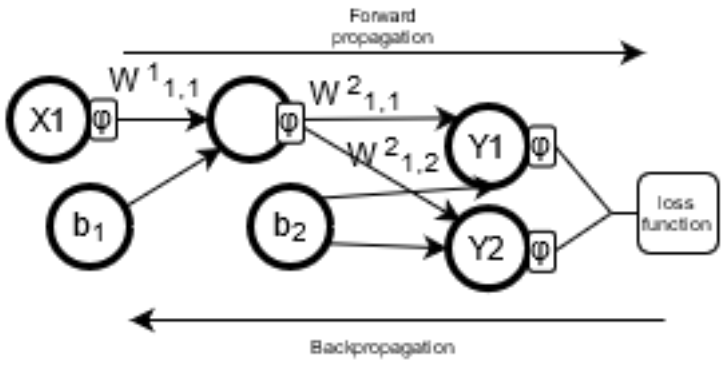


Figure 4.6: A part of 4.4 figure. Here the biases and activation functions are depicted.

Let b_1, b_2 be the biases of the respective layers, $W^l_{i,j}$ be the weight of the ith to jth node of layer l and s^l_i, z^l_i be the ith node's value before and after the activation function ϕ respectively. The chain rule states that given the differentiable functions $w = f(x)$ and $e = g(f(x))$,

then the dependence of the f function on the changes of x can be expanded and interpreted as dependence on the independent variable t, since x is dependant on t. Mathematically it translates as:

$$\frac{de}{dx} = \frac{de}{dw}\frac{dw}{dx}$$

This rule can be expanded on multivariable functions and partial derivatives. Loss functions are multivariable functions since they depend on multiple weights, which also depend on the weights of previous layers' connections. To adjust the each weight towards the direction of the steepest descent, it is needed to calculate the partial derivative of L with respect to each weight starting with weights on the output layer[58]. Working on the example above and starting from $W^2_{1,1}$, so the weight on the second layer connecting the first node - $y1$ to the previous layer's first node.

$$\frac{\partial L}{\partial W^2_{1,1}} = \frac{\partial L}{\partial z^2_1}\frac{\partial z^2_1}{\partial s^2_1}\frac{\partial s^2_1}{\partial W^2_{1,1}}$$

The calculation of $\frac{\partial L}{\partial z^2_1}$ is trivial since for given real values it becomes a simple numerical problem. The second term $\frac{\partial z^2_1}{\partial s^2_1}$ is also a numerical problem. It is to be noted that since $z^l_i = \phi s^l_i$, the sigmoid function is a very often used since its derivation is very simple $\sigma'(x) = \sigma(x)(1-\sigma(x)$ and it also keeps the values between [0,1], which is very useful for predicting models. Finally the third term $\frac{\partial s^2_1}{\partial W^2_{1,1}}$:

$$\text{For every node } s^l_j = \phi_{l-1}(\textstyle\sum_{i=0}^{k} W^l_{i,j} s^{l-1}_i) \; + b_l$$

where k = number of nodes in the $l-1$ layer, the partial derivative with respect to $W^l_{i,j}$ equals

$$\frac{\partial s^l_j}{\partial W^l_{i,j}} = \phi s^{l-1}_i = z^{l-1}_i$$

So $\frac{\partial L}{\partial W^2_{1,1}}$ is calculated and in similar way all the other partial derivatives, in this example $W^2_{1,2}$, of the same layer can be computed. The updated weights derive from the formula $w' = w - \alpha\frac{\partial L}{\partial w}$ where α is called learning rate and is a trade-off value between learning time and

optimal learning. A common value is $\alpha = 0.1$. Going backwards to the previous layer, the first partial derivative to be calculated is $\frac{\partial L}{\partial W^1_{1,1}}$.

$$\frac{\partial L}{\partial W^1_{1,1}} = \frac{\partial L}{\partial z^1_1}\frac{\partial z^1_1}{\partial s^1_1}\frac{\partial s^1_1}{\partial w^1_{1,1}}$$

The only difference to the first operation is that $\frac{\partial L}{\partial z^1_1}$ is now affecting two nodes, h1 and h2. So:

$$\frac{\partial L}{\partial z^1_1} = \frac{\partial L}{\partial z^2_1}\frac{\partial z^2_1}{\partial s^2_1} + \frac{\partial L}{\partial z^2_2}\frac{\partial z^2_2}{\partial s^2_2}$$

Using the same process and reusing some already calculated values, the weights of layer 1 are updated. In a similar way the biases are updated using the chain rule and backpropagation. The difference between GD, SGD and MSGD is that GD updates the variables after all samples are test, SGD after each sample (so one full iteration) and MSGD after a predetermined batch of samples is tested. Depending on the algorithm choice, time or better weight optimization is gained[59].

It is clear that the more inputs are equivalent to more weights and thus the training becomes more and more time and power consuming. For high dimensionality inputs like images, this simple neural network architecture would be very inefficient. For an image of 256x256x3 dimensions, the weights required for the first hidden layer would be 196608 times the number of neurons. To solve this problem, the full connectivity of the neurons is removed and a special type of network is used called convolutional neural network.

4.3 Convolutional neural networks

The essential characteristic that differentiates the convolutional networks is the implicit assumption that the input will be consisted of images or more generally a three dimension input - width, height and depth. Therefore these type of architectures are ideal for object detection and recognition. A CNN can be viewed as a two - stage network, where in the first stage features are extracted from the image and in the second stage, probabilities of objects being in the image based on the extracted features are computed as output. Furthering this view, each stage is comprised of layers and thus, as any neural network a CNN can be

viewed as a sequence of different layers. The most common layers found in a CNN are the a) the convolutional layer b) the activation layer (also found as ReLU layer in literature) c) the pooling layer and finally d) the dense or else the fully connected layer. Each of these layers has a distinct purpose. In short the convolutional layer is where most of the computations heavy processes and features extraction is done, the activation layer's purpose is for optimized and faster training, the pooling layer is responsible for dimensionality reduction of the data and the dense layer for computing the output and classifying. All of these layers will be explained more thoroughly below.

4.3.1 Convolutional Layer

The core operation in this layer is Convolution. Convolution in CNN is usually occurring when an input X is convolved with a kernel of weights w to produce an output array. It is essentially a matrix multiplication or else a dot product between arrays. More generally convolution is defined as:

$$(f \circledast g)(t) = \int_{-\infty}^{\infty} f(r)g(t-r)dr$$

and can be intuitively thought as a way of merging two pieces of information. The neurons in the Convolutional layer are connected only to a specific region of the input (see figure 4.7, in contrast to the normal neural networks where they were fully connected.

Different filters are used in order to detect edges, lines, blobs of a certain color etc. These filters could be Haar, LBP or similar like kernels that were mentioned in chapter 2. Each filter, which in essence is a set of weights, is slided with a stride S across the whole input and convolves with a local region to produce a dot product output. Each neuron is connected only the the region convolved by the filter and the corresponding input and this is called receptive field of the neuron. The receptive field is equal the the size of the filter. Filters' height and width have no strict value limitations, however they must always have a depth equal to the input's, which in images is three (RGB). Suppose an input image of 5x5x1 dimensions as one of the three slices of the total input and a slice of the filter with 3x3x1 dimensions with $S = 1$ (see figure 4.8). The output will be a 3x3 array which is called feature or activation map.

Even if the depth dimension was included, the size of the activation map is not dependant

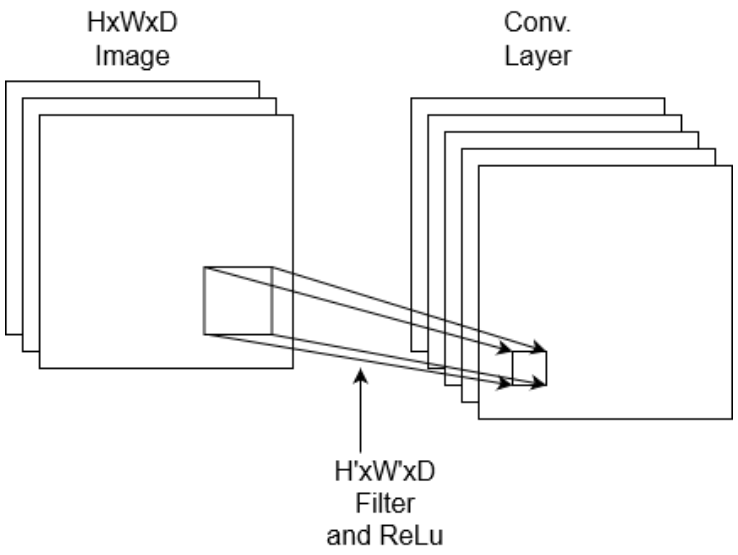


Figure 4.7: Convolutional Layer

of the depth and would remain the same. It is important to notice that in each activation map, each neuron is related to the same set of weights that are contained in the filter. So for the above example and accounting the depth, each neuron has 3x3x3 weights plus one bias which equals to 28 parameteres. If it was a normal neural network it would require $5x5x3 + 1 = 46$ parameters. Even from this extremely low dimensional example it can be deducted that the difference in parameteres is significant.

As seen in the above example, the activation map is of lower dimensions than the input. Consecutive convolutional layers or a stride of higher value could very rapidly lead to low dimensions. To counter this problem, a technique called zero-padding is used. Zero-padding is the addition of zero values rows and columns with the purpose of manipulating the dimensions of the activation map and avoid losing information. For example the addition of two zero valued rows and columns at the top, bottom, left and right of the image in figure 4.8 would result in a 5x5 activation map.

The activation map's dimensions can be calculated by the formula:

$$(W - F + 2P)/s + 1$$

where W is the size of the the input (considering it is rectangular), F is the size of the receptive field of the neurons, S and P is the zero padding. F, P, S are called hyperparameters and are values that are manually set that greatly affect the performance of the system.

The final hyperparameter and thus the last variable that affects the convolutional layer's final dimension, is the number of filters used. For each different filter, an activation map is

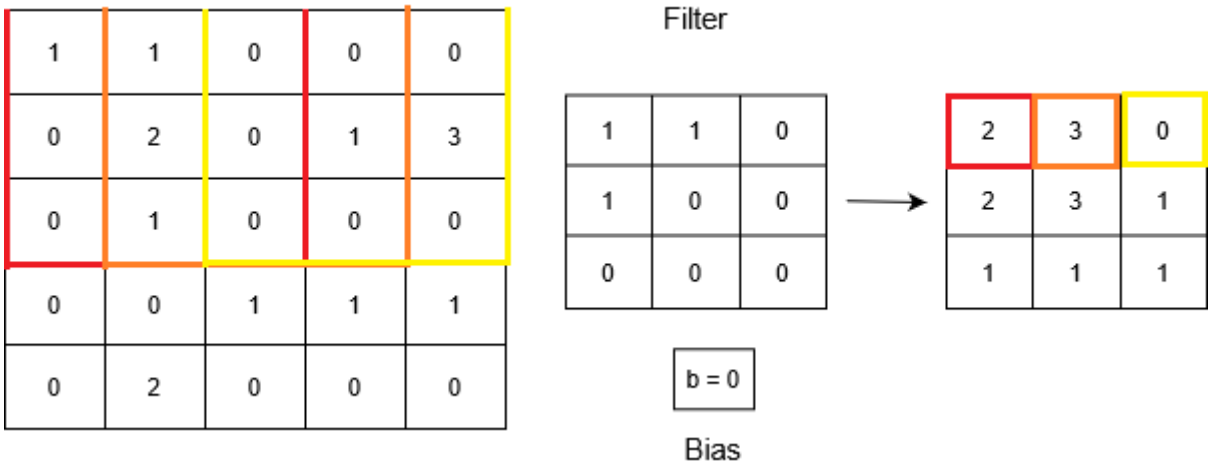
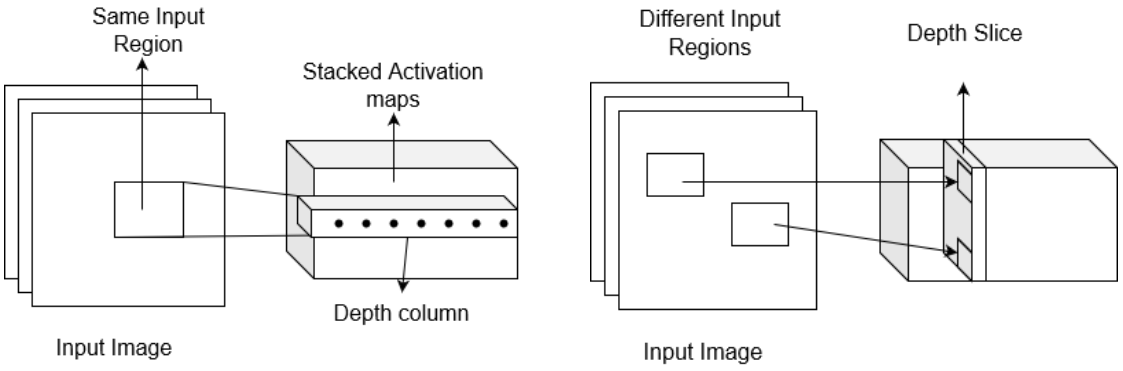


Figure 4.8: A 2D example of a 5x5 Image and a 3x3 filter with bias = 0, stride = 1 and no zero padding. The color-bordered areas are the first 3 dot products that correspond to the 3 first, also color-bordered values in the activation map.

produced and the Convolutional layer's depth is a result of all these activation maps stacked one upon another. Neurons along the depth dimension are essentially connected to the same input region and are called a depth column. Likewise neurons of same depth dimension residing on the same 2D activation map plane share the same receptive field and thus share the same weights. Each activation map is referred as a depth slice in the convolutional layer. Consequently neurons of a depth column try to find different features in the same input region, and neurons of a depth slice try to find the same feature along of all the input.



(α□) Depth column **(β□)** Depth slice

Figure 4.9: Neurons of a depth column have the same input region, while neurons of a depth slice share weights.

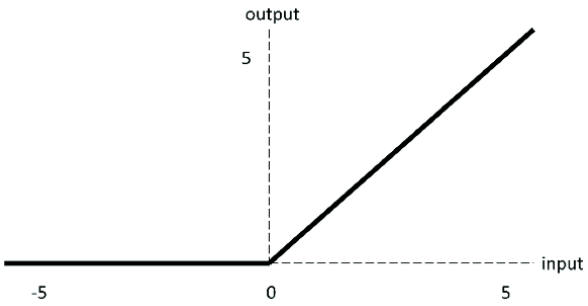
The most typical values for F, S and P are $3, 5$ $s = 1, 2$ and $P = 1$. It is important to choose these values in such a combination, so that the output dimensions are integers.

In symmetry with the simple neural networks, the forward pass the main purpose of the convolutional layer is to calculate and forward information the next layer, until an output is produced. Furthermore like the weights of synapses had to undergo training, Convolutional layer's three dimensional weights must undergo training too. This is achieved in a similar manner with the method of backpropagation and the use of the chain rule.

After the operation of convolution is done, the activation function operates. In the literature it is sometimes depicted as a separate layer after the Convolutional layer, but is often integrated in the operation of Convolution. The activation function that is dominating in most CNN architectures is the Rectified Linear activation function - ReLU.

The advantage of ReLU in deeper systems like CNN is reduced likelihood of the vanishing gradient effect. A significant problem that common used activation functions like the sigmoid or the hyperbolic tangent are susceptible to. Moreover ReLU discards negative values resulting in more sparsity, while still adding non-linearity to the system succesfully. All these advantages lead to a faster and better training and this is the reason that ReLU has almost become a default activation function for CNN. It is defined by:

Figure 4.10: ReLU function.



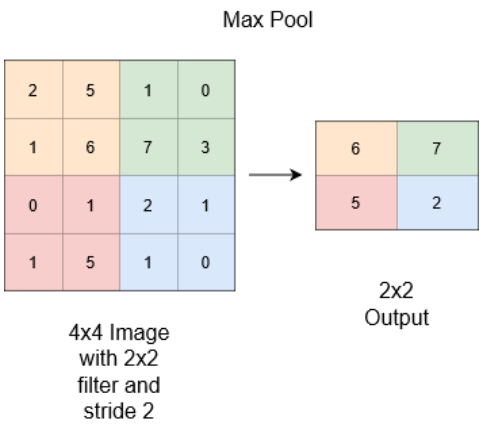
$$h = \max(0,a)$$

where a is the value of a neuron. More on activation functions can be found in the following literature [60][61].

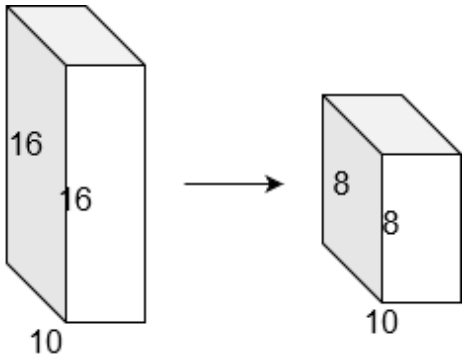
4.3.2 Pooling layer

Pooling layers are inserted after every few convolutional layers. Their purpose is to reduce the number of parameters of the system and thus the dimensionality, leading to shorter training times and reduced chances of overfitting. Pooling layer has 2 hyperparameters but no true parameters and this is the reason it does not impact the training negatively. These hyperparameters are the filter size F and the stride S. Pooling also uses the technique of

sliding window, sliding a filter along the input resulting in a downsampled output but with little to no information lost. Max pooling is the most common pooling method and common filter size and stride is $2x2$ and 2 respectively. Low dimension filters with no zero padding are preferred so as not to destroy too much information. It is to be noted that pooling operates on the activation maps and does not modify the depth of the input. A max pooling example can be seen below:



(α) Max pooling operated on a 4x4 Activation map with a $2x2$ filter and stride 2.



(β) Same filter applied on a 3D convolutional layer. The depth does not change, however the width and height are halved.

Figure 4.11: a) Max pooling 2D example b) Max pooling on a 3D example

Apart from max pooling, general pooling or other pooling methods can be used too. However experiments have shown that max pooling has by far demonstrated the best results among the pooling methods. This is mainly attributed to the fact that general pooling, which is the second most common method, tends to smooth out edges and hence lose feature information. There are implementations that prefer to replace pooling with bigger stride in the convolutional layer in order to downsample. More information about the pooling methods can be found in [62][63].

4.3.3 Fully Connected layer

This is the last part of the CNN and where the classification of the input happens. The fully Connected layer is in reality a simple Artificial Neural Network. It is comprised of an input, a hidden and an output layer and all the neurons are fully connected to the neurons of the next layer. The input is the flattened output of the last layer of the CNN. Flattening is simply the process of converting the three dimensional output into a one dimensional vector, since ANN

require simple values as input. These values are then processed by the hidden layers, where the activation function is also usually ReLU and fed to the output layer. The output produced a probability vector and its size is dependant on the number of classes that were used to train the network. The only detail that needs to be emphasized on is that the output layer does not use the ReLU or sigmoid activation function, but the softmax activation function. Softmax is mathematically defined as:

$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

where $\vec{x}$ is a vector of K real numbers. In essence the softmax function normalizes a vector x with K real values into a probability distribution of K probabilities $f_I(\vec{x}$. Each probability has value in range of $(0, 1)$ and the sum of these probabilities is 1, a characteristic that is very useful when classifying images.

4.4 Face detection with CNN

One the most innovative and groundbreaking deep learning face detection and alignment algorithm was introduced in 2016 by Zhang et. al.[10]. It is known as MTCNN or Multi - task Cascaded Convolutional Neural Network. Many deep learning detection algorithms that were developed after the popularization of CNN, like YOLO, SSD or R - CNN demonstrated high accuracy in detecting faces and generally in classifying objects. However Zhang et. al. reduced the multi - class detecting problem into a binary problem of face detection to achieve even greater performance. This resulted in less complex and smaller in size filters, which led to less computing requirements. Furthermore they emphasized on the correlation between facial landmarks and bounding boxes and the need for an online hard sample mining methods, since it was usually done manually and hindered progress. In this section, the MTCNN algorithm as well the methods it is utilizing will be explained thoroughly.

4.4.1 MTCNN architecture

MTCNN is consisted of three different convolutional networks and each one has a distinct task to complete. The networks are called P-net or Proposal Network, R-net or Refine network and O-net or Output Network. P-net's purpose is to rapidly propose bounding boxes

around probable face areas feed forward them as input to R-net. R-net's purpose to refine these bounding boxes and construct new bounding boxes with higher accuracy and also pass them as input to the next network, O-net. O-net will calculate the final bounding box and also find five key facial features. One can make the connection of this cascaded system to Viola - Jones algorithm, where the first stages were meant to be quick to detect possible face areas and the latter stages had the job to classify with higher accuracy of whether the probable areas were indeed a face or not. Following there is an explanation of each network and of the methods of Image pyramid, Bounding box regression, Non Maximum Suppression - NMS and Intersection over Union - IoU.

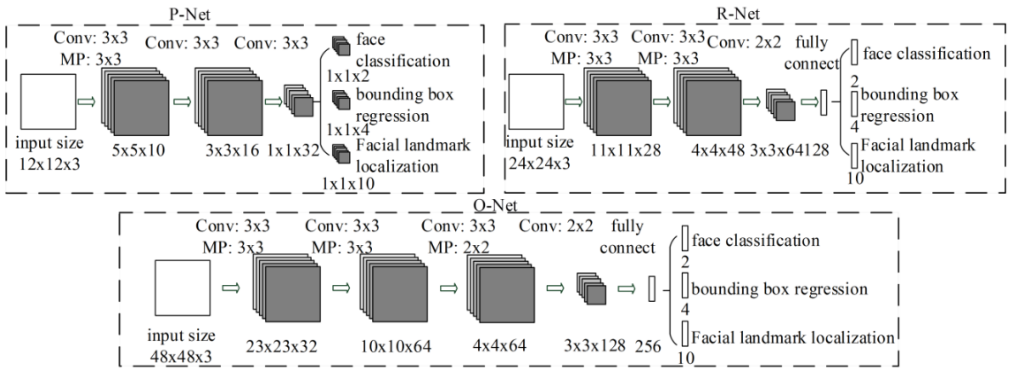


Figure 4.12: The three networks of MTCNN algorithm[10].

P - Net

The input of P-net is a series of different scale images that are created from an original one. This method is called Image Pyramid and it is employed in order to detect images of different sizes in the image. Each image is a copy of the original with a different size. As seen in the figure 4.12 the input of P-net is a 12x12x3 Image. For each scaled image a sliding window of these dimensions will slide from top left to bottom right with a set valued stride and feed them the P-net, which then tries to find a face area. The value of the stride is important, since a high value could result into a big shift and thus to missing a face and a low value could lead to increased computation times with no benefit. Each window is marked with a confidence score and if the confidence score is good enough, the area is bounded by a box and is a candidate face area. Then, NMS is conducted for each scaled image to delete high-overlapping bounding boxes with other bounding boxes of a higher confidence score. More

details about NMS and Bounding Box regression will be given later.

The remaining bounding boxes coordinates and sizes are then converted into the original's image size. This is usually done with the following process. Each copied image has coordinates of (0,0) as top left and (1,1) as bottom right. By multiplying the bounding boxes' coordinates of a copied image with the original image's size, the right coordinates are obtained. It is important to mention that each bounding box needs to be square and of this reason all non square bounding boxes are squared by increasing the elongating the short side of the parallelogram.

P - Net's architecture

P-net is a Fully Convolutional Network - FCN[64], not to be confused with fully connected networks. The main difference with the common CNN is that FCN do not have a dense layer at the end of the network. It contains 3 Convolutional layers which use 3x3 filters and the activation function used after each Convolutional layer is the Parametric ReLU. Max-Pooling is used only after the first Convolutional layer and the output is split into 3 categories. In the first category two neurons take part, one for classifying whether a region is a face or not and the second for providing the confidence score. The second category is the information about the bounding box's top left and bottom right coordinates and its width and height. Finally the third contains the x, y coordinates of the 5 facial landmarks. However P-net and R-net do not really focus on the facial feature localization.

R - Net

The bounding boxes produced by P-net will be the input that is given to R-net. To perform this, the bounding boxes must be resized according to the input needed in the R-net architecture. It is to be noted, that it is possible for a bounding box to exceed the borders of the image. To account for this, the missing part is usually filled with padding. R-net takes the resized bounding boxes and similar to P-net tries to detect face areas with higher precision. To refine the bounding boxes, it uses strict rules and discards the majority of the candidate face areas and bounding boxes with low confidence score. Then it proceeds to convert the remaining bounding boxes coordinates to the original image's coordinates and performs NMS to further decrease the candidate areas. R-net has a simple CNN architecture as seen in the figure 4.12, which has already been explained.

O - Net

O-net is also a simple CNN with increased depth and complexity than the first two, since in essence it is O-net that is responsible for the final output and for the facial landmarks localization. In a similar procedure, the bounding boxes produced by R-net are scaled to the O-net's required input size and are used as input. Then O-net with even higher accuracy and higher facial feature preservation due to the 256 FCN, judges whether there is a face or not. If a face is detected then through regression and NMS, in a similar way to the previous networks, it produces the final bounding box and facial landmarks coordinates.

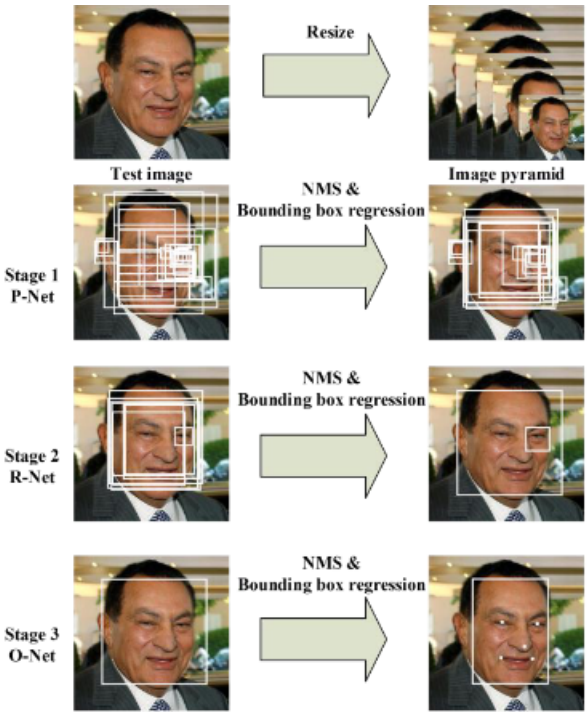


Figure 4.13: An example using MTCNN system [10].

Bounding Box Regression: This method is used in order to calibrate the bounding boxes. In essence the systems makes a prediction on where the bounding box should be instead of where it calculated it and computes a new bounding box based on this prediction. In order to train[65][66] the system to make such prediction a loss function dedicated only to the bounding boxes is inserted. This loss function the authors chose is the euclidean loss function:

$$L_i^{box} = ||y_i^{box} - g_i^{box}||_2^2 \tag{4.2}$$

where y_i^{box} are the top left and bottom right coordinates, the height and the width of the predicted bounding box and g_i^{box} and the nearest ground truth information. Ground truth information are the coordinates of the manually annotated "right" result.

The exact same loss function and procedure is followed for the five facial landmarks' coordinates.

Intersection Over Union - IOU: Intersection Over Union is self explanatory. It is used to calculate how much 2 regions, in this case 2 bounding boxes, overlap.

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ Union} = \frac{S_1 \cap S_2}{S_1 \cup S2}$$

Non-Max Suppression - NMS: Non-Max Suppression is a technique used after each stage in order to discard redundant, highly overlapping bounding boxes[67]. When IoU of 2 bounding boxes is greater than a set threshold value, the bounding box with the lowest confidence score will be discarded resulting in fewer bounding boxes without losing information.

Training and Training Data: As stated before, MTCNN has three different tasks to complete. In order to train for the bounding box and face landmark regression the Euclidean distance loss function is used. For the face classification problem the cross-entropy loss function is used and it is defined as follows:

$$L_i^{det} = -(g_i^{det} \log(p_i) + (1\ -\ g_i^{det})(1\ -\ \log(p_i)))$$

Where p_i is the probability computed by the system of the image being an input and g_i^{det} is the ground truth and $g_i^{det} \in \{0, 1\}$.

Each individual stage is focused in different tasks. For this reason, different types of images are used samples and each task and stage is trained to differently. For example, when a negative image is to be used for training, there is no reason to use it for the bounding box or face landmark regression training. In this case, the $L_i^{landmark}$ and L_i^{box} are set to 0 and only L_i^{det} is computed. As a result, an overall loss function that takes into account all these factors is required. The total loss function is defined:

$$L_{total} = \sum_{i}^{N} \sum_{j \in \{det, box, landmark\}} \alpha_i^j \beta_i^j L_i^j \tag{4.3}$$

where α_i is denoted the importance of the task for each stage, $\beta_i \in \{0, 1\}$ denoting the sample type, L_i is the corresponding Loss function to be used and N is the number of samples. For P-net and R-net: $a_{det} = 1,\ a_{box} = 0.5,\ a_{landmark} = 0.5$, since the main task is to find probable faces and for O-net: $a_{det} = 1,\ a_{box} = 0.5,\ a_{landmark} = 1$. The optimization algorithm used is the Stochastic Gradient Descent.

The training samples used are collected from cropping parts of already labeled images from different Datasets and are separated in four categories. In the first category there are the negative images, where the IoU of the cropped image with the annotated ground-truth is less than 0.3. The following two categories are the positive and part-faces and have $IoU > 0.65$, $0.4 < IoU < 0.65$ respectively. The final category is the landmark faces, which are face images with the five landmarks' positions labeled and are used for facial landmark localization task training. For bounding box regression training the positive and part-faces images are used and for the face classification the positive and negative. The authors also utilized a modified version of a method called Online Hard Sampling Mining. With this method, each sample was firstly forward propagated through the system and had its loss calculated. The 30% of the samples with the lowest loss were then discarded, since it was thought that they had little to offer in the training in comparison with 70% of the remaining hard samples.

P-net is trained with these 70% of the samples first. Then using the trained P-net, predictions are made for these samples. Following the hard samples of the generated samples from P-net are used as training samples for R-net. Similarly O-net uses as training the hard samples from both P-net and R-net. In [68] a slightly different approach of this method is proposed to improve MTCNN's efficiency, however without clear results.

Overall MTCNN is a robust, precise and extremely fast algorithm. Performance analysis compared to other popular algorithms can be found in [10][69][70][71].

Chapter 5

Summary and the future of Face detection and recognition

The current thesis presented and explained a number of face detection and recognition algorithms, that used different approaches to achieve their purpose. To accomplish this, literature corresponding to each algorithm was reviewed, along with code from real applications. The goal was to provide the reader a concise but detailed view of how each algorithm operates and which technologies it utilizes, as well as how the face detection and recognition field advanced through time.

Nonetheless, considering the extensive amount of algorithms, together with the different versions of each algorithm, it is not plausible nor desirable to cover all of them. Certain algorithms, that played a major role in the advancement of the field of face detection and recognition were chosen and expanded upon. Furthermore, certain technologies like Neural Networks that benefited to this advancement were also demonstrated to a basic degree.

A future expansion of this research could include some of the most recent variations of deep learning algorithms, the new experimental technologies that are being employed, as well as the new uses of such algorithms. Since the error rates are decreasing year by year and the algorithms become more and more efficient, newer perspectives on different applications become possible. An increasingly interesting application, that expands on the detection and recognition of faces, is the recognition of the person's emotions based on the structure of the face in a present image [72][73]. This technology would further extend the efficiency of security systems that use facial detection and would be great research material for psychologists among other uses[74]. The most prominent new technology, that is to improve the current

state is the detection using 3D space, instead of 2D images. There are still several challenges to be confronted with 3D face recognition, however it shows that it could prove superior to the 2D image analysis[75].

Face detection and recognition together with their applications, will with no doubt be a major point of interest for many researchers in the future and will have a leading role in the research of Computer Vision related problems. The future of many fields like robotics, psychology, security systems and others is now closely related to the research and development of better and faster detection and recognition algorithms.

Bibliography

[1] pyimagesearch. http:/https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/.

[2] towardsdatascience. https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b.

[3] researchgate. https://www.researchgate.net/figure/SVM-Kernel-Function-Types tbl2 239386696.

[4] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, January 1991.

[5] Victor lavrenko. https://www.youtube.com/watch?v= lY74pXWlS8.

[6] scholarpedia. http://www.scholarpedia.org/article/Fisherfaces.

[7] J A Movshon, I D Thompson, and D J Tolhurst. Spatial summation in the receptive fields of simple cells in the cat's striate cortex. *The Journal of Physiology*, 283(1):53–77, Oct 1978.

[8] Matteo Carandini. What simple and complex cells compute. *The Journal of Physiology*, 577(2):463–466, Nov 2006.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[11] Rajeev Ranjan, Ankan Bansal, Jingxiao Zheng, Hongyu Xu, Joshua Gleason, Boyu Lu, Anirudh Nanduri, Jun-Cheng Chen, Carlos D. Castillo, and Rama Chellappa. A fast and accurate system for face detection, identification, and verification. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 1(2):82–96, 2019.

[12] Galal Hassan and Khalid Elgazzar. The case of face recognition on mobile devices. In *2016 IEEE Wireless Communications and Networking Conference*, pages 1–6, 2016.

[13] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Face-tld: Tracking-learning-detection applied to faces. In *2010 IEEE International Conference on Image Processing*, pages 3789–3792, 2010.

[14] Joshila Grace.L.K and K. Reshmi. Face recognition in surveillance system. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–5, 2015.

[15] Muhammad Mahboob, Muhammad Iqbal, Iftikhar Ahmad, Madini Alassafi, Rayed Al-Ghamdi, Mohammad Basheri, and Muhammad Waqas. Real-time surveillance through face recognition using hog and feedforward neural networks. *IEEE Access*, 7:1–1, 08 2019.

[16] D. Yang, Abeer Alsadoon, P.W.C. Prasad, A.K. Singh, and A. Elchouemi. An emotion recognition model based on facial recognition in virtual learning environment. *Procedia Computer Science*, 125:2–10, 2018. The 6th International Conference on Smart Computing and Communications.

[17] Ying-Li Tian, Takeo Kanade, and Jeffrey F. Cohn. *Facial Expression Analysis*, pages 247–275. Springer New York, New York, NY, 2005.

[18] Shan An, Xin Ma, Rui Song, and Yibin Li. Face detection and recognition with surf for human-robot interaction. In *2009 IEEE International Conference on Automation and Logistics*, pages 1946–1951, 2009.

[19] Viola Paul and Jones Michael. Rapid Object Detection using a Boosted Cascade of Simple Features. *Computer Vision and Pat-tern Recognition, 2001. CVPR 2001. Proceedings of the 2001IEEE Computer Society Conference on*, 1:1–9, 2001.

[20] Franklin C. Crow. Summed-area tables for texture mapping. *In Proceedings of the 11th annual conference on Computer graphics and interactive techniques, SIG- GRAPH '84*, 3:207–212, 1984.

[21] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, pages 23–37, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[22] Huachun Yang and Xu An Wang. Cascade classifier for face detection. *Journal of Algorithms and Computational Technology*, 10:187–197, 2016.

[23] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. *Proceedings, ICIP 2002 International Conference on Image Processing*, 1:900–904, 2002.

[24] Amelia Wong Azman Yasir Mohd Mustafah. Out-of-plane rotated object detection using patch feature based classifier. *Elsevier*, 41:170–174, 2012.

[25] Yong Ma and Xiaoqing Ding. Real-time rotation invariant face detection based on cost-sensitive adaboost. *Proceedings 2003 International Conference on Image Processing*, 3:921–924, 2003.

[26] Zhenchao Xu, Li Song, Jia Wang, and Yi Xu. Improving detector of viola and jones through svm. In Reinhard Koch and Fay Huang, editors, *Computer Vision – ACCV 2010 Workshops*, pages 64–73, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[27] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[28] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[29] Bernhard Fröba and Andreas Ernst. Face detection with the modified census transform. *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pages 91–96, 2004.

[30] Lun Zhang, Rufeng Chu, Shiming Xiang, Shengcai Liao, and Stan Z. Li. Face detection based on multi-block lbp representation. In Seong-Whan Lee and Stan Z. Li, editors, *Advances in Biometrics*, pages 11–18, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[31] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification. In Sameer Singh, Nabeel Murshed, and Walter Kropatsch, editors, *Advances in Pattern Recognition — ICAPR 2001*, pages 399–408, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[32] Timo Ahonen, Abdenour Hadid, and Pietikäinen Matti. Face description with local binary patterns: application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–41, 2006.

[33] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face recognition with local binary patterns. In *Computer Vision - ECCV 2004*, volume 3021, pages 469–481, Berlin, Heidelberg, 05 2004. Springer Berlin Heidelberg.

[34] Laura Sánchez López. ocal binary patterns applied to face detection and recognition, November 2010.

[35] Pedro A. Marín-Reyes, Javier Lorenzo-Navarro, and Modesto Castrillón-Santana. Comparative study of histogram distance measures for re-identification. *arXiv e-prints*, nov 2016.

[36] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):272–297, 1995.

[37] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[38] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[39] A. Hadid, M. Pietikainen, and T. Ahonen. A discriminative feature space for detecting and recognizing faces. In *Proceedings of the 2004 IEEE Computer Society Conference*

on Computer Vision and Pattern Recognition, 2004. CVPR 2004., volume 2, pages 707–804, 2004.

[40] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4(3):519–524, Mar 1987.

[41] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[42] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[43] Sheifali Gupta, O Sahoo, Ajay Goel, and Rupesh Gupta. A new optimized approach to face recognition using eigenfaces. *Global Journal of Computer Science and Technology*, 10:15–17, 2010.

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

[45] Ronald Aylmer Fisher. The use of multiple measures in taxonomic prob-lems,. *Ann. Eugenics*, 7:179–188, 1936.

[46] Xiaofei He, Shuicheng Yan, Yuxiao Hu, P. Niyogi, and Hong-Jiang Zhang. Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

[47] Md Manjurul Ahsan, Yueqing Li, Jing Zhang, Md Tanvir Ahad, and Kishor Datta Gupta. Evaluating the performance of eigenface, fisherface, and local binary pattern histogram-based facial recognition methods under various weather conditions. *Technologies*, 9(2), 2021.

[48] Delpiah Wahyuningsih, Chandra Kirana, Rahmat Sulaiman, Hamidah, and Triwanto. Comparison of the performance of eigenface and fisherface algorithm in the face recognition process. In *2019 7th International Conference on Cyber and IT Service Management (CITSM)*, volume 7, pages 1–5, 2019.

[49] Dake Zhou and Xin Yang. Face recognition using improved-lda. In Aurélio Campilho and Mohamed Kamel, editors, *Image Analysis and Recognition*, pages 692–699, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[50] Keun-Chang Kwak and Witold Pedrycz. Face recognition using a fuzzy fisherface classifier. *Pattern Recognition*, 38(10):1717–1732, 2005.

[51] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591, Oct 1959.

[52] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, apr 1980.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[54] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[55] Feiping Nie, Hu Zhanxuan, and Xuelong Li. An investigation for loss functions widely used in machine learning. *Communications in Information and Systems*, 18:37–52, 01 2018.

[56] Katarzyna Janocha and Wojciech Czarnecki. On loss functions for deep neural networks in classification. *ArXiv*, abs/1702.05659, 2017.

[57] Xue Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168, feb 2019.

[58] A. Pacut. Symmetry of backpropagation and chain rule. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, volume 1, pages 530–534, 2002.

[59] Sebastian Ruder. An overview of gradient descent optimization algorithms. *ArXiv*, abs/1609.04747, 2016.

[60] Jonas Teuwen and Nikita Moriakov. *Convolutional neural networks*, page 481–501. Elsevier, 2020.

[61] Chigozie Nwankpa, W. Ijomah, A. Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *ArXiv*, abs/1811.03378, 2018.

[62] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *arXiv e-prints*, sep 2020.

[63] Jost Tobias Springenberg, A. Dosovitskiy, T. Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2015.

[64] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society.

[65] Ross B. Girshick, J. Donahue, Trevor Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[66] Naomi Dickerson. Refining bounding-box regression for object localization, 2017.

[67] A. Neubeck and L. Gool. Efficient non-maximum suppression. *18th International Conference on Pattern Recognition (ICPR'06)*, 3:850–855, 2006.

[68] Long-Hua Ma, Hang-Yu Fan, Zhe-Ming Lu, and Dong Tian. Acceleration of multi-task cascaded convolutional networks. *IET Image Processing*, 14(11):2435–2441, 2020.

[69] Deisy Chaves, Eduardo Fidalgo, Enrique Alegre, Rocío Alaiz-Rodríguez, Francisco Jáñez-Martino, and George Azzopardi. Assessment and estimation of face detection performance based on deep learning for forensic applications. *Sensors*, 20(16):4491, Aug 2020.

[70] YingGang Xie, Hui Wang, and ShaoHua Guo. Research on mtcnn face recognition system in low computing power scenarios. *Journal of Internet Technology*, 21(5):1463–1475, 2020.

[71] Jiankang Deng, J. Guo, Y. Zhou, Jinke Yu, I. Kotsia, and S. Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. *ArXiv*, abs/1905.00641, 2019.

[72] Ninad Mehendale. Facial emotion recognition using convolutional neural networks (ferc). *SN Applied Sciences*, 2(3), Feb 2020.

[73] Aneta Kartali, Miloš Roglić, Marko Barjaktarović, Milica Đurić Jovičić, and Milica M. Janković. Real-time algorithms for facial emotion recognition: A comparison of different approaches. In *2018 14th Symposium on Neural Networks and Applications (NEUREL)*, pages 1–4, 2018.

[74] Agata Kołakowska, Agnieszka Landowska, Mariusz Szwoch, Wioleta Szwoch, and Michał Wróbel. Emotion recognition and its applications. *Advances in Intelligent Systems and Computing*, 300:51–62, 07 2014.

[75] Muhammad Sajid Khan, Muhammad Jehanzeb, Muhammad Imran Babar, Shah Faisal, Zabeeh Ullah, and Siti Zulaikha Binti Mohamad Amin. Face recognition analysis using 3d model. In Mahdi H. Miraz, Peter Excell, Andrew Ware, Safeeullah Soomro, and Maaruf Ali, editors, *Emerging Technologies in Computing*, pages 220–236. Springer International Publishing, 2018.