Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Magistrale in Bioingegneria

Develpoment of new techniques for the automatic assessment of retinal image quality

Candidato:
Silvia Gazzina
Matricola 1156869

Relatore:
Ch.ma Prof.ssa MariaPia Saccomani

Correlatore:
Dott. Enea Poletti

Anno Accademico 2018-2019

Contents

Abstract

Retinal images acquired by means of digital photography are widely used for analysis and evaluation of ocular fundus, since they provide in a non invasive way key diagnostic information for the early detection of retinal pathologies, such as diabetic retinopathy, glaucoma, macular degeneration and vascular abnormalities [1]. Nonetheless, the clinical usefulness of an image is highly dependent on its quality: our goal is to develop an automated system able to perform a reliable and fast measurement of image quality. In this way, a real time feedback regarding the quality of the acquired image can be given to the operator and corrective actions can be taken immediately, avoiding both waste of resources and inconveniences for the patients [2].

In particular, this work contains the development of three different approaches that attempt to solve the problem of retinal image quality assessment in different ways: the objective is to analyze their own peculiarities and drawbacks in order to understand which one is more suitable for the embedding in an highly automated screening system.

All the methods have been trained and tested on CenterVue proprietary data sets manually labeled by an expert human observer, which contain both images acquired with Eidon and with a prototype of fundus camera still in a development stage.

The results obtained strongly recommend the adoption of the SVM based method, which presents an acceptable computational complexity and shows the best performances in comparison with the other two methods developed.

Chapter 1

Introduction

Retina is a multi-layered sensory tissue that lies on the back of the eye. It contains millions of photoreceptors that capture light rays and convert them into electrical signals. These signals travel along the optic nerve towards the brain, where they are converted into images [3].

There are two types of photoreceptors in the retina: rod cells and cones. The former are generally located at the periphery of the retina, and used for scotopic vision. They are sensitive to changes in contrast even at low light levels, so they are able to detect movement, but imprecise and insensitive to color. On the other hand, cones are high precision cells capable of detecting colors, and they are mainly concentrated in the *macula*, the area responsible for photopic vision. The very central portion of the macula is called *fovea*, which is where human eye is able to distinguish visual details at its best. All the photoreceptors are connected to the brain through a dense network of roughly 1.2 millions of nerves, that bundle together to leave the eye through the optic nerve head, called *optic disc* [4]. In figure 1.1a) the optic disc is clearly visible, together with the fovea and retinal blood vessels, that supply oxygen and nutrients to retina's inner and outer layers. There

(a) Healty retina (b) Pathological retina

Figure 1.1: Comparison of fundus images of an healthy and unhealthy patient

are several pathologies that can affect the retina, an example of which is introduced in figure 1.1b). Eye diseases such as diabetic retinopathy, age-related macular degeneration, glaucoma and vascular abnormalities are very serious conditions that can be easily detected by visual inspection of eye fundus: digital fundus photography plays a key role in this scenario, because it provides crucial diagnostic information in a rapid and non-invasive way [1]. Additionally, over the past few years health care systems have been strongly oriented towards the development of telemedicine systems, in a view of large scale screening programs. Thus, while the diagnosis is always performed by an expert ophthalmologist, personnel with varying level of experience usually carries out images acquisition, subjectively evaluating if they are of sufficient quality [5]. Hence, it is rather common that low-quality images are mistakenly accepted, leading to the necessity of a reacquisition, which is time consuming and expensive [6]. Moreover, image reacquisition implies delayed diagnosis and treatment, which is potentially dangerous for the patient: obtaining the highest possible image quality is thus fundamental in order to enable the clinician to make an accurate and reliable diagnosis and avoid all the kinds of issues above mentioned.

(a) Poor illumination (b) Poor focus and field definition

(c) Lash artifact (d) Total eye blink

Figure 1.2: Examples of impaired retinal images

Besides, there are many factors that may impair retinal image quality, degrading it to the point of rendering it *ungradable*. According to a study performed by the University of Wisconsin-Madison [7], the parameters focus and clarity, field definition, visibility of the macula, visibility of the optic disc, and artifacts must be taken into account for the correct evaluation of retinal image quality. The study found that image artifacts are mostly caused by the occurrence of haze, presence of dust and dirt, partial occlusion by eyelashes, improper cleaning of camera lens, uneven illumination over the macula, uneven illumination of the optic disc and of image edges, and total eye blink.

Moreover, image quality is an highly *subjective* concept, and its human evaluation can vary broadly even among experts, especially when dealing with images of borderline quality.

Thus, retinal image quality assessment is a very challenging task: consequently, we decided to cope with it by developing three different methodologies, all with their strengths and drawbacks, in order to compare them and understand which one could solve the problem at best.

In the following chapter a brief description of the data sets employed in order to train and test all the methodologies developed is reported, together with some details regarding acquisition instruments and their characteristics. Subsequently, in chapter 3 the descriptors adopted and their extraction process from images is explained. The remaining part of the chapter is dedicated to method's exposure, whose results are then presented in chapter 4. Ultimately, in chapter 5 a discussion of such results is reported, followed by a brief description of the future perspectives.

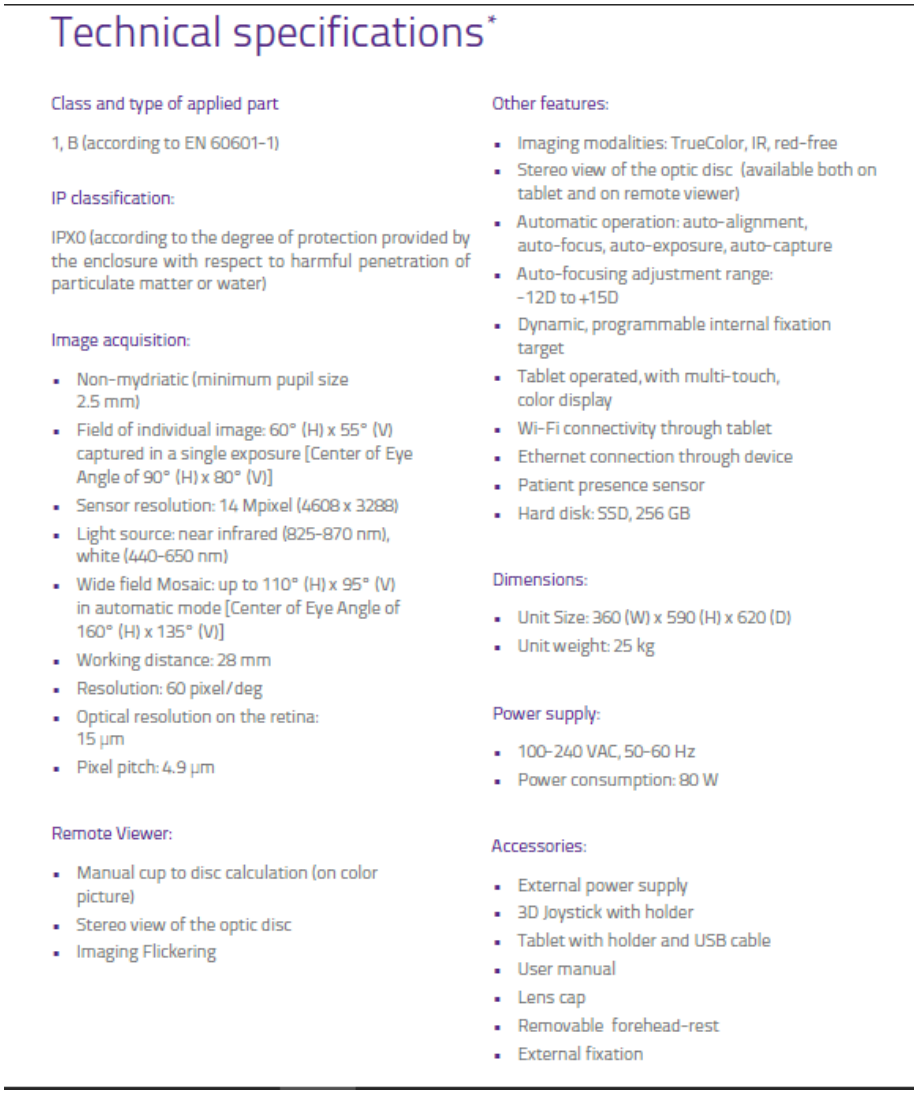
Chapter 2

Materials

Instruments and data sets

All retinal images employed in this thesis work have been acquired by means of two different instruments designed and manufactured by CenterVue: the true color confocal scanner Eidon and a new prototype of fundus camera, which is still in a development stage. For this reason, in the following we will give only a few information about its technical specifications so as to make possible a comparison with Eidon, whose details are instead provided in figure 2.1. The main technical differences among those two instruments are regarding their confocality (because Eidon is more confocal than the other fundus camera prototype), field of view (60° FOV Eidon and 45° FOV prototype) and resolution (14 MPixel Eidon instead of 8 MPixel prototype). Moreover, their clinical target is quite different: Eidon is a *diagnostic* system, mainly used in specialized clinics by eye care professionals for the detection of retinal pathologies, while this new instrument has been mostly conceived for *screening* purposes. Consequently, it will be an highly automated system so as it could be employed not only by highly trained clinicians, but also by personnel with varying level of experience in less specialized environments.

As regards the data sets used in this work, they represent a very heterogeneous sample of retinal images population, since they are composed by both healthy and pathological images of children, adults and elderly people. Before their usage, all these images have been anonymized and manually labeled as of good or bad quality by an expert human observer, as table 2.1 shows. This manual grading represents a fundamental pre processing step on our data sets, since it is then used in the following by all the methods as *ground truth*.



Technical specifications*

Class and type of applied part

1, B (according to EN 60601-1)

IP classification:

IPX0 (according to the degree of protection provided by the enclosure with respect to harmful penetration of particulate matter or water)

Image acquisition:

- Non-mydriatic (minimum pupil size 2.5 mm)
- Field of individual image: 60° (H) x 55° (V) captured in a single exposure [Center of Eye Angle of 90° (H) x 80° (V)]
- Sensor resolution: 14 Mpixel (4608 x 3288)
- Light source: near infrared (825-870 nm), white (440-650 nm)
- Wide field Mosaic: up to 110° (H) x 95° (V) in automatic mode [Center of Eye Angle of 160° (H) x 135° (V)]
- Working distance: 28 mm
- Resolution: 60 pixel/deg
- Optical resolution on the retina: 15 µm
- Pixel pitch: 4.9 µm

Remote Viewer:

- Manual cup to disc calculation (on color picture)
- Stereo view of the optic disc
- Imaging Flickering

Other features:

- Imaging modalities: TrueColor, IR, red-free
- Stereo view of the optic disc (available both on tablet and on remote viewer)
- Automatic operation: auto-alignment, auto-focus, auto-exposure, auto-capture
- Auto-focusing adjustment range: -12D to +15D
- Dynamic, programmable internal fixation target
- Tablet operated, with multi-touch, color display
- Wi-Fi connectivity through tablet
- Ethernet connection through device
- Patient presence sensor
- Hard disk: SSD, 256 GB

Dimensions:

- Unit Size: 360 (W) x 590 (H) x 620 (D)
- Unit weight: 25 kg

Power supply:

- 100-240 VAC, 50-60 Hz
- Power consumption: 80 W

Accessories:

- External power supply
- 3D Joystick with holder
- Tablet with holder and USB cable
- User manual
- Lens cap
- Removable forehead-rest
- External fixation

Figure 2.1: Eidon datasheet

Instrument	Total	Good	Bad	Size
Eidon	2242	1624	618	3680x3288
Prototype	322	177	145	3046x2630

Table 2.1: Data sets composition

Moreover, it is important mentioning that the numbers of table 2.1 are relative to the *entire* data sets, containing images acquired with various fields. As regards the Eidon data set, we did not make any kind of distinction, employing it all for the development of the deep learning-based method reported in section 3.5, which requires a huge number of data to be properly

trained from scratch. On the contrary, we decided to cream off the data set hosting images acquired with the prototype, selecting and considering only the ones with central field (macula-centered). Consequently, we reduced this data set to 181 images, respectively 95 of good quality and 86 of bad quality. This choice is driven by the fact that we do not have a sufficient number of images acquired with other fields to be used for the development of the first two methods (cfr. sections 3.3 and 3.4), which rely on the extraction of handcrafted features from various images' ROI (cfr. 3.2). Thus, with the adoption of such a reduced data set we are able in the end to obtain statistical significant results for the comparison of all the methodologies developed.

Finally, all the processing made have been carried out with Matlab R2018b, running on an Inter Core i7- 2600 @ 3.40 GHz with 16 GB RAM and an enabled GPU Nvidia GeForce GTX 1060-6GB.

Chapter 3

Methods

3.1 Features Presentation

The first two approaches developed for the automatic assessment of retinal image quality rely on the identification of global and local features, which are well correlated with human visual perception [2] and thus completely interpretable. In particular, we decided to extract from images' regions of interest *generic* image quality descriptors, which are introduced in table 3.1. These parameters make use of simple statistical measures based on image histogram to capture its overall quality content, avoiding eye structure segmentation procedures, which are complex and error-prone especially in case of images with poor quality [8].

Brightness
1^{st} Quartile
2^{nd} Quartile
3^{rd} Quartile
Contrast
Kurtosis
Skewness
Focus
Entropy
3 Haralick features

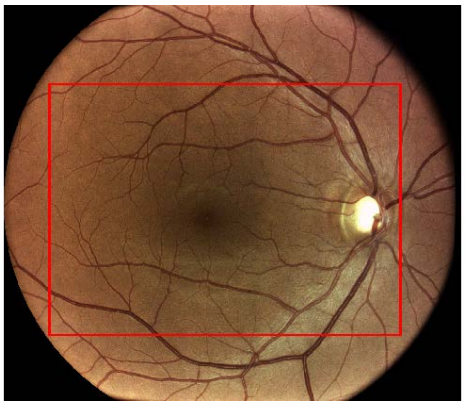
Table 3.1: Descriptors used to determine image quality

As a matter of fact, although literature provides successful examples of *combined* approaches [9], i.e. solutions that make use of both generic image quality indicators and structure related parameters, in this pre processing phase we wanted to keep the computational complexity low, in the view of developing agile and fast methodologies that can be used in real time applications.

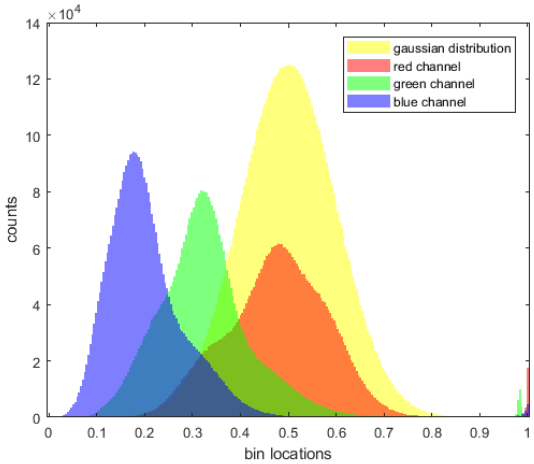
Each descriptor is computed for the three color channels in different regions of interest of the image, described in section 3.2. The only exception is given by the focus descriptor, because it is computed at a global level rather than on local regions of interest.

In order to better understand the role of each descriptor in highlighting a specific image characteristic and to prove its sensibility to changes in such characteristic, we applied several image processing operations intended to simulate different scenarios of impaired image quality.

First of all, we selected the rectangular ROI shown in figure 3.1a) on the original image to leave out its black edges, that are not of our interest for quality assessment purposes. In this way we remove unimportant information and include only pixels showing retinal data, simulating the regions of interest selection process performed when the actual algorithms are developed. At the same time, we also considered a phantom image with a



(a) Rectangular ROI



(b) Gaussian distribution and original RGB histogram

Figure 3.1

Gaussian pixel intensity distribution: as a matter of fact, as we can notice from figure 3.1b), the pixel intensity distributions in the three color planes of the original image are a bit asymmetric compared to it, so we want to have a theoretical reference to which we can refer to.

Brightness

Brightness is a very simple and powerful descriptor used to capture image quality, since it is related to the general level of illumination in an image. It is simply computed by taking the mean value of pixels inside a ROI, and it relies on the fact that evenly illuminated and bright images present a higher average.
In order to show it, we added and subtracted the following constant values $c = [0.1, 0.2, 0.3, 0.4, 0.5]$ to the phantom image and to the three color channels of the original image. In terms of pixels intensity distribution, the consequence is a shift of image histogram to the left or to the right without changes of shape, but with a little saturation effect in 0 and 1, which are our dynamic range's limits. Moreover, we can notice from figure 3.3 how

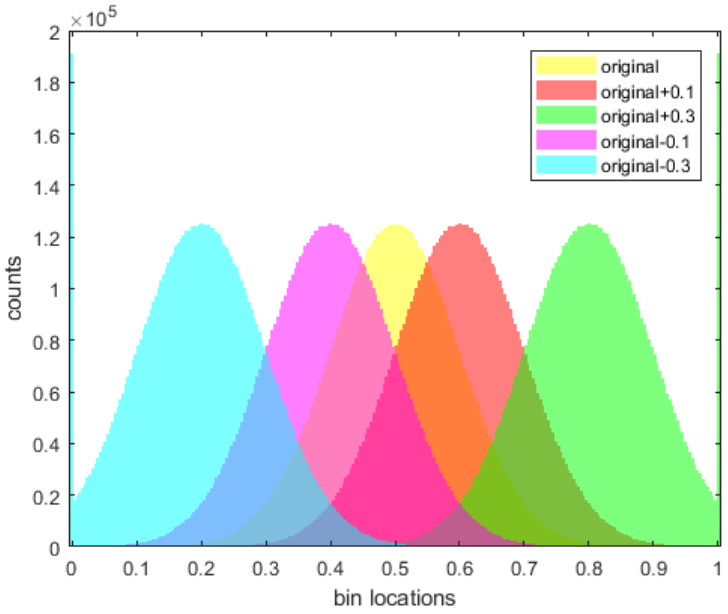


Figure 3.2: Histogram shift

the brightness descriptor respectively decreases and increases compared to its original value as the histogram is centered into darker and lighter values: underexposure and overexposure are both factors that contribute in reducing image quality, and are well captured by this descriptor.

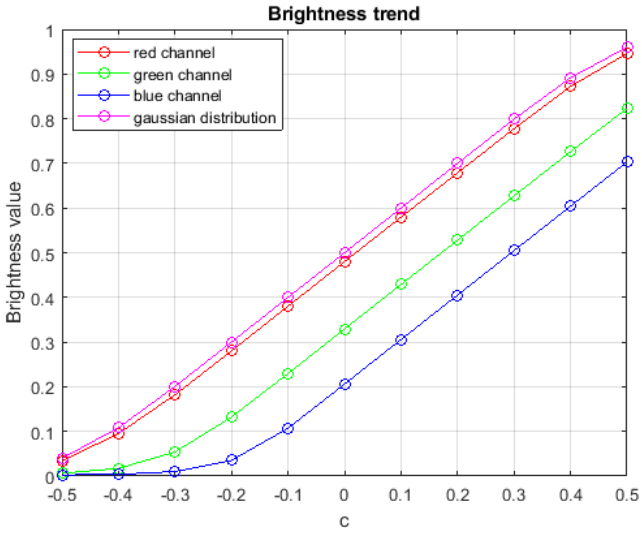


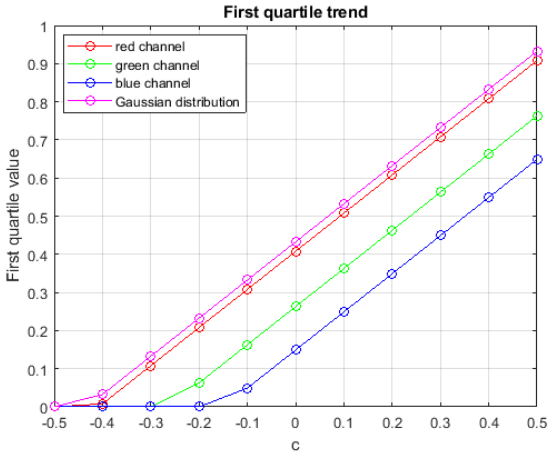
Figure 3.3: Brightness trend

Quartiles

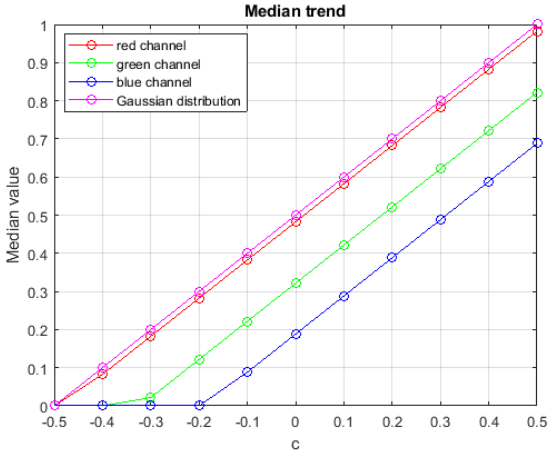
In descriptive statistics, quartiles are values that divide a set of data into four parts of equal size. Particularly, in this work we have considered:

- First Quartile: defined as the middle number between the smallest value and the median of image's pixel intensity distribution.
- Second Quartile: also called median, it divides image's pixel intensity distribution into two parts of equal size.
- Third Quartile: defined as the middle value between the median and the highest value of image's pixel intensity distribution.

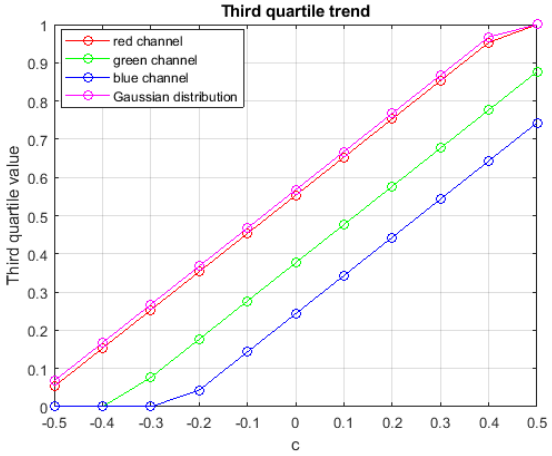
In terms of image quality, quartiles bring an information similar to the one carried by the brightness descriptor: as a matter of fact, adding and subtracting the same constant values used to validate the brightness descriptor to the three color channels of the original image and to the phantom Gaussian image, we obtain the linear trends shown in figure 3.4.



(a) First quartile trend



(b) Median trend

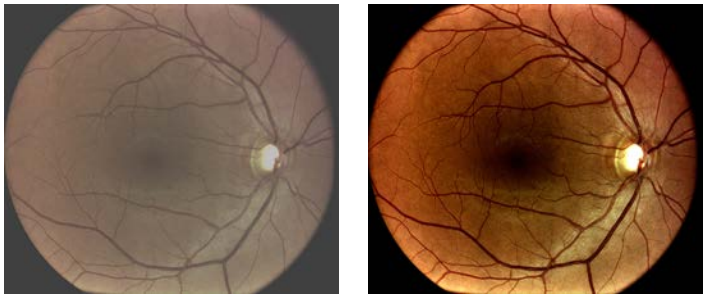


(c) Third quartile trend

Figure 3.4: Quartiles trend

Contrast

The contrast measure we adopt in this work is both visual effective and simple, because it can be expressed by computing the standard deviation of pixel intensities for each ROI and color channel of the image: low contrast will reflect in low standard deviation, regardless of overall image brightness. Conversely, sharp images will present a high contrast value. This concept is exemplified in figure 3.5, where we can see how different contrast levels affect image quality.



(a) Low contrast image (b) High contrast image

Figure 3.5: Examples of images with different contrast

In order to demonstrate that this descriptor is actually sensitive to changes in image contrast, we applied the intensity transformation [10] shown in figure 3.6 to the three color channels of the original image and to the Gaussian phantom image, for various values of c.

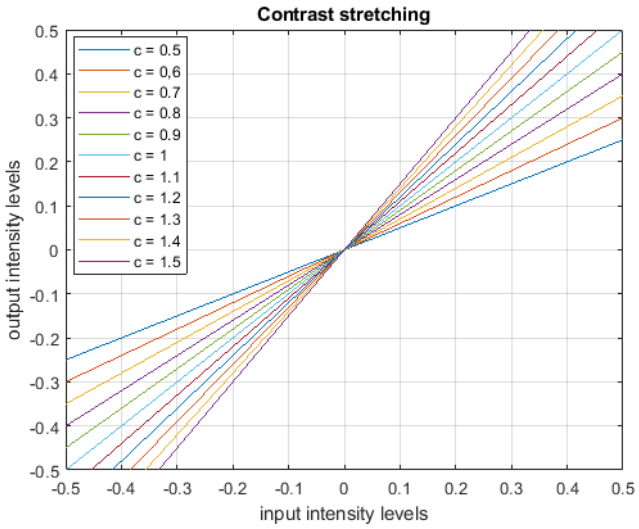


Figure 3.6: Contrast stretching transformation

The result of this operation in terms of image histogram is shown in figure 3.7.

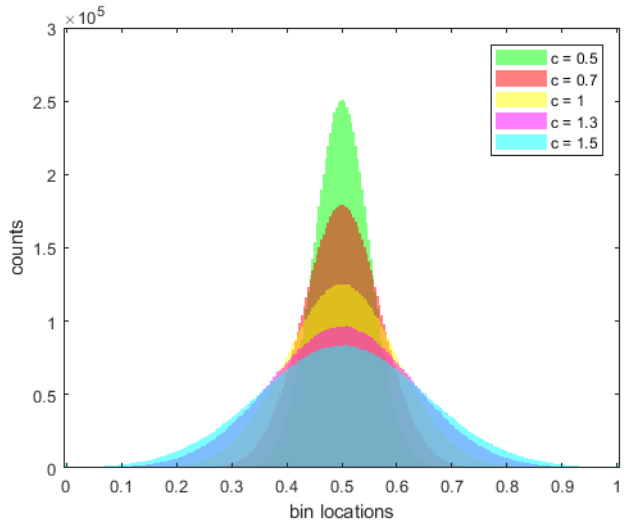


Figure 3.7: Changes in Gaussian image histogram for various values of c

In particular, we can notice that:

- for $c < 1$ the resulting image is smoother. The entire range of input intensity values is mapped into a narrower range in output, so image contrast is decreased.
- for $c > 1$ the resulting image is sharper. As a matter of fact, we map image input intensity values range into all the available output values, with a little saturation effect that becomes much more present as c becomes greater than 1.

Then, we computed the contrast values of all the images derived by applying the transformation above introduced with various values of c, and compared them with the contrast value of the original image. The results obtained clearly show the ability of this descriptor in detecting image contrast changes, and are presented in figure 3.8.

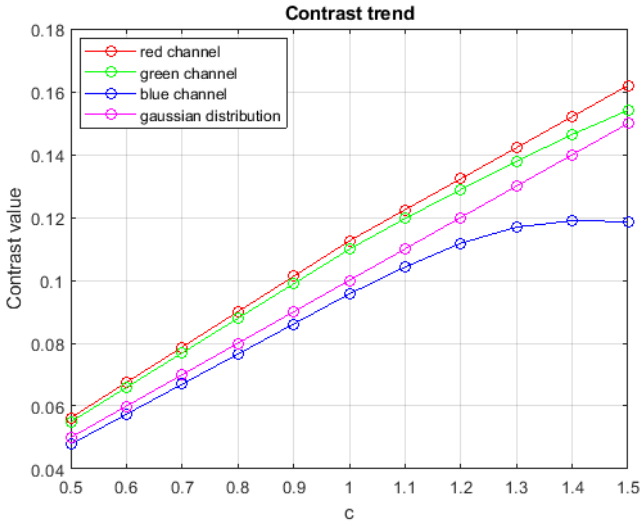


Figure 3.8: Contrast trend

Kurtosis

This descriptor is defined as:

$$k = \frac{E(x-\mu)^4}{\sigma^4} \tag{3.1}$$

and is computed for each color channel of the original image and for the Gaussian phantom image.

It is a measure of whether the histogram of pixel intensities is peaked or flat with respect to a normal distribution, which has a kurtosis of 3. As a matter of fact, as we can see from figure 3.9, data sets with high kurtosis (greater than 3) tend to to decline rather rapidly, with long and fat tails, and they often have a distinct peak near the mean. These types of distributions are said to be *leptokurtic*. Vice versa, data sets with kurtosis less than 3 tend to have a flat top near the mean and shorter, thinner tails, so they are said to be *platikurtic*. As a result, the former are more outlier-prone, while the latter produce fewer and less outliers than the normal distribution does. In terms of image quality, we noticed that this descriptor is sensitive to both image resolution and lightness homogeneity, but it is hard to provide a precise visual interpretation.

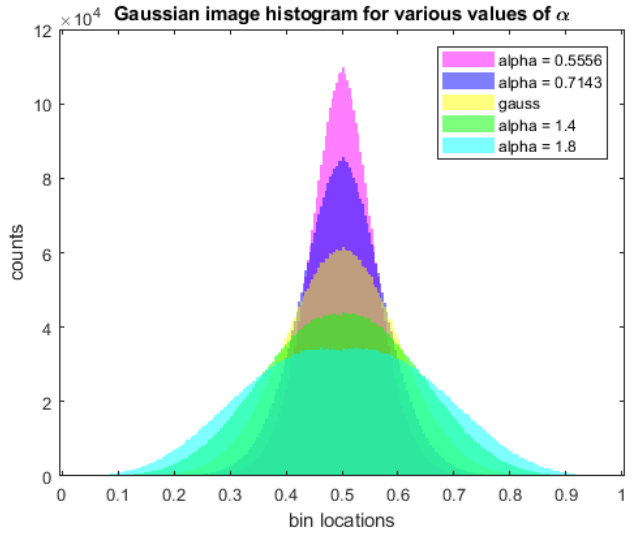


Figure 3.9: Histogram of the Gaussian phantom image with mean = 0.5, variance = 0.01 and different excess kurtosis

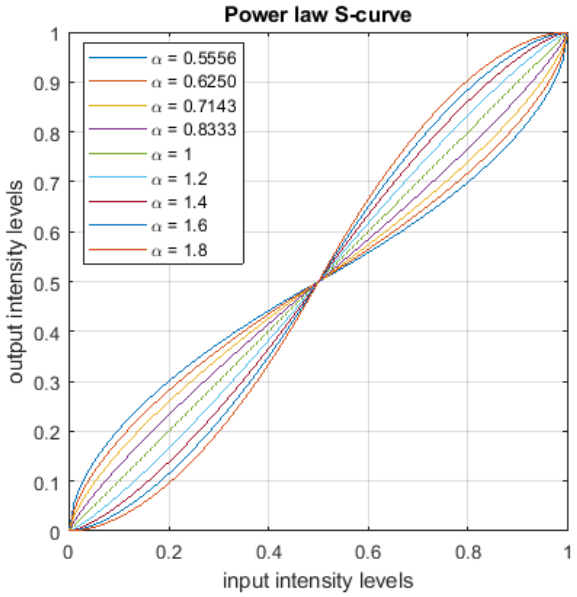


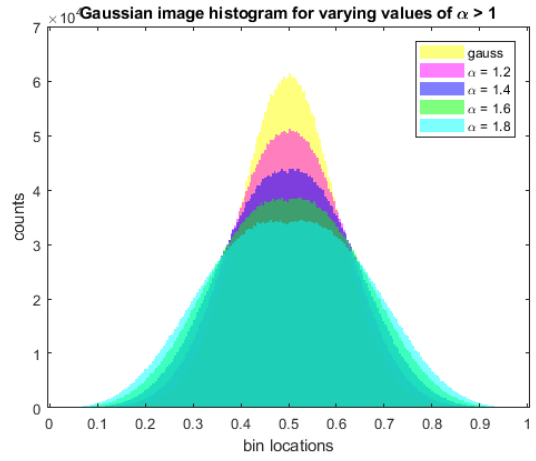
Figure 3.10: Sigmoid transformation

The transformation showed in figure 3.10 has been exploited to modify both the Gaussian pixel intensity distribution and the three color channels' histogram, in order to point out different excess kurtosis cases:

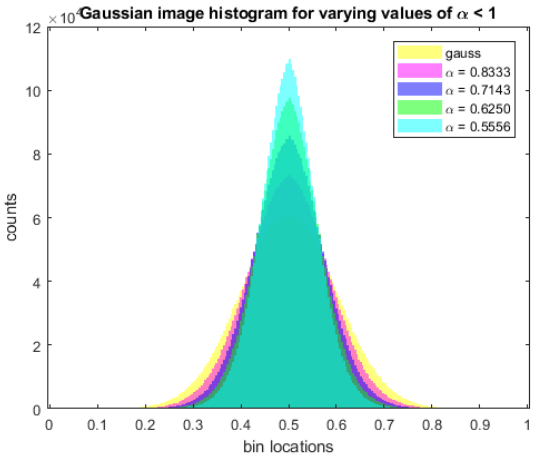
- $\alpha > 1$: kurtosis decreases, since the central part of input values is mapped into a wider portion of central output values. Furthermore,

darker (lighter) input values become even more darker (lighter) in output.

- $\alpha = 1$: identity transformation. Kurtosis remains the same.
- $\alpha < 1$: kurtosis increases, as a wide part of central input values is mapped into a narrower output range. Consequently, as pointed out in figure 3.11b), we will have a more peaked distribution with heavy tails.



(a) Decreasing kurtosis



(b) Increasing kurtosis

Figure 3.11: Different excess kurtosis cases

The resulting kurtosis trend is reported in figure 3.12.

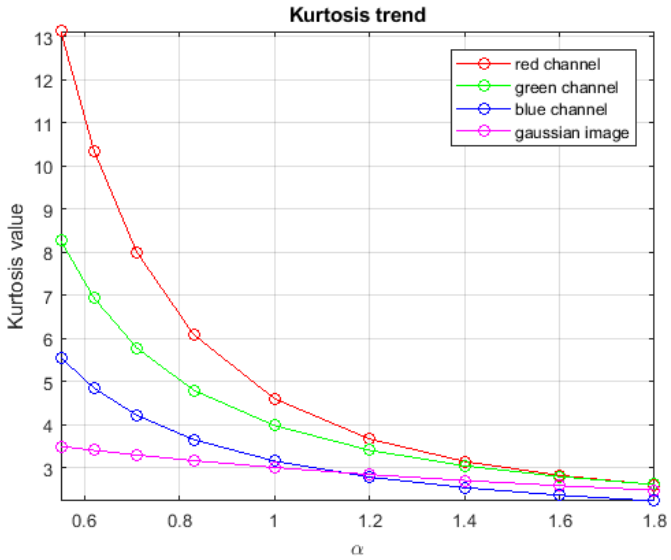


Figure 3.12: Kurtosis trend

Skewness

The skewness descriptor is defined as

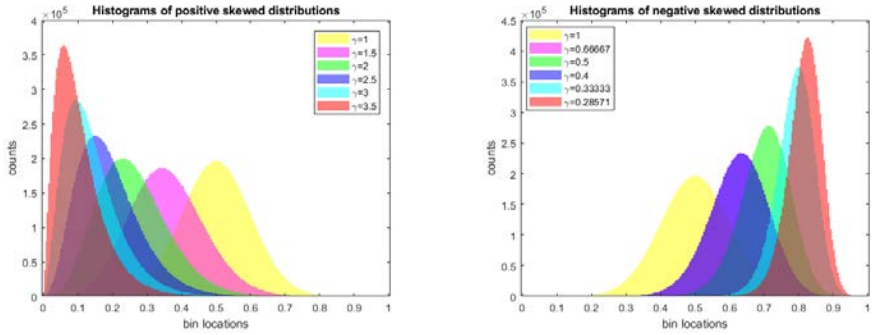
$$k = \frac{E(x-\mu)^3}{\sigma^3} \tag{3.2}$$

and is computed for each color channel of the original image and for the Gaussian phantom image.

It is a measure of symmetry, or more precisely, the lack thereof. We can say that a distribution is symmetric if the two tails on both sides of the mean are the mirror of each other: this is the case of a normal distribution, which has a skewness of 0. Whereas, left-skewed distributions (also called *negatively skewed* distributions) have a long left tail with respect to their right tail, and are characterized by negative values of skewness. Lastly, right-skewed distributions (otherwise known as *positive skewed* distributions) have a long right tail compared to their left tail, and are characterized by positive values of skewness. We can appreciate it in figure 3.13. In terms of visual interpretation, this descriptor is mainly associated with illumination homogeneity in the image, which is another important factor to be taken into account when assessing image quality.

In order to enlighten how this descriptor works, the gamma transformation introduced in figure 3.14 has been applied to the Gaussian phantom image and to the three color planes of the retinal image, with different values of γ.

As a matter of fact:



(a) Positive skewed distributions (b) Negative skewed distributions

Figure 3.13: Examples of skewed distributions

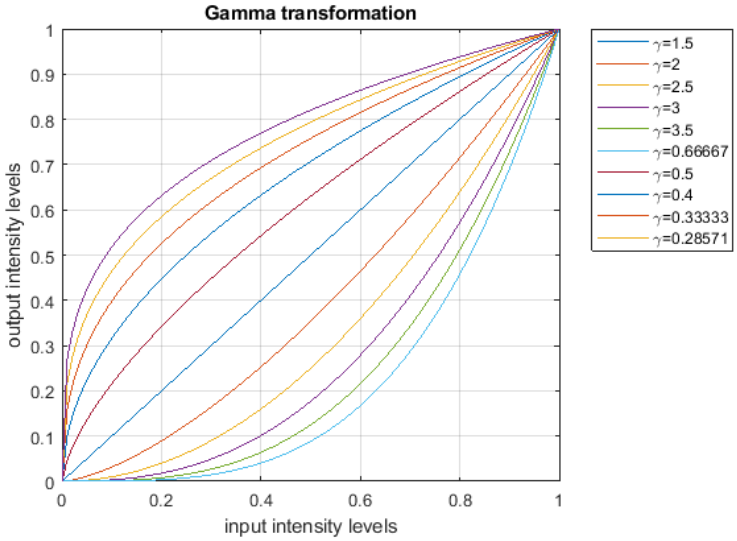


Figure 3.14: Gamma transformation

- for $\gamma > 1$: the mapping of input values is weighted towards lower output values. Consequently, skewness increases.
- for $\gamma = 1$: each value of the input image is mapped into the same intensity level in output. Skewness remains the same.
- for $\gamma < 1$: the mapping of input values is weighted towards higher output values. Thus, skewness is decreased.

In the end, from figure 3.15 we can see how skewness changes as γ is modified. In particular, it is important to notice the different starting points of the curves, relative to the initial skewness values of the distributions to which they relate. The curve associated with skewness changes of the Gaussian pixel distribution starts from 0, while the others are relative respectively to

the red, green and blue channel of the retinal image, which are positively skewed since the beginning. For this reason they have a positive starting point, even if it is linked to the situation where the identity transformation is applied.

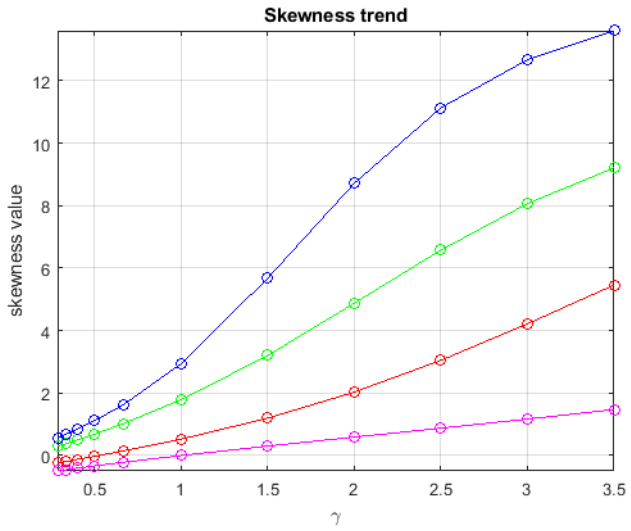
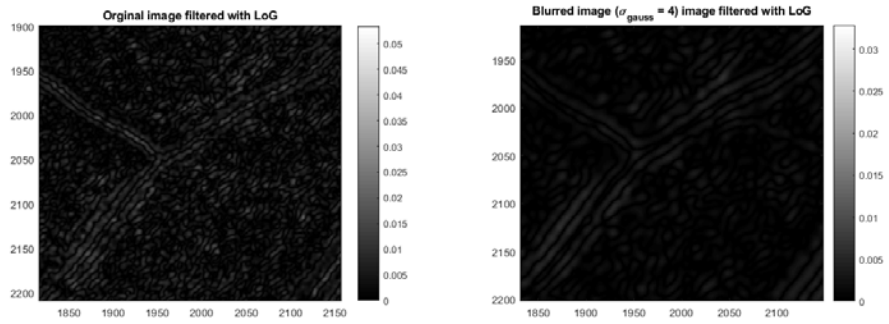


Figure 3.15: Skewness trend

Focus

Focus is an essential descriptor to detect image blur, that even on its own can significantly affect the quality of a retinal image. First of all, in order to develop and test our focus measure, we artificially degraded a retinal image blurring it with a Gaussian filter of increasing standard deviation (σ_{gauss}). Then, we built up a Laplacian of Gaussian (LoG) filter trying out various values of the parameter σ_{LoG}, so as to pick up the one with best performances: namely, this filter must be able to highlight the vessel component from the background together with the tiny details present in the image, being at the same time insensitive to noise.

Subsequently, we convolved the original image and its blurred versions with all the LoG filters candidates. The choice of $\sigma_{LoG} = 2.3$ turned out being the best one for our problem, as exemplified from figure 3.16, because, before the application of the Laplacian operator, it introduces an optimal smoothing level in our image: as a matter of fact, it is sufficient to erase noise while preserving image useful information. Moreover we can notice that when the image is focused, the filter emphasizes all image characteristics of our interest above listed, while, as soon as the image is out of focus, its filtered version is smoothed and missing of sharp edges and details. It's worth mentioning that the LoG filter highlights regions of rapid intensity



(a) Detail of focused image filtered with LoG ($\sigma_{LoG} = 2.3$)

(b) Detail of out of focus image filtered with LoG ($\sigma_{LoG} = 2.3$)

Figure 3.16: Examples of LoG filtered images

changes, giving both positive and negative responses depending on whether a dark edge is detected over a light region or vice versa. Since we are actually interested in the absolute value of the variation and not in its sign, we take the absolute value of pixels of the filtered image before computing the focus descriptor, which is finally defined as the mean value of pixels of the LoG filtered image. In figure 3.17 we can appreciate the power of this descriptor in discriminating images with different focus levels: first of all, we can notice how it is capable of revealing even little amounts of blur, because there is a difference between the focus value associated with the original image ($\sigma_{gauss} = 0$) and the image blurred with $\sigma_{gauss} = 1$, even if visually the two images look almost the same. Besides, the focus value computed for the original image is approximately 10 times greater than the one computed for the image blurred with $\sigma_{gauss} = 10$.

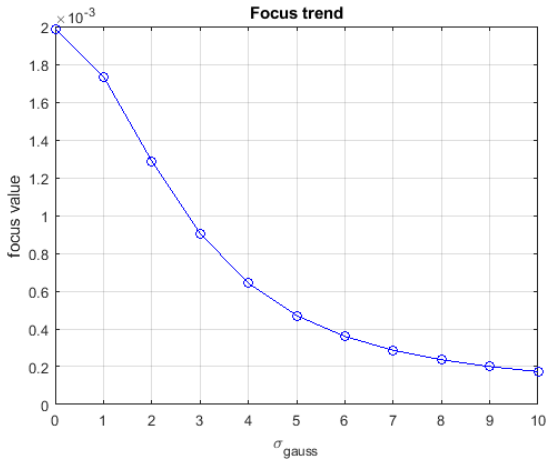


Figure 3.17: Focus trend

Entropy

Entropy is a statistical measure of randomness, that can be used to measure image resolution and contrast. It is defined as:

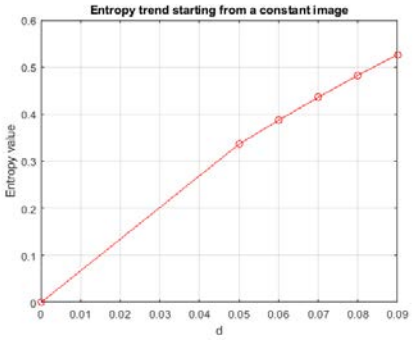
$$H = -\sum p \log_2(p) \tag{3.3}$$

where p represents the probability of occurrence of each intensity level in the image.

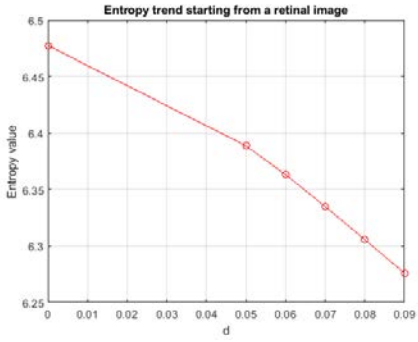
In order to understand how this descriptor works, we decided to start from situations in which we know what to expect in terms of entropy (cases of constant and uniform image) and we added two different types of noise: salt and pepper and Gaussian.

As a matter of fact, from equation 3.3 follows that $H = 0$ for a constant image, which is characterized from just one level of intensity occurring with $p = 1$ allover the image. Whereas, H assumes its maximum value when image has a pixels' uniform distribution, i.e. when each intensity level occurs approximately with the same probability.

In figure 3.18a) we can see how the entropy value of a constant image varies when adding Salt and Pepper noise with increasing density: progressively, from the only intensity level present in the image with $p = 1$, other two intensity levels (0 and 1) are introduced with increasing probability. Consequently, entropy grows from the initial value of 0. However, repeating the same process but starting from a retinal image, we obtain the result shown in figure 3.18b). Of course the starting point is far away from 0, since a retinal image is characterized approximately by a Gaussian pixel intensity distribution (especially the red channel), and so by a positive entropy level.



(a) Result of adding salt and pepper noise with increasing density d to a constant image



(b) Result of adding salt and pepper noise with increasing density d to a retinal image

Figure 3.18: Examples of images with added salt and pepper noise of increasing density

The behaviour noticed from figure 3.18b) is qualitatively similar to the case of salt and pepper noise added to an image with an uniform distribution of pixels, as we can see from figure 3.19. That's because, as the salt and pepper noise density is increased, the probability of occurrence of black and white pixels increases as well with respect to the occurrence probability of all the other intensity levels present in the image. As a result, the level of randomness present in the image is reduced, as well as the entropy level.

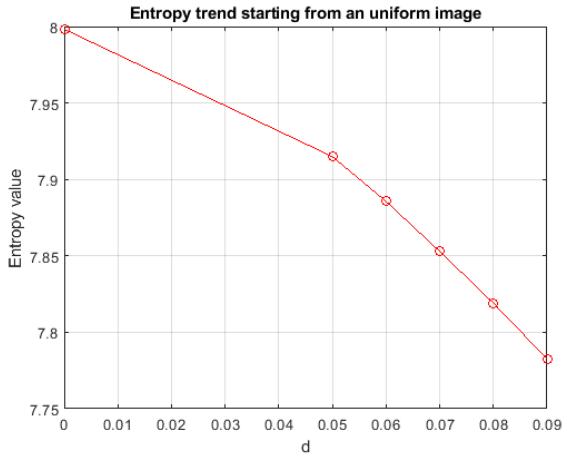


Figure 3.19: Result of adding salt and pepper noise with increasing density d to an uniform image

In the end, we added Gaussian noise with zero mean and increasing standard deviation both to the constant and to the retinal image. In the first case, the result obtained in terms of distributions is shown in figure 3.20: we can notice how an increasing number of intensity levels is assumed by image pixels with increasing probability. For this reason, as confirmed by figure 3.21a), the entropy level increases with respect to its initial value of 0. The same behaviour can be traced in case of Gaussian noise added to the retinal image, as we can see from figure 3.21b). The only difference from the previous case is the starting entropy level, that of course in this case is greater than 0.

Haralick features

Haralick features belong to the so called *second order* statistics, that investigate the relationship existing between pairs of pixels in an image. This is opposed to the concept of *first order* statistics, that are instead measures linked to the observation of a single intensity level in the image, regardless of where it is placed. For this reason, Haralick features are related to the texture information present in an image, that can be derived from the Grey Level Co-occurrence Matrix (GLCM).

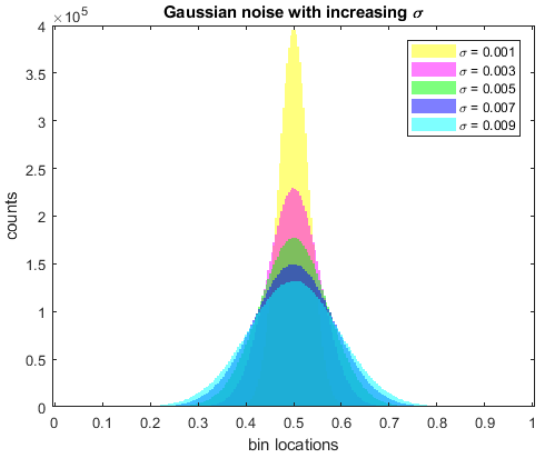
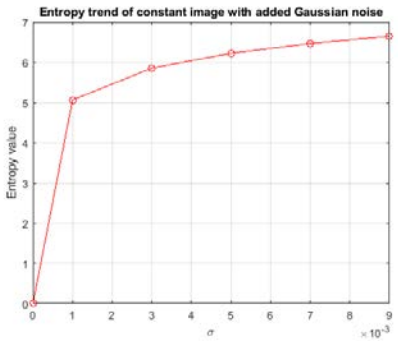
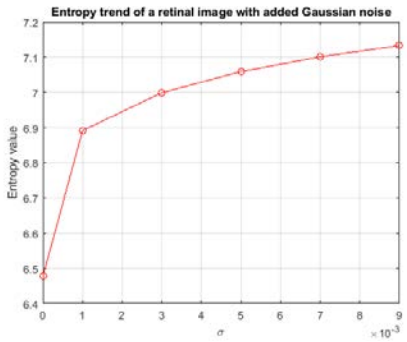


Figure 3.20: Gaussian noise with increasing σ



(a) Result of adding Gaussian noise with increasing σ to a constant image



(b) Result of adding Gaussian noise with increasing σ to a retinal image

Figure 3.21

GLCM is a tabulation how of often a pixel with intensity level i falls next to another pixel with intensity level j in an image, according to a specific spatial relationship: in particular, in this work we considered couples of adjacent pixels (i.e. 1 pixel offset) along 4 fixed spatial directions, introduced in figure 3.22. Moreover, because the computational effort required to compute the GLCM for the full dynamic range of our images is prohibitive, we scaled them in order to reduce the number of intensity levels from 256 to 8. In this way we also avoid ending up with a matrix full of 0, since it is very unlikely that intensity levels very far away in the color scale occur next to each other in the image. By making this choice we determine the size of the GLCM too, which will be an 8x8 matrix for each spatial relationship considered. Once the GLCM is computed, it is normalized by dividing up each of its values for the sum of values of the matrix itself. This allows us to express it as a probability table, even if in an approximate way.

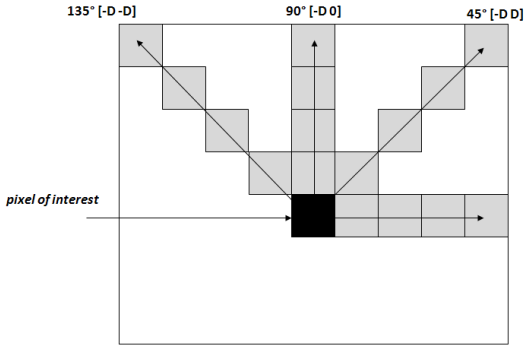


Figure 3.22: Spatial relationships considered for the computation of GLCM, with offset $D = 1$

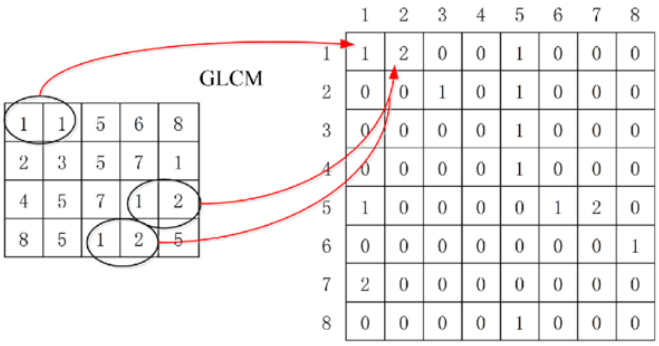


Figure 3.23: Example of GLCM with horizontal spatial relationship

At this point it is possible to compute the Haralick texture features, that are statistics measures defined as weighted averages of the normalized GLCM cell contents [11]. In particular, we have considered:

Contrast $h_1 = \sum_{i,j} |i-j|^2 p(i,j)$
: This descriptor measures the local variations in the GLCM by using weights that increase as the distance from the GLCM diagonal increases: thus, it assumes a value of 0 for a constant image.

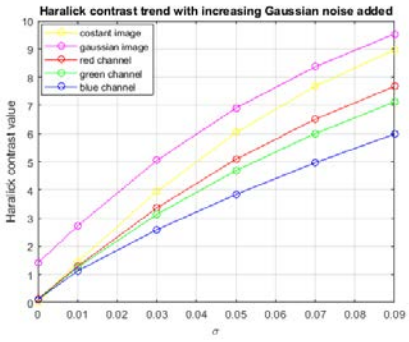
Energy $h_3 = \sum_{i,j} p(i,j)^2$
: This descriptor provides the sum of squared elements in the GLCM: an homogeneous image will be associated with an high value of this descriptor, at the limit of the value 1 relative to a constant image.

Homogeneity $h_4 = \sum_{i,j} \frac{p(i,j)}{1+|i-j|}$
: This descriptor is influenced by the homogeneity level of an image. Because of the weighting factor $(1 + |i-j|)^{-1}$, it gets small contributions from inhomogeneous areas, for which $i \neq j$. Thus, homogeneous images will be associated with high values of this descriptor.

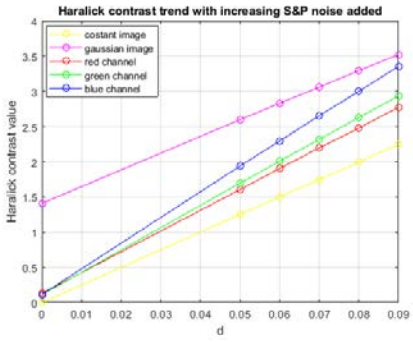
where i and j are respectively row and column indexes, and $p(i, j)$ is the (i, j)-th element of the specified GLCM.

Ultimately, since we computed each of these measures for all the spatial relationships considered, we came up with four values for each feature: the final Haralick features are generated by taking the mean in all directions.

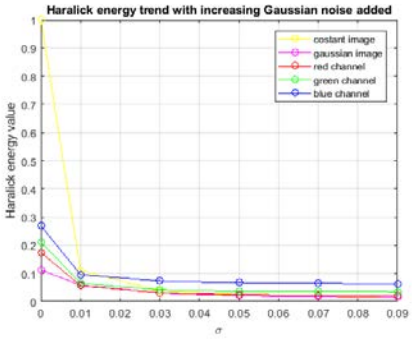
In order to get an insight of how these descriptors might vary, we added different levels of Gaussian and salt and pepper noise to the three color channels of the original image, to a constant image and to a Gaussian phantom image. The results are shown in figure 3.24. The three descriptors are clearly highly correlated, in particular we can appreciate this looking at the trends of the Haralick contrast and homogeneity. However, we believe that their combination can be useful when dealing with the problem of retinal image quality assessment, since they are all very sensitive even to low levels of noise added to images.



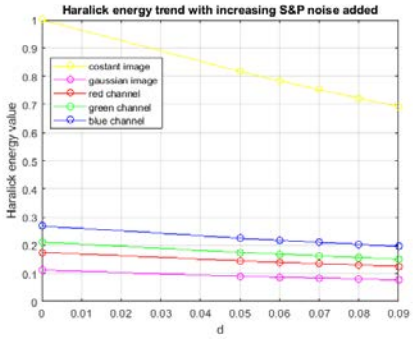
(a) Haralick contrast trend with increasing Gaussian noise added



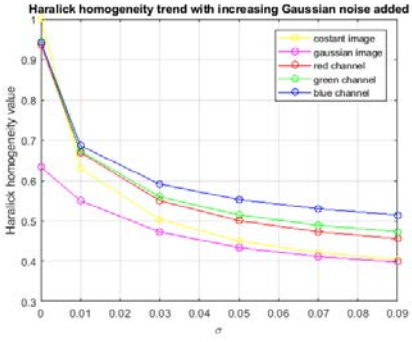
(b) Haralick contrast trend with increasing salt and pepper noise added



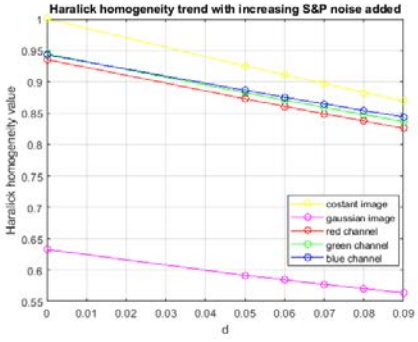
(c) Haralick energy trend with increasing Gaussian noise added



(d) Haralick energy trend with increasing salt and pepper noise added



(e) Haralick homogeneity trend with increasing Gaussian noise added



(f) Haralick homogeneity trend with increasing salt and pepper noise added

Figure 3.24

3.2 Features extraction

Feature extraction is the first fundamental step we need to perform once identified the descriptors of our interest: as a matter of fact, both the heuristics-based algorithm and the SVM do not require as input the orig-

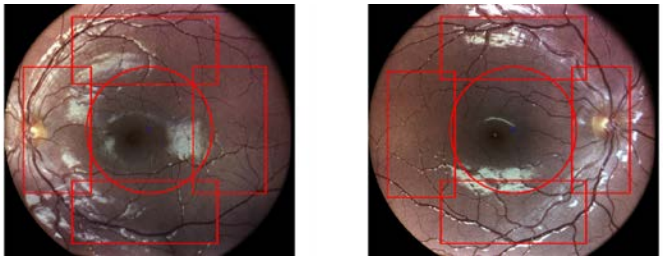
inal images, but a set of features associated with each image present in the data set. Thus, the main goal of features extraction is to capture the most relevant information carried by the original data while simultaneously representing it into a much lower dimensional space [12].

More specifically, even though we decided to focus on images of both the left and right eye with *central* field (cfr 2) when developing the algorithms, in principle we would have to address the problem of retinal image quality assessment also for images with *nasal*, *temporal*, *superior* and *inferior* field. Since these kind of images show different retinal areas, they host in different positions various anatomical landmarks: for example, images with temporal field do not enclose the optic disc, but just the fovea and minor vessels, while images with superior field host the fovea and the optic disc in their inferior part, with a relative positioning depending on which eye is considered.

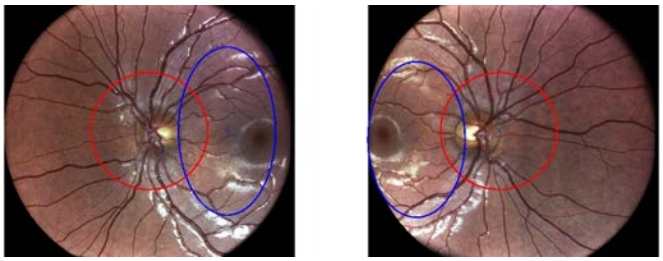
All this variability must be taken into account in order to provide valuable and non-redundant information to the classification algorithms: thus, we designed dedicated regions of interest so as to appropriately deal with all the sort of images that can be acquired.

In figure 3.25 we present all the cases of images introduced above, respectively for the left and the right eye, together with the regions of interest identified on them. In particular, we decided to employ rectangular ROI to capture the information relative to the left, right, superior and inferior portions of images with central field, and circular or elliptic ROI in all the other cases. Moreover, these ROI have been designed manually by using parametric coordinates: thus, they are potentially suitable also for images acquired by other instruments with different fields of view and characteristics.

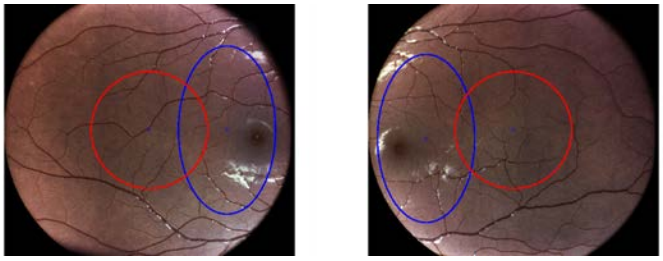
Practically, in the view of developing a fully automated image quality assessment system, we produced a framework which manages these pre processing steps, from image identification to the generation of features vectors, in a total automatic way. First of all, we prepared the images present in our data set so that we can recognize directly from the file name their field and if they are relative to the left or the right eye. Consequently, we have designed the quality evaluation framework by making a loop in which the images are loaded one at a time and then processed: in particular, after their identification, the pixels relative to the appropriate ROI are extracted and used in order to compute all the descriptors introduced in 3.1. Hence, at every iteration a features vector for a specific image is produced and temporarily stored into a matrix, which is progressively filled row by row as images of that same type are processed. Consequently, we end up with 10 different matrices (one for each image type), which are then converted into tables and written into separate Excel files in order to facilitate their readout.



(a) Central field, left eye (b) Central field, right eye

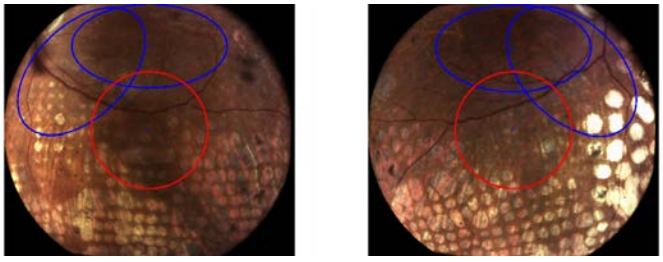


(c) Nasal field, left eye (d) Nasal field, right eye



(e) Temporal field, left eye (f) Temporal field, right eye

(g) Superior field, left eye (h) Superior field, right eye



(i) Inferior field, left eye (j) Inferior field, right eye

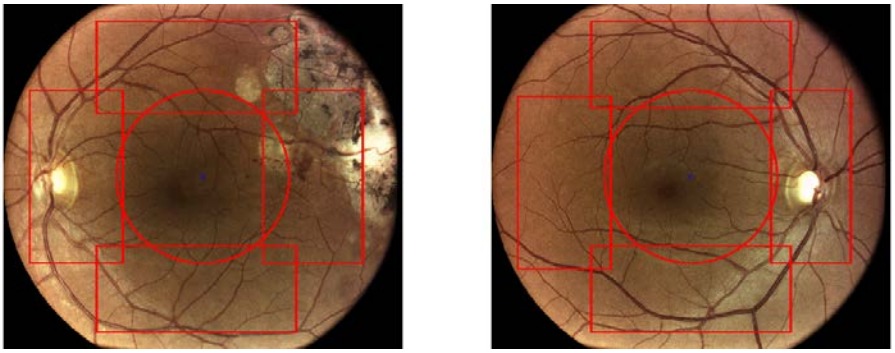
Figure 3.25

3.3 First method: Heuristics

3.3.1 Introduction to heuristics

The first image quality assessment algorithm developed is the most simple among all, because it exploits subsets of the quality descriptors introduced in 3.1 to derive arguments and threshold values capable of discriminating images of different quality.

In particular, with the view of comparing the performances of different methods in solving a same problem, we focused on images of both the right and left eye with *central* field, which are the same used to train the Support Vector Machine classifier described in 3.4. In figure 3.26 we can see an example of such images, along with the regions of interest identified on them.



(a) Left eye, central field

(b) Right eye, central field

Figure 3.26

Notably, the combination of these regions allow us to gain a picture of the *overall* image quality, since all images' significant areas are covered and the black edges discarded [2]. Either way, the left and central ROIs of the left eye and the right and central ROIs of the right eye are the ones to care about the most from a *diagnostic* point of view, since they host the optic disc and the fovea. These anatomical landmarks, together with the vessel component, must remain clearly visible in order to allow the clinician to make a reliable analysis of the fundus oculi.

For this reason, all the heuristics developed require that some specific conditions are verified in *at least* those two regions and one among the three left over. As a matter of fact, we believe that claiming a quality criterion to be satisfied for *all* the regions of figure 3.26 is a too strict demand, since very often our images show a good level of general quality but at the same time some *localized* artifacts, such as partial occlusion by eyelashes in their upper or lower part: nevertheless, this is not a sufficient motivation to classify them as ungradable. Additionally, by making this choice we allow

the algorithm to be a little bit more versatile, in particular when dealing with pathological images, which have completely different characteristics from the healthy ones: an example of this is introduced in figure 3.26a), where the pathological condition only affects the features computed in the right and superior ROI.

3.3.2 Threshold identification process

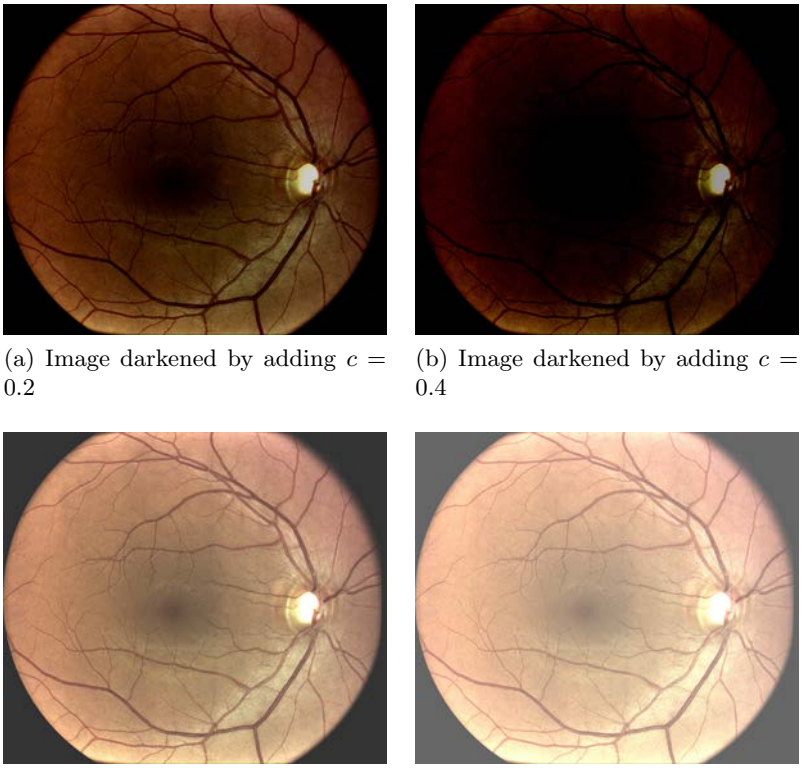
First of all, before building the actual algorithm's architecture, we tried to identify *reasonable* threshold values for all the descriptors introduced in section 3.1 and computed for each ROI of the image: as a matter of fact, making meaningful and effective choices in this preliminary phase of the algorithm is fundamental in order to develop decision rules that can both work well on our data and be generalized.

With this in mind, we employed the same image processing techniques used to validate our descriptors (as explained in section 3.1) in order to modify in different ways a good quality image, which is used as reference, and to generate new little training data sets, each holding that image itself and several its impaired versions (for just one characteristic at a time). Subsequently, we gave these modified images in input to the quality evaluation framework introduced in 3.2, so as to compute the descriptors for every ROI of our interest. Thereby we avoid using the features extracted from our original data set to deduce descriptors' threshold values and adjust them on the basis of the specific patterns contained in it, because otherwise we would enter the field of *supervised* approaches.

Moreover, since the original data set includes images classified as ungradable for a *combination* of reasons, we believe it might be more accurate inferring the threshold values for a descriptor on the basis of images we are sure contain modifications in just that characteristic highlighted by that specific descriptor.

Brightness threshold identification

The first artificial data set built and given in input to the quality evaluation framework is composed by several darkened and lightened images compared to the original one, which instead shows a good level of overall illumination and for this reason is taken as reference. It is generated by respectively adding and subtracting the constant values $c = [0.1, 0.2, 0.3, 0.4]$ to the original image, exploiting the same procedure adopted for the validation of this descriptor (cfr. section 3.1).



(a) Image darkened by adding $c = 0.2$

(b) Image darkened by adding $c = 0.4$

(c) Image lightened by subtracting $c = 0.2$

(d) Image lightened by subtracting $c = 0.4$

Figure 3.27: Examples of darkened and lightened images

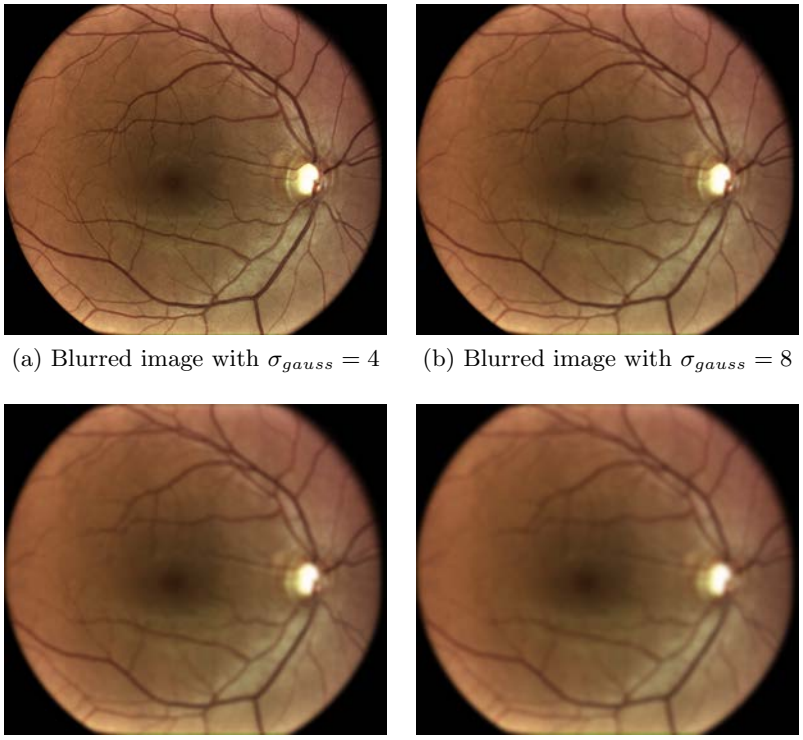
On the basis of the visual result of this operation, which can be appreciated in figure 3.27, we decided to adopt a threshold value only for the underexposure case, because the lightened images show anyway a good level of overall quality. In table 3.2 the threshold values adopted to discard underexposed images are introduced. In particular, they are associated with the brightness values of figure 3.27a).

	Central field
	t_{dark}
Superior ROI	0.1328
Inferior ROI	0.1930
Nasal ROI	0.2215
Temporal ROI	0.1180
Central ROI	0.0729

Table 3.2: Threshold values for the brightness descriptor

Focus threshold identification

Another important artificial data set produced contains images progressively out-of-focus, obtained by blurring an original image with a Gaussian filter of increasing standard deviation $\sigma_{gauss} = [4, 8, 12, 16, 20]$.



(a) Blurred image with $\sigma_{gauss} = 4$ (b) Blurred image with $\sigma_{gauss} = 8$

(c) Blurred image with $\sigma_{gauss} = 16$ (d) Blurred image with $\sigma_{gauss} = 20$

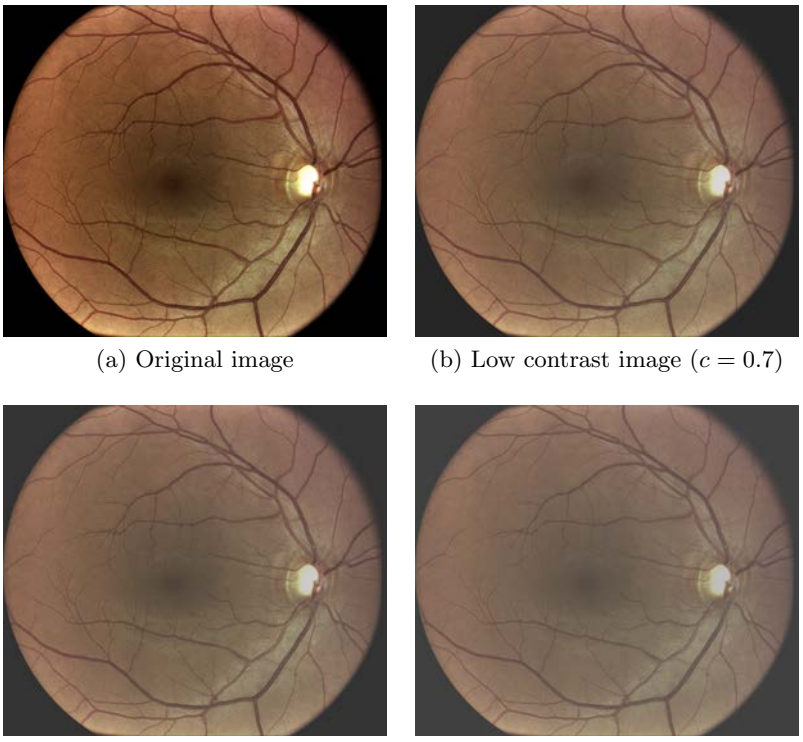
Figure 3.28: Examples of images with increasing artificial blur

The visual result of this operation is introduced in figure 3.28. Despite the image smoothed with Gaussian filter of standard deviation $\sigma_{gauss} = 4$ (figure 3.28a)) could still be considered of good quality for some observers, we prefer to be conservative and develop an algorithm with strict demands, which will discard images even with a low level of blurring: for this reason, we adopt as threshold value for this descriptor the focus value associated with the image blurred with $\sigma_{gauss} = 4$: $t_{focus} = 9.9715 * 10^{-4}$.

Contrast threshold identification

Subsequently, we need to find appropriate threshold values for the contrast descriptor, which is another key element helping us discriminating between good and poor quality images. In order to do so, we produced a data set of images with lowered contrast in comparison with the original one by simply multiplying its pixel intensity distribution in the three color channels for the

following constant values: $c = [0.9, 0.8, 0.7, 0.6, 0.5]$. The details about this operation can be found in section 3.1, and some examples of images with lowered contrast are shown in figure 3.29.



(a) Original image (b) Low contrast image ($c = 0.7$)

(c) Low contrast image ($c = 0.6$) (d) Low contrast image ($c = 0.5$)

Figure 3.29: Examples of images with lowered contrast

This data set has then been given in input to the quality evaluation framework, so as to compute the contrast descriptor for all images' ROIs.

Finally, in table 3.3 we present the threshold values chosen for each ROI, below which we consider the image with a too low level of contrast to be considered as of good quality. These values are associated with the contrast level of image 3.29c).

	Central Field
	t_{contr}
Superior ROI	0.0436
Inferior ROI	0.0622
Nasal ROI	0.0900
Temporal ROI	0.0319
Central ROI	0.0405

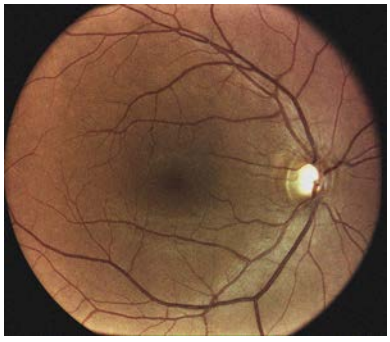
Table 3.3: Threshold values for the contrast descriptor

Entropy and Haralick features threshold identification

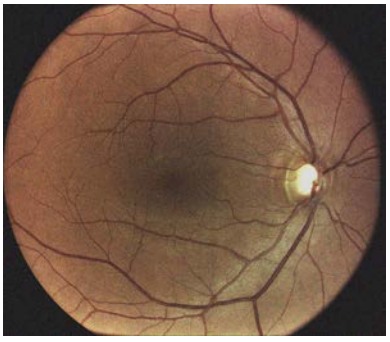
The threshold values for the entropy descriptor and the Haralick features have instead been derived starting from a data set of noisy images, produced by adding Gaussian noise of increasing standard deviation σ_{gauss} to the original good quality image considered as reference.

(a) Original image

(b) Noisy image ($\sigma_{gauss} = 0.03$)



(c) Noisy image ($\sigma_{gauss} = 0.05$)



(d) Noisy image ($\sigma_{gauss} = 0.09$)

Figure 3.30: Noisy images generated by adding Gaussian noise of increasing standard deviation to the original image

More specifically, we employed two different sets of standard deviation values:

- $\sigma_{gauss_1} = [0.001, 0.003, 0.005, 0.007, 0.009]$, to derive proper threshold values for the Haralick features.
- $\sigma_{gauss_2} = [0.01, 0.03, 0.05, 0.07, 0.09]$, to deduce threshold values for the entropy descriptor.

This choice is driven by the considerations made in section 3.1, where we noticed how the Haralick features are sensitive to much lower levels of noise than the entropy descriptor: this is because, although we highlight modifications in their values using the same kind of mathematical trick, they are actually sensitive to different image properties.

In table 3.4 the threshold values chosen for the Haralick features are introduced. In particular, an image would be considered of good quality according to these descriptors if its Haralick contrast value is below t_{h1}, its Haralick energy value is above t_{h3}, and its Haralick homogeneity value is above t_{h4}.

	Central field		
	t_{h1}	t_{h3}	t_{h4}
Superior ROI	0.74423	0.1136	0.7237
Inferior ROI	0.5851	0.1043	0.7557
Nasal ROI	0.4758	0.1236	0.7877
Temporal ROI	0.9464	0.1061	0.6906
Central ROI	0.7945	0.1006	0.7143

Table 3.4: Threshold values for Haralick features

Ultimately, in table 3.5 the thresholds values relative to the entropy descriptor are introduced: on the only basis of this descriptor, an image would be considered as of good quality if its entropy value is below t_{entr}.

	Central field
	t_{entr}
Superior ROI	7.3975
Inferior ROI	7.5563
Nasal ROI	7.6711
Temporal ROI	7.3457
Central ROI	7.2768

Table 3.5: Threshold values for the entropy descriptor

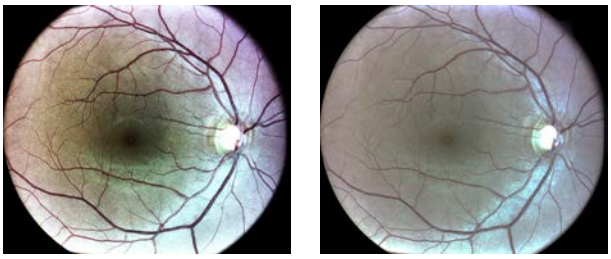
Skewness and kurtosis threshold identification

Skewness and kurtosis play a relative marginal role compared to descriptors such as brightness or focus when dealing with the problem of discriminating images with different quality.

As pointed out in section 3.1, they are useful when considered *in combination* with the brightness descriptor, so as we could theoretically have an information regarding resolution and illumination homogeneity in an image. However, they are not able to provide good and solid results when taken into account *alone*, as we can appreciate by looking at figures 3.31 and 3.32, which show different cases of images processed so as to modify their kurtosis and skewness values. In particular, these images belong to two different artificial data sets:

- a data set of images with different kurtosis values with respect to the one assumed by the original good quality image. Particularly,

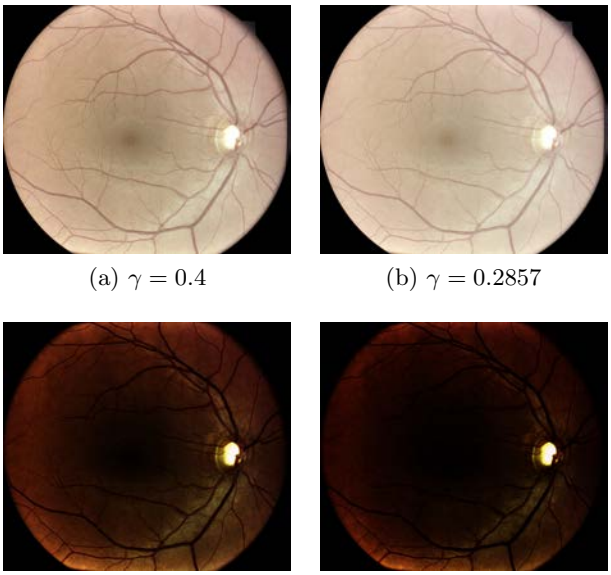
in order to properly modify its pixel intensity distribution, we used the sigmoid transformation introduced in figure 3.10 respectively with $\alpha = [1.2, 1.4, 1.6, 1.8]$ to decrease the kurtosis value and with $\alpha = [0.8333, 0.7143, 0.6250, 0.5556]$ to progressively increase it.



(a) $\alpha = 1.8$ (b) $\alpha = 0.5556$

Figure 3.31: Example of images with different kurtosis value with respect to the original one

- a data set of images with different skewness values with respect to the one assumed by the original good quality image. In order to do this, we exploited the gamma transformation of figure 3.14 with the following values for the parameter γ: $\gamma = [1.5, 2, 2.5, 3, 3.5]$ to progressively increase the skewness value, and $\gamma = [0.6667, 0.5, 0.4, 0.3333, 0.2857]$ to instead progressively decrease it.



(a) $\gamma = 0.4$ (b) $\gamma = 0.2857$

(c) $\gamma = 2.5$ (d) $\gamma = 3.5$

Figure 3.32: Example of images with different skewness value with respect to the original one

Even if the transformations applied to our original image modify its distribution of pixels according to theory, visually the results are images which, al least for the case of the kurtosis descriptor, can still be considered of good quality: thus, it would be hard identifying from them proper threshold values. For this reason, we decided to avoid using the information carried by the kurtosis descriptor, because we could not embed it efficiently into the decision process of this first algorithm.

Besides, figure 3.32b) and 3.32c) show images with modified skewness that can be considered borderline in terms of quality: consequently, we decided to adopt their skewness values as thresholds when discriminating between good and poor quality cases. Nevertheless, the considerations made at the beginning about the marginal role played by this descriptor still hold, so we decided to define these threshold values, which are introduced in table 3.6, only for the nasal ROI.

	Central field	
	$t_{skew_{neg}}$	$t_{skew_{pos}}$
Nasal ROI	0.3384	3.7177

Table 3.6: Threshold values for the skewness descriptor

In this way we make sure that specific cases of images with extreme skewness values (not included into the range identified by the thresholds of table 3.6) are detected and correctly classified as of bad quality.

3.3.3 Heuristics-based algorithm construction

Once appropriate and meaningful threshold values have been identified for all the descriptors of our interest, it is necessary to combine the information carried by each of them and consequently arrange the actual heuristics-based algorithm's structure.

Since the watchwords of this algorithm are *simplicity* and *interpretability*, we built up an architecture with several nested conditions (all based on combinations of our descriptors), in a way that allows to minimize the number of comparisons necessary to understand if an image can be considered of good quality or not. Moreover, every time a classification is performed, it is made clear by a brief description why a certain label (0 or 1) is attached to a specific image. This implies that, in case of rejected image, a real time feedback suggesting what is the cause of the reduced quality can be given to the operator, and so a specific corrective action can be taken immediately.

More in particular, the core of the algorithm is built around conditions on the brightness and the focus descriptor: as a matter of fact, we believe that if an image is underexposed or overexposed and out of focus, it must be classified as of bad quality regardless the value assumed by all the other descriptors, which capture minor aspects of images. Thus, by making this

algorithmic choice, we assign immediately a label of 0 to all the images which do not pass the test over the brightness and the focus descriptors in the nasal, central and one of the remaining ROI of the image.

All the images which instead present a good level of general illumination and focus must satisfy another condition before we assign them a label of 1, because image quality is an elaborate concept, and as such it cannot be expressed by just two descriptors, no matter how much information they bring. This condition combines the contrast descriptor, the entropy descriptor and the Haralick features, and must again be satisfied for the nasal, the central and one among the superior, temporal and inferior ROI (cfr 3.3.1). If an image passes the test over this condition is finally classified as of good quality, otherwise it is associated with a label of 0 even if the condition over the brightness and focus descriptor was verified. If that is the case, a message saying that the image has a too level of contrast and is too noisy to be accepted is given to the operator.

Ultimately, the condition over the skewness descriptor has been put outside this main framework, because with a simple threshold-based algorithm like this one we are not able to efficiently integrate it with the rest of our descriptors. Anyway, we employ it separately in order to be able to identify anyway extreme skewness cases, which must be rejected.

As a concluding remark, it's worth noticing that even if the workflow of this algorithm resembles the one of decision trees in machine learning, the decisions taken here are not the result of a learning process, nor any information regarding the classification error made by the algorithm is used to modify its parameters and gain better performances. On the contrary, the actual data set is never used in order to drive the threshold values identification process introduced in 3.3.2, but only to test algorithm performances.

3.4 Second method: Support Vector Machines

3.4.1 Introduction to Support Vector Machines

Support vector machines (SVM) are a supervised learning technique that can be employed for both classification and regression purposes. Nevertheless, they are most commonly used for classification problems, and as such this is the task we will use them for in this work. In particular, as we need to discriminate between good quality and poor quality images, we are in the setting of a *binary* classification problem.

Additionally, since we are in a supervised learning context, we have at our disposal a training set of labeled examples $S = (\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)$, each consisting of a collection of features: $\mathbf{x_i} \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$.

The goal of SVM, as the one of any other supervised learning technique, is to learn a model on the basis of the training set, able to predict the target values (labels) of test samples, given only their features.

What makes SVM so unique is the way this objective is achieved: the basic idea is to find the best hyperplane that separates all data points of one class from those of the other class. Typically, if a data set is linearly separable, there is an infinite number of possible separating hyperplanes, but the *best* one picked up by SVM is that with largest *margin* between the two classes, so the one that maximizes the distance to the closest data points of both classes.

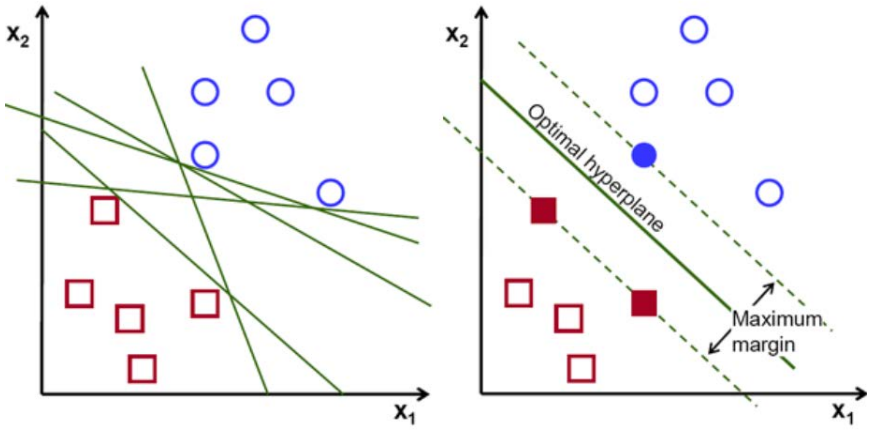


Figure 3.33: Different separating hyperplanes for the same data set

This concept is exemplified in figure 3.33. We can see that the green hyperplanes on the left are all possible choices, since they are able to perfectly separate the training data in the two classes. However, we are interested in classifier's performance on the testing data rather than on the training data: the choice of the maximum margin hyperplane of figure 3.33b) allows a testing point to fall between the hyperplane itself and the outer dotted line

of the margin, and to be anyway correctly classified. Conversely, any green hyperplane of Figure 3.33a) could not handle a situation like the one just described (that besides is rather common in practice) without introducing a misclassification error.

The maximum margin hyperplane is then identified by the so called *support vectors*, that are the highlighted points in Figure 3.33b).

SVM can then also be successfully employed when the training data are not linearly separable or in cases in which a linear decision boundary does not exist in the original input space. Those situations require little modifications of the hard SVM approach presented in 3.4.2.1, and they will be addressed respectively in section 3.4.2.2 and 3.4.2.3.

3.4.2 Mathematical formulation

Our training data consist of the set $S = (\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)$ collected during the features extraction step described in section 3.2. The labels y_i can be either 1 or 0, depending on the class to which the corresponding point $\mathbf{x_i}$ belongs. Moreover, each $\mathbf{x_i}$ is a d-dimensional real vector containing the features extracted from an image.

As explained in section 3.4.1, we want to find the maximum margin hyperplane that allows to separate the points belonging to the two classes we are dealing with (good quality vs poor quality images). First of all, we start giving the following mathematical formulation of an hyperplane: it is defined through $\mathbf{w}, b$ as the set of points such that $\mathcal{H} = \{\mathbf{x} | \mathbf{w}^\mathsf{T}\mathbf{x} + b = 0\}$, where $\mathbf{w} \in \mathbb{R}^d$ is the normal vector to the hyperplane, and b is a real number. The margin γ can then be defined as the distance from the hyperplane to the closest points of the two classes. Broadly speaking, the distance of a generic point $\mathbf{x}$ to the hyperplane $\mathcal{H}$ can be derived as follows: we consider $\mathbf{d}$ as the vector of minimum length that goes from $\mathcal{H}$ to $\mathbf{x}$, and $\mathbf{x}^P$ the projection of $\mathbf{x}$ onto $\mathcal{H}$. Consequently, we can write that:

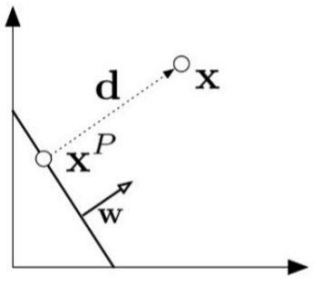


Figure 3.34

$$\mathbf{x}^P = \mathbf{x} - \mathbf{d} \tag{3.4}$$

Additionally, as we can see from Figure 3.34, $\mathbf{d}$ is parallel to $\mathbf{w}$, so it can be stated that, for some $\alpha \in \mathbb{R}$:

$$\mathbf{d} = \alpha\mathbf{w} \tag{3.5}$$

Furthermore, $\mathbf{x}^P \in \mathcal{H}$, which implies:

$$\mathbf{w}^\mathsf{T}\mathbf{x}^P + b = 0 \tag{3.6}$$

Substituting (3.4) and (3.5) into (3.6), we obtain that $\mathbf{w}^\mathsf{T}(\mathbf{x} - \alpha\mathbf{w}) + b = 0$. It follows that:

$$\alpha = \frac{\mathbf{w}^\mathsf{T}\mathbf{x} + b}{\mathbf{w}^\mathsf{T}\mathbf{w}} \tag{3.7}$$

We can now define the length of $\mathbf{d}$ as:

$$\|\mathbf{d}\|_2 = \sqrt{\mathbf{d}^\mathsf{T}\mathbf{d}} = \sqrt{\alpha^2\mathbf{w}^\mathsf{T}\mathbf{w}} = \frac{|\mathbf{w}^\mathsf{T}\mathbf{x} + b|}{\sqrt{\mathbf{w}^\mathsf{T}\mathbf{w}}} = \frac{|\mathbf{w}^\mathsf{T}\mathbf{x} + b|}{\|\mathbf{w}\|_2} \tag{3.8}$$

Finally, we can derive that the margin of $\mathcal{H}$ is:

$$\gamma(\mathbf{w}, b) = \min_{\mathbf{x}} \frac{|\mathbf{w}^\mathsf{T}\mathbf{x} + b|}{\|\mathbf{w}\|_2} \tag{3.9}$$

3.4.2.1 Hard margin SVM

The hard SVM rule can be expressed in terms of a *constrained* optimization problem [13]:

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \gamma(\mathbf{w}, b) \\ \text{s.t.} \quad & \forall i \;\; y_i(\mathbf{w}^\mathsf{T}x_i + b) \geq 0 \end{aligned} \tag{3.10}$$

As a matter of fact, the objective is to maximize the margin under the constraint that all data points lie on the correct side of the hyperplane. If we substitute in (3.10) the definition of margin, we obtain:

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \frac{1}{\|\mathbf{w}\|_2} \min_{\mathbf{x}} |\mathbf{w}^\mathsf{T}\mathbf{x} + b| \\ \text{s.t.} \quad & \forall i \;\; y_i(\mathbf{w}^\mathsf{T}\mathbf{x_i} + b) \geq 0 \end{aligned} \tag{3.11}$$

At this point we can use the fact that the hyperplane is *scale invariant*: $\gamma(\beta\mathbf{w}, \beta b) = \gamma(\mathbf{w}, b), \quad \forall \beta \neq 0$. It means that $\mathbf{w}$ and b can be multiplied by any constant value, but either way they are going to identify the same hyperplane. Therefore, we make the following choice to fix the scale:

$$\min_{\mathbf{x}} \quad |\mathbf{w}^\mathsf{T}\mathbf{x} + b| = 1 \tag{3.12}$$

The re-scaling expressed in (3.12) is an equality constraint, that substituted in (3.11) leads to the following reformulation of the problem:

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \frac{1}{\|\mathbf{w}\|_2} \\ \text{s.t.} \quad & \forall i \;\; y_i(\mathbf{w}^\mathsf{T}\mathbf{x_i} + b) \geq 0, \\ & \min_{\mathbf{x}} |\mathbf{w}^\mathsf{T}\mathbf{x} + b| = 1 \end{aligned} \tag{3.13}$$

Finally, given the fact that maximizing $\frac{1}{\|\mathbf{w}\|_2}$ is equivalent to minimizing $\|\mathbf{w}\|_2$ and that, for the optimal solution, the two constraints of equation (3.13) con be combined, we come up with the final formulation:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \|\mathbf{w}\|_2 \\ \text{s.t.} \quad & \forall i \;\; y_i(\mathbf{w}^\mathsf{T}\mathbf{x_i} + b) \geq 1 \end{aligned} \tag{3.14}$$

Consequently, hard SVM searches for $\mathbf{w}$ of minimal norm among all the vectors that are able to separate the data and for which the constraint of equation 3.14 holds: finding the maximum margin hyperplane boils down finding $\mathbf{w}$ whose norm is minimal.

This new formulation is a quadratic optimization problem (the objective is quadratic and the constraint is linear) known as the *primal* problem, that can be efficiently solved using a quadratic programming (QP) code. Exploiting the theory of Lagrange multipliers, we can also look at this problem from another perspective and derive the so called *dual* formulation, that will be key in addressing the kernel trick of section 3.4.2.3. First of all, in order to obtain the dual, we consider the Lagrange multipliers α_i multiplied by each constraint of (3.14), and we subtract these terms from the objective function:

$$L = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} - \sum_{i=1}^{n} \alpha_i(y_i(\mathbf{w}^\mathsf{T}\mathbf{x_i} + b) - 1) \tag{3.15}$$

The goal is again to find the optimal values for $\mathbf{w}$ and b. Once α is fixed, we can notice that the problem with respect to $\mathbf{w}$ is unconstrained and the objective is differentiable. Consequently, we set the gradient of L to 0, and we obtain for the optimal solution:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x_i} \tag{3.16a}$$

$$\sum_i \alpha_i y_i = 0 \tag{3.16b}$$

In particular, equation (3.16a) shows that the vector $\mathbf{w}$ can be expressed as a linear combination of the input data: consequently, it is never necessary to compute the full vector $\mathbf{w}$, we just need to know the values of $\alpha_i, \forall i = 1, ..n$.

Finally, substituting (3.16a) and (3.16b) into equation (3.15), we obtain the final dual formulation, that shows how the optimization problem only depends on the inner product of input samples:

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x_i}^\mathsf{T}\mathbf{x_j} \tag{3.17}$$

This function has to be maximized over $\alpha_i \geq 0$.

3.4.2.2 Soft margin SVM

The problem with the hard SVM formulation is that it assumes the training data to be *linearly separable*: it is a rather strong assumption, since in many practical applications is not verified, and therefore there is not a solution to the optimization problem previously stated. Soft SVM has been introduced as a relaxation of the hard SVM rule, that allows the constraint of equation 3.14 to be violated for some of the examples in the training set: this is done with the introduction of the so called slack variables ξ_i. These variables allow the input data to be closer to the hyperplane, or even on its wrong side, as we can see from Figure 3.35. The optimization problem is then

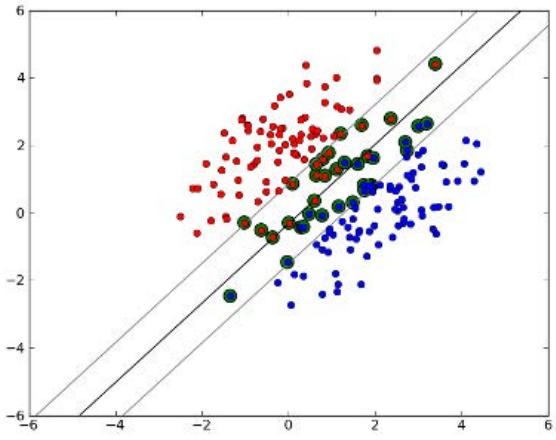


Figure 3.35: Soft margin SVM

restated in the following way:

$$\begin{aligned} \min_{\mathbf{w},\, b} \quad & \|\mathbf{w}\|_2 + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & \forall i \;\; y_i(\mathbf{w}^{\mathsf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i, \\ & \forall i \;\; \xi_i \geq 0 \end{aligned} \tag{3.18}$$

The objective is to jointly minimize the norm of $\mathbf{w}$ (corresponding to the margin) and the average of ξ_i (corresponding to the violation of the constraints). The parameter C is a penalty parameter, that makes a trade off between the two terms of the cost function: if it is set to a large value, it means that it's very expensive for the constraint to be violated, so the SVM becomes very strict and tries to get all points to be on the correct side of the hyperplane, at the expense of solution's regularity. Vice versa, if C has a small value, SVM admits more "slack" and is more loose. The final result

is then a balance between the need of having a model that is both simple and well explanatory of the useful information carried by the input data.

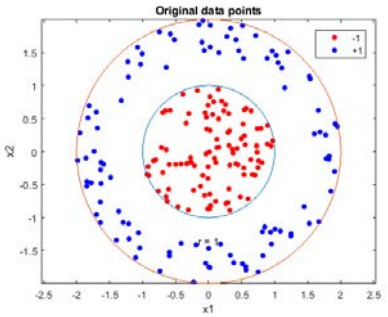
Even in this case of soft SVM we can derive a useful *dual* formulation, following the same procedure adopted previously:

$$\begin{aligned} \max_{\alpha_1,...,\alpha_n} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x_i}^\mathsf{T} \mathbf{x_j} \\ \text{s.t.} \quad & \sum_i y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{3.19}$$

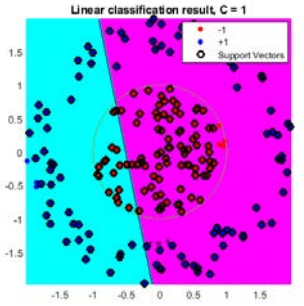
The only difference from equation (3.17) is the second constraint, that basically keeps the allowable values for the Lagrange multipliers in a bounded region between 0 and C.

3.4.2.3 Kernel trick

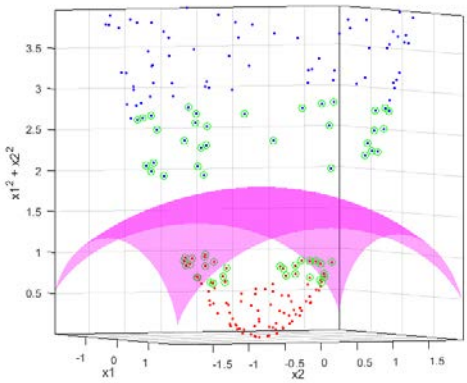
In many practical classification problems it can happen that a linear decision boundary does not exist: such cases can still be handled with SVM, by employing the so called *kernel trick*. It basically allows us to use the framework of linear classifiers in a much higher dimensional feature space, with on top a computationally efficient implementation. Formally, we can apply the following feature transformation to our input training vector $\mathbf{x} \in \mathbb{R}^d$: $\mathbf{x} \mapsto \phi(\mathbf{x})$, where $\phi(\mathbf{x}) \in \mathbb{R}^D$. Usually, $D \gg d$, because our objective is to add dimensions capable of capturing non linear interactions among the original features, in order to make them more expressive of the variance of our data. Consequently, this new representation is very effective in decreasing bias, since it makes the image of data linearly separable in the feature space. In figure 3.36a) we can see an example of data non linearly separable in the original input space: clearly, if we use the soft margin SVM classifier in such a situation we would obtain miserable results in terms of performances, as shown in figure 3.36b). By applying the simple feature mapping: $\phi : (\mathbf{x_1}, \mathbf{x_2}) \mapsto (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_1^2} + \mathbf{x_2^2})$ data are projected into a 3D space, where they become linearly separable and so a linear separating hyperplane can be found (in this case a plane, as we can see from figure 3.36c)).



(a) Data points in $\mathbb{R}^2$



(b) Failing of linear classifier



(c) Projection of data points in $\mathbb{R}^3$

Figure 3.36

However, the problem is that $\phi(\mathbf{x})$ might become very high dimensional in practical applications, so performing its calculation and store it in memory would be unbearable for computational issues. In addition, we would need many more samples in order to learn a classifier in this higher dimensional feature space.

A common solution to deal with this problem is the so called kernel trick, that owes its name to the use of *kernel functions*: they enable us to work in a high dimensional feature space without the need of expressing the explicit mapping ϕ or even computing the entire vector $\phi(\mathbf{x})$. As a matter of fact, as we can see from the dual formulation of equation (3.19), the classifier only depends on the inner product computation between all pairs of samples. Consequently, by applying the theory of maximum margin classifiers to the *mapped* data set, we can reformulate (3.19) in the following way:

$$\begin{aligned} \max_{\alpha_1,...,\alpha_n} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi(\mathbf{x_i})^\mathsf{T} \phi(\mathbf{x_j}) \\ \text{s.t.} \quad & \sum_i y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{3.20}$$

Besides, we can define the kernel function as $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^\mathsf{T}\phi(\mathbf{x_j})$: it takes the feature vector $\mathbf{x}$ as input and directly returns the value of the inner product of the mapped data points in a high dimensional feature space. The main advantage of such a kernel function is that the complexity of the optimization problem remains bounded by the dimensionality of the input space, not the feature space, and for this reason the *kernelized* SVM formulation is very computationally efficient.

3.4.3 Proposed procedure

In the following we show the adopted procedure and the reasoning at the basis of the SVM framework adopted.

First of all, since SVM requires that each data instance is represented as a vector of real numbers, we need to process our images in order to extract from them the descriptors introduced in section 3.1. This is done in a specific features extraction step, reported in section 3.2, whose output is a table in which each row corresponds to an image and each column to a feature computed for that image. Notably, the last column of this table contains the labels assigned to the images by an expert human observer, which will be used as ground truth when evaluating the performances of this method.

Before starting the tuning phase of the algorithm, we need to perform an important pre processing step, that consists in *scaling* the features: this is also known as data *normalization*, and has a critical influence in all the subsequent operations carried out by the machine learning algorithm. As a matter of fact, if the range of data varies widely, this might lead to numerical difficulties during the calculations. Equations 3.17 and 3.19 clearly show how the optimization problem only depends on the inner products between data instances: consequently, we want to avoid having attributes in broader numerical ranges dominating others in smaller numerical ranges. In particular, the normalization is performed by centering and scaling each column of the table containing the training data by the weighted column mean and standard deviation, respectively. Notably, the same scaling is then used for the correspondent validation set when evaluating model performances (in a cross validation setting).

Moreover, since we don't have any missing value in the table, we don't need to perform additional pre processing steps or any kind of data cleaning,

so the normalized table of data is ready to be used as input of our learning algorithm.

The first matter we need to address after data pre processing is the non linearity of our problem. As outlined in 3.4.2.3, the solution is kernel based learning, so we need to make a proper choice of the kernel function. Although we could define a custom kernel, the Gaussian one turned out being notably suitable for our specific application, since it enables us to operate in an *infinite* dimensional feature space:

$$K(\mathbf{x_i}, \mathbf{x_j}) = e^{-\gamma\|\mathbf{x_i}-\mathbf{x_j}\|^2} = e^{-\frac{\|\mathbf{x_i}-\mathbf{x_j}\|^2}{\sigma^2}} \tag{3.21}$$

with $\gamma > 0$.

In general, a kernel function can be interpreted in terms of *similarity* measure between vectors: the Gaussian kernel expresses it as a decaying function of the Euclidean distance between vectors. That is:

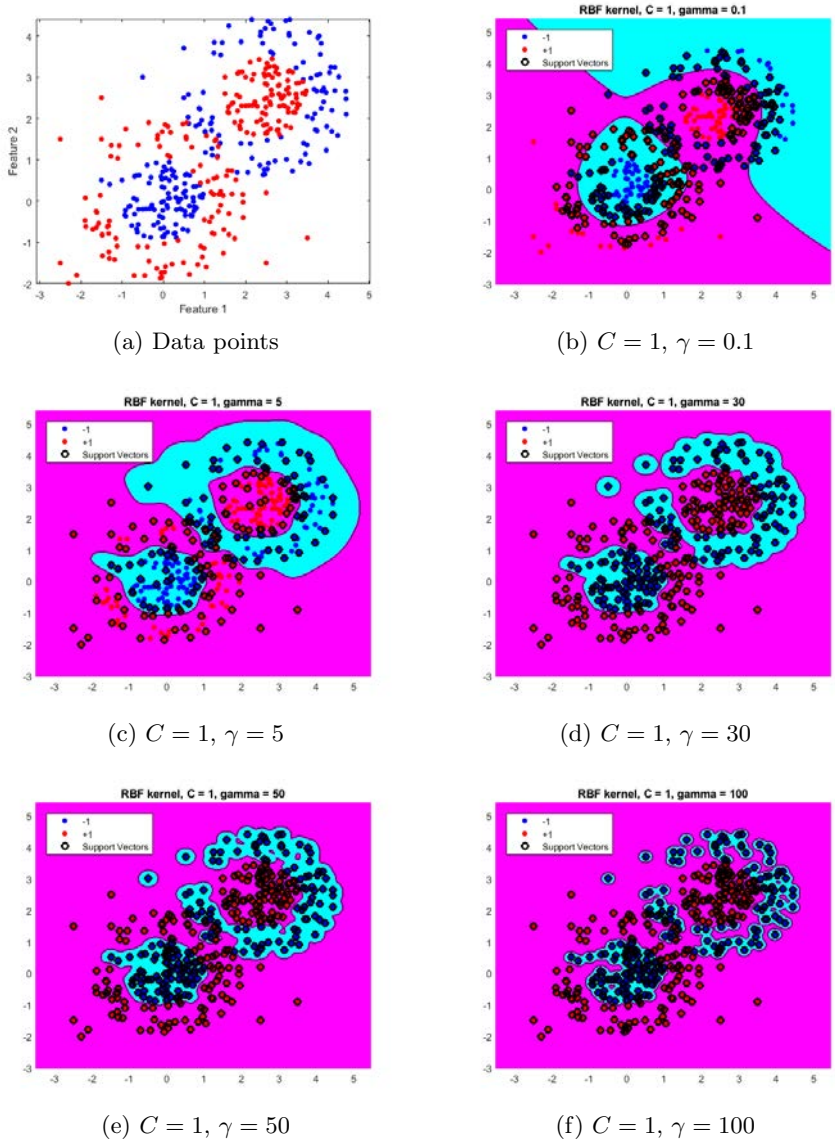
- the more $\mathbf{x_i}$ and $\mathbf{x_j}$ are close together, the more $\|\mathbf{x_i}-\mathbf{x_j}\|^2$ will be small and $K(\mathbf{x_i}, \mathbf{x_j})$ moves towards 1.
- on the contrary, the more $\mathbf{x_i}$ and $\mathbf{x_j}$ are far away, the more $\|\mathbf{x_i}-\mathbf{x_j}\|^2$ will be large and $K(\mathbf{x_i}, \mathbf{x_j})$ get close to 0.

Thus, closer vectors have a larger Gaussian kernel (i.e. are more *similar*) than farther vectors.

The parameter γ controls the spread of the Gaussian kernel and basically defines how far the influence of each support vector reaches, so it is an index exploited to weight the distance between vectors:

- the more γ is large, the more the Gaussian function is narrow and thus selective: it is sufficient a little displacement of a sample point from one of the support vectors to bring K close to 0.
- if γ is set to a small value, then the Gaussian function is less selective, and thus K goes to 0 more slowly as the distance between points and support vectors increases.

Consequently, γ has a nice interpretation as the reverse of the radius of a sphere centered on the support vectors: all the points inside the sphere can be considered *similar* to the support vector at the center. So the more γ is large, the more the radius of each sphere is small, and thus the number of points that will be considered similar to the support vector at the center will be less. This concept is exemplified in figure 3.37. The choice of a proper value for such a parameter is not a trivial task, since it takes into account different aspects of the problem: for example, if we expect to have a lot of noise affecting our features, it would be wise not to choose a too large value for γ. As a matter of fact, as γ increases the more complex



(a) Data points

(b) $C = 1$, $\gamma = 0.1$

(c) $C = 1$, $\gamma = 5$

(d) $C = 1$, $\gamma = 30$

(e) $C = 1$, $\gamma = 50$

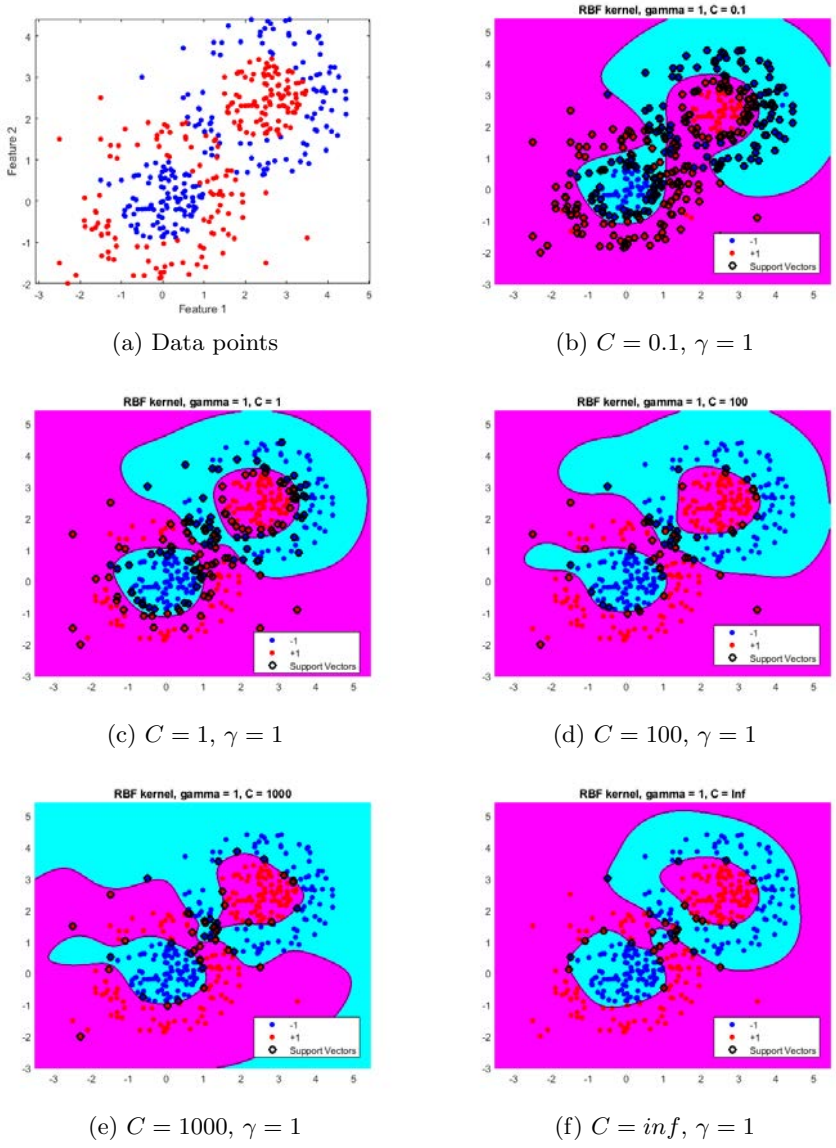
(f) $C = 1$, $\gamma = 100$

Figure 3.37: Role of γ: decision boundaries for a non linear classification problem with different hyper parameter values (C fixed, γ varied)

the model becomes: we can see this from the increasing number of support vectors that are selected by the classifier to build the decision boundary in figure 3.37. In the extreme case of $\gamma = 100$ all the points are considered support vectors, and the *training* error goes down to zero: we are in a clear overfitting condition.

Before moving to the explanation of the tuning phase of our algorithm, it is important to address the role of the other important hyper parameter of the algorithm, which is the penalty factor C introduced in 3.4.2.2. Since

our problem has a huge number of features, we prefer to give an intuition of what is the effect of different values of C with the same simple example used above to understand the role of γ, but this time keeping γ fixed to 1 and varying C. As we can see from figure 3.38, the more we increase C, the more



(a) Data points

(b) $C = 0.1$, $\gamma = 1$

(c) $C = 1$, $\gamma = 1$

(d) $C = 100$, $\gamma = 1$

(e) $C = 1000$, $\gamma = 1$

(f) $C = inf$, $\gamma = 1$

Figure 3.38: Role of C: decision boundaries for a non linear classification problem with different hyper parameter values (γ fixed, C varied)

costly becomes to violate the constraint of equation 3.18: consequently, the classifier tries to become more flexible in order to get all points to the correct side of the margin, at the expense of solution regularity. Conversely, with small values of C we allow the model to misclassify some training data in

the interests of obtaining a smoother solution.

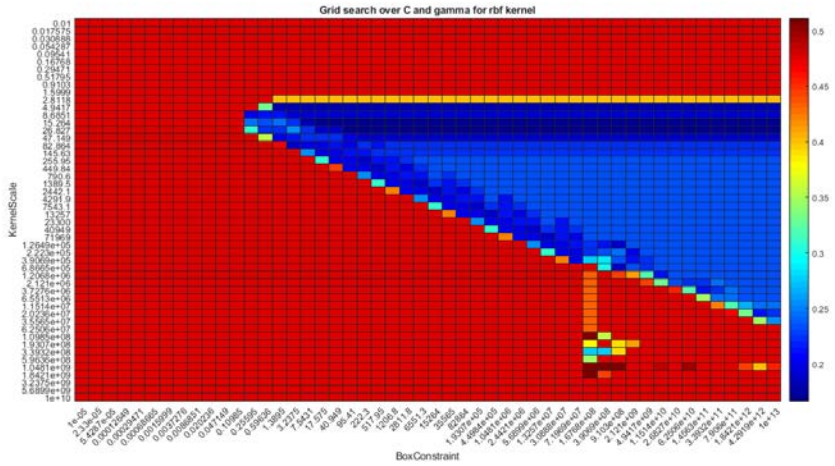
Once understood the role played by both C and γ with a simple example, we now need to find the *optimal* hyper parameter values for our specific problem, choosing their best possible *combination*. As a matter of fact, an optimization procedure carried out over C and γ together is key in order to build a classifier which not only works well on training data, but most importantly is able to generalize to unseen data. Consequently, we decide to devote half of our entire data set to this tuning phase of the algorithm: with this choice of a sufficient large and thus representative sample of our data we are confident that the couple (C, γ) picked in the end is entirely reasonable. The approach adopted in this work for the hyper parameter optimization is a telescopic grid search: basically, this method consists in evaluating different combinations of (C, γ) values in order to find the one with the best cross-validation accuracy.

More in detail, we further divide our "tuning" set into K different folds: for all the couples (C, γ) considered, each of these K folds is used in turn to validate the classifier trained on the others $K - 1$ folds. At the end of this cross validation process we come up with K measures of performances, which can be averaged in order to produce a single error estimation for each pair (C, γ). For our specific application we observed that a value of $K = 2$ is a proper setting in order to both obtain reliable performance measures and avoid overfitting.

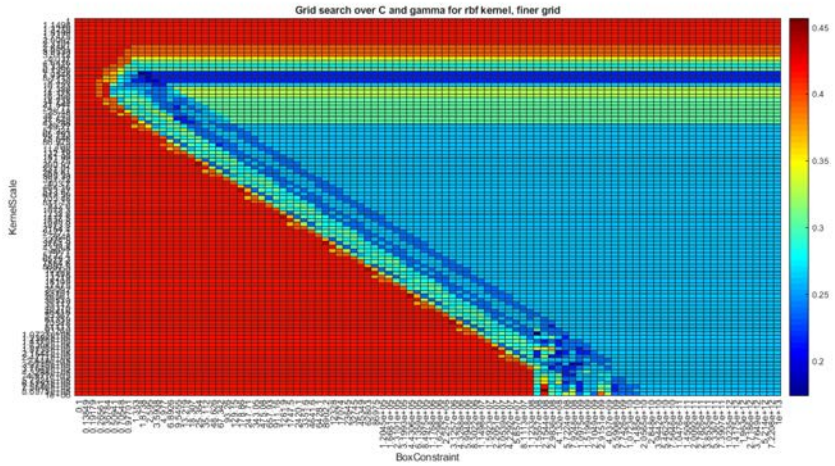
Another point that we need to take into account is that an exhaustive search over C and γ can be really time consuming and expensive in terms of computational power. For this reason we decide to start with a coarse grid, so as to locate on that a "good" region where it is worth focusing and doing a finer search: these steps are shown in figure 3.39, where the areas in which the misclassification error is lower are highlighted with shades of blue.

According to the theory, we should now pick the couple (C, γ) associated with the highest cross validation accuracy, which is placed on the dark blue point on the top of the blue area of figure 3.39a), associated with the value of $C = 1.3895$ and $\gamma = 8.6851$. Nevertheless, as we can see from that same figure, there are many possible and reasonable choices for these parameters: we prefer to be conservative and settle down approximately in the middle of the blue area, instead of picking the couple that works best on our training data, but is placed close to a region (the green one of figure 3.39b)) where performances get quickly worse. In this way we certainly avoid achieving high training accuracy and thus overfitting our data.

The next steps we need to perform after tuning algorithm's hyper parameters are training the classifier and testing its performances. Ideally, this last step should be carried out on a data set completely independent from the ones exploited for model tuning and training, in order to get unbiased performance measures. However, we can't afford to discard any data, so



(a) Coarse grid



(b) Finer grid

Figure 3.39: Grid search over C and γ with 2-fold cross validation

this approach is not suitable for our purposes: a possible solution would be devoting a very small portion of data set for model tuning and then discard it, but in this case we could hardly obtain reliable values for C and γ. Consequently, we decide to adopt a strategy that allows us to employ all data available, even if in this way we get performance measures that we can trust approximately at 50%.

First of all, we put together again the "tuning" data set with the rest of our samples. Then, we perform 3 fold cross validation on this set keeping the values of C and γ fixed to the one selected in the previous step, using in turn each fold as validation set for the model trained on the other two folds. Conceptually, we know that:

- one fold will be composed by data never seen by the algorithm before. Consequently, the "partial" performance measures derived from this fold are completely trustworthy.
- one fold will instead contain data totally used for algorithm's tuning. Thus, we know that the performance measures resulting from this fold are going to be overoptimistic.
- one fold will be composed by roughly half unseen data and half tuning data. Therefore, we obtain a performance measure that, in terms of reliability, is intermediate compared to the ones obtained in the other two folds.

Nevertheless, the data set partition we select *randomly* divides the observations into 3 disjoint subsets, so we are not actually getting folds composed as described above. However, the results obtained in practice are exactly the same, because independently on how we divide the data, we know for sure that half of them have been used by the algorithm and half are new for it.

3.5 Third method: Deep Learning

3.5.1 Introduction to Deep Learning and CNN

The third and last method that we want to develop in this work addresses the retinal image quality assessment problem through a deep learning approach. As a matter of fact, even if with the SVM-based method previously described in section 3.4.3 we expect to gain good performances, in the last few years deep learning has became progressively more integrated with ophthalmic care [14], thanks to the remarkable performances demonstrated.

Basically, deep learning is a branch of machine learning that attempts to model high-level abstractions in data by using multiple processing layers [15]. In particular when dealing with images, the tool of Convolutional Neural Networks (CNN) is adopted, because it holds the important property of translation invariance and allows us to contain the number of parameters the network has to learn with respect to a fully connected neural network. As a matter of fact, in order to use the conventional framework of neural networks for image recognition, we would have to vectorize images before feeding them as input to the network, consequently loosing all the information deriving from their 2D structure. Moreover, in that scenario, each image pixel would turn into a neuron of the network, and consequently the number of weights and biases to learn would dramatically increase. On the contrary, with CNN we can directly give the entire images as input to the network, which then processes them with a succession of hidden layers, before producing the final classification in output. This way to proceed strongly differentiates this approach to both the heuristics and SVM-based methods previously developed, which instead rely on handcrafted and well interpretable features to extract useful information from data.

Nevertheless, CNN's number of parameters is anyway huge, so a very large data set is necessary in order to make the training of such a network from scratch, with randomly initialized weights. For this reason, we designed an architecture with a limited number of hidden layers, so as to be sure a proper training could be performed with the number of data at our disposal. The details about this designing phase of the network are reported in section 3.5.2. Then, in order to make a fair comparison between the performances achieved with this method and the ones obtained with the previous two approaches developed, we had to *fine-tune* the network so as it could be able to classify images acquired with the prototype: section 3.5.3 reports all the steps of this *transfer learning* process, together with the choices made during network's training.

3.5.2 Proposed CNN architecture

The deep convolutional neural network architecture adopted is shown in figure 3.40, and it is composed by two convolutional layers, both followed by a batch normalization layer, a ReLu activation function and a max pooling layer. Then, there is a fully connected layer as last learnable layer before the softmax function and the classification output.

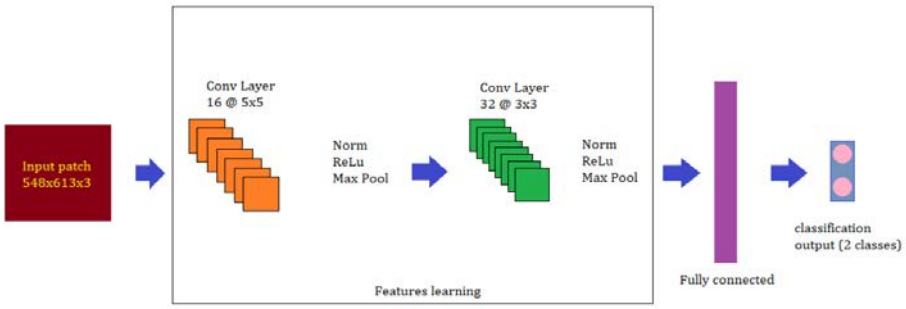


Figure 3.40: CNN architecture developed

Basically, network structure starts with an *image input layer*, which takes as input randomly selected patches of size 548x613x3 from our original images. As a matter of fact, even if in general with CNN we are allowed to give raw images in input to the network, so as it can learn on its own significant features and patterns in data, we anyway performed some pre-processing steps in order to reduce their initial size. This is because images of Eidon data set employed to train the network are of size 3288x3680x3 pixels, thus, without any kind of downsizing, intermediate storing of training data batches resulted unfeasible both in terms of memory and speed requirements.

Consequently, we adopted the following solution: first of all, we selected the squared ROI shown in figure 3.41 from all our original images, so as to discard the black edges at their corners. Then, random patches of size 548x613x3 are cropped from these squared ROI and finally given in input to the network, which thus is trained with images of dimension equal to $\frac{1}{6}$ of their original size. This solution, as can be appreciated from figure 3.42, allows also to improve network's ability to generalize and correctly label images with some sort of distortion, given that these patches are taken randomly and as such they show different retinal image areas.

These patches are then processed through a *convolutional layer*, consisting of 16 learnable kernels of size 5x5: basically, such a layer applies sliding convolutional filters to the input by computing the dot product between

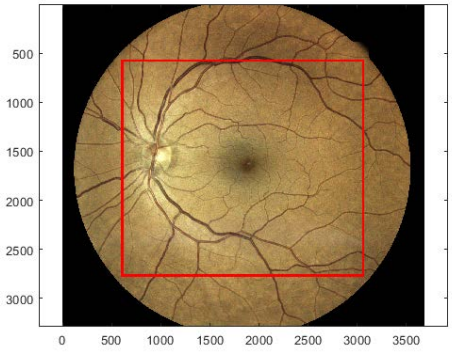


Figure 3.41: Squared ROI selected

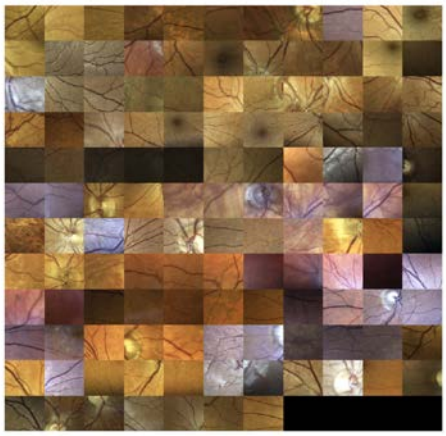


Figure 3.42: Examples of random patches

the input itself and its weights, adding then a bias term. This operation produces 16 feature maps, subjected to a *batch normalization* before going through the *ReLu activation function*, which effectively removes negative values from such features maps by performing a threshold operation. More specifically, the normalization step placed between the convolutional layer and the non linearity is performed in order to reduce network's sensitivity to weights and biases random initialization.

In the end, a *max pooling layer* performs a downsampling operation of the feature maps previously produced so as to reduce their size and remove redundant information, before giving them in input to a second convolutional layer, this time consisting of 32 kernels of size 3x3, followed again by a succession of batch normalization, ReLu and max pooling layers. By way of illustration, figure 3.43 provides an example of features maps produced in output by the second convolutional layer of the network.

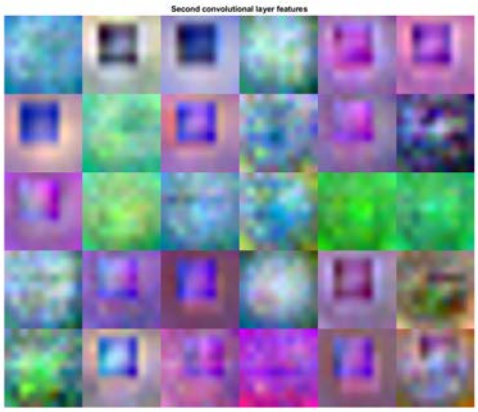


Figure 3.43: Examples of features extracted by network's second convolutional layer

The last layer with learnable weights is a *fully connected layer*, with a number of outputs equal to 2, since we are dealing with a binary classification problem. This layer contains information on how to combine the features extracted by network's previous layers, and it is followed by a *softmax layer* and a *classification layer*, which respectively produce in output class probabilities and final labels.

3.5.3 CNN training and transfer learning procedure

Once network's architecture has been designed, the training step has been performed by using the Eidon data set introduced in section 2, which in particular consists of 2242 images manually labeled by an expert human observer. We decided to devote 90% of these images to network's training, and the remaining 10% to network's validation and testing phase.

In particular, the network has been trained by using the stochastic gradient descent algorithm (SGD), which updates network's parameters (weights and biases) in order to minimize a *non-convex* loss function [16, 17]: this represents one of the major differences between this approach and the SVM method introduced in section 3.4, which instead is a *convex* optimization problem, and as such it is guarantee that the optimal solution is reached at the end of the minimization process, regardless parameters' initialization. On the contrary, with neural networks parameters' initialization matters, and there are several options we need to properly set in order to find a good solution.

More specifically, with the SGD algorithm the loss function is minimized

by taking small steps in the direction of the negative gradient of the loss:

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) \tag{3.22}$$

where l represents the iteration number, $\alpha > 0$ is the learning rate, θ is the parameter vector and $E(\theta)$ is the loss function.

The gradient of the loss function, $\nabla E(\theta)$, is evaluated by using a subset of training set data, called *mini-batch*. Consequently, the result is just an estimation of the real gradient, that instead would be computed by the standard gradient descent algorithm employing all the training data available. Conversely, SGD takes a more noisy path towards the solution, thus avoiding being trapped in bad local minima and saddle points. Each evaluation of the gradient using mini-batch is an *iteration*, and, at each iteration, the algorithm takes one step towards minimizing the loss function. The full pass of algorithm over the entire training set across mini-batches is an *epoch*.

As regards our specific problem, we made the following choices in order to properly train our network:

- the initial learning rate has been set to 10^{-2} and reduced by a factor 10 every 2 epochs. In this way, at the beginning (first two epochs) the steps taken along gradient's direction are pretty large, so as small local minima are avoided. Once network's starts learning, we periodically reduce the size of these steps in order to properly converge to the solution.
- the mini batch size has been set to 30: this choice is driven by the fact that lower batch sizes led us to worse solutions in terms of accuracy, but at the same time higher values caused memory problems and longer training times.
- during network's training, a validation step is performed at the end of every epoch, in order to both understand if there are overfitting problems (which would cause a gap between training and validation error) and to stop the training if the validation loss does not decrease after a number of epochs that we set to 15. This is second stop criterion in addition to the one associated with the maximum number of epochs, which has been set to 100.

Once the network has been trained, the following step consists in *fine-tuning* its parameters so as to specialize it to the classification of images acquired with the prototype instrument, which are different from the ones of the Eidon data set, as explained in chapter 2. In this way, a fair comparison can be made between the performances achieved with this method and the ones relative to the heuristics-based and SVM algorithms, which instead are trained with images acquired with the prototype from the beginning.

The intermediate usage of Eidon data set with this CNN-based method has been necessary in order to properly train the network: as a matter of

fact, the prototype data set considered consists of just 181 images, and this is not a sufficient number to make a proper CNN training, because, even if the network has a simple architecture like the one we designed, its number of parameters is anyway huge. For this reason, we adopted such data set only for the *transfer learning* process, which allows to quickly transfer network's learned features to a new task by using a smaller number of training images.

In particular, we just updated fully connected layer's parameters by using the same training options previously reported, while keeping freezed all the remaining weights and biases associated with the previous convolutional layers: this is simply done by setting their learning rates to zero. In this way, our network employs the rich features representations learned during its actual training phase in order to extract information from the new images, while at the same time learning how to combine such information into class probabilities and labels by updating fully connected layer's parameters.

Chapter 4

Results

This chapter presents the classification results of the three image quality assessment algorithms developed and introduced in the previous sections. In particular, we will perform this analysis following the same order adopted for methods' exposure, and using the same statistical measures to describe the performances of all of them, so as to facilitate their comparison.

More specifically, we decided to report for each method the confusion matrix, which is the starting point to compute all the statistical measures we could be interested to, together with three specific indexes: sensitivity, specificity and accuracy. This choice is driven by the fact that they are the most widely used measures able to express in an immediate and effective way how much good and reliable our classifiers are. Before giving their actual formulation, it is important to clarify the terminology adopted for our specific setting:

- True Positives (TP): number of images which have been evaluated as of good quality by an human observer and correctly classified as such.
- True Negatives (TN): number of images which have been evaluated as of bad quality by an human observer and correctly classified as such.
- False Positives (FP): number of images which are evaluated as of bad quality by an human observer and wrongly classified in the class of good quality images.
- False Negatives (FN): number of images which are evaluated as of good quality by an human observer and wrongly classified in the class of bad quality images.

Finally, sensitivity, specificity and accuracy have been computed according to the following equations:

Sensitivity

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{4.1}$$

Sensitivity, also called True Positive Rate (TPR), measures the proportion of actual positives (good quality images) that are correctly identified as such. The higher the sensitivity value, the less likely the classifier will return FN results.

Specificity

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{4.2}$$

Specificity, also called True Negative Rate, measures the proportion of actual negatives (bad quality images) that are correctly identified as such. The higher the specificity value, the less likely the classifier will return FP results.

Accuracy

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

Accuracy measures the proportion of correctly classified images (both TP and TN) among our total population of images. It represents the ability of a classifier to correctly differentiate between good and poor quality cases.

It's worth mentioning that, even if neither sensitivity nor specificity are influenced by the *prevalence* of the good quality class over the bad quality one [18], we used a balanced data set for all the methods developed: in particular, as stated in chapter 2, we have at our disposal 181 images with central field, divided in 86 bad and 95 good by an expert human observer. These images represent a reduced version of the prototype data set, which contains also images with other fields, for a total of 322 images (177 good and 145 bad).

As regards the heuristics-based algorithm, the results introduced in table 4.1 have been obtained by computing all the descriptors defined in section 3.1 (except for the kurtosis one), for all images' ROI and considering only the value they assume in the green channel.

As a matter of fact, even if the quality evaluation framework described in section 3.2 extracts each feature for all images' color channels, with this simple algorithm we would not get any benefit in including such sophisticated information into the decision making process. Therefore, we chose to employ the features computed for the green channel because it shows the highest contrast with respect to the red and blue channel and it is the most invariant to age (and this is a factor we need to take into account considering the heterogeneous composition of our data sets).

		True class		
		Good	Bad	Total
Predicted class	Good	64	32	96
	Bad	31	54	85
	Total	95	86	181

Table 4.1: Confusion matrix for the heuristics-based algorithm

On the other hand, the results relative to the Support Vector Machine algorithm, introduced in table 4.2, are obtained by using the entire table of features generated by the quality evaluation framework, which includes all the descriptors computed for all images' ROI and color channels.

		True class		
		Good	Bad	Total
Predicted class	Good	74	14	88
	Bad	21	72	93
	Total	95	86	181

Table 4.2: Confusion matrix for SVM algorithm

As concerns the results relative to the CNN architecture developed, we obtained a 72,45% training and 72,32 % testing accuracy using the Eidon data set. However, in order to perform a fair comparison with the other two methods, we would have to employ the same kind of images used for both the heuristics-based and the SVM algorithms. For this reason, we fine tuned our network with transfer learning, so as to specialize it for the task of classifying images acquired with the prototype. In table 4.3 the confusion matrix relative to such fine tuned network is reported. In particular, the total number of images tested has been 181, in a cross validation setting with 2 folds.

		True class		
		Good	Bad	Total
Predicted class	Good	78	29	107
	Bad	17	57	74
	Total	95	86	181

Table 4.3: Confusion matrix for CNN architecture adopted

Finally, in table 4.4 the statistical measures of sensitivity, specificity and accuracy are reported for each of the three methods developed, so as to facilitate their comparison.

Method	**Sensitivity**	**Specificity**	**Accuracy**
Heuristics	0.6737	0.6279	0.6519
SVM	0.7789	0.8372	0.8066
Deep Learning	0.8210	0.6628	0.7458

Table 4.4: Methods performances comparison

Chapter 5

Discussions

Although literature provides many works related to the topic of automatic retinal image quality assessment [20, 21, 22], no method or approach can be considered established because better than the others, given the broad range of different applications and aims associated with this research field. For this reason, we considered appropriate not to discard *a priori* any kind of solution: we started with the development of one simple algorithm based on heuristics, whose details can be found in section 3.3.

This method presents several advantages, first and foremost its simplicity, speed and interpretability: as a matter of fact, it does not fit in the field of machine learning supervised techniques, and as such it does not require any sort of initial training, which can take a significant amount of time. Moreover, its architecture has been optimized so as the minimum number of comparisons is required in order to perform the classification of an image. As regards its interpretability, all the threshold values chosen are supported by strong and solid arguments based on the descriptors defined and validated in 3.1, and so it is easy to understand why an image is classified into a certain category. This also implies that if we are not satisfied with the classification results, either because the algorithm tends to be too sensitive or too specific, it is possible to tune one or more of its parameters to accordingly change its behaviour as desired, since all the choices made are arbitrary and based on subjective evaluations. This point in particular strongly differentiate the heuristics-based algorithm from all the supervised learning techniques further developed, whose decisions are based on more complex models and as such are not directly and easily manageable.

However, as any other technique, this method has also some drawbacks arising from its intrinsic simplicity, which was the characteristic that made us consider this algorithm as a good candidate in the first place. As a matter of fact, finding reasonable and solid threshold values for all the descriptors considered turned out being troublesome, especially for those based on second order statistics, which express an information not directly visible into

the images. In order to efficiently embed such an information, we would have to employ more complex and sophisticated techniques, which would also allow more versatility when dealing with pathological images. This is in fact another critical point of this heuristics-based algorithm, despite the choices made during its designing phase so as to make it less strict and suitable for a broad range of heterogeneous images. Furthermore, in the light of the results achieved by applying this algorithm to our specific data set, we decided it was worth moving towards the development of a more elaborate approach.

The Support Vector Machine algorithm finds its place in this framework, and it actually represents our preferred choice among the three methods developed, since it is complex enough to capture and explain the variability present in our data set, while keeping the computational burden totally affordable. Moreover, the number of hyper parameters we need to set is limited to the only values of C and γ, for whose choice we adopted the well-established technique of telescopic grid search, which ensures an exhaustive search is performed. Additionally, the results obtained with this solution are quite satisfactory, considering also that our data set includes several images of borderline quality, which are an hard challenge even for an expert human observer. Besides, as we can see from table 4.4, the algorithm is very specific, so we can consider it as a trustworthy detector of good quality, which is one of the goals we wanted to achieve.

Lastly, the CNN architecture developed in this thesis work represents the most complex solution adopted, that we considered in the first place for the remarkable performances demonstrated and reported in literature. However, in order to make possible its development, we had to make since the beginning some specific and constrained choices, which certainly affected the goodness of the results obtained in the end.

First of all, the number of data needed to train such a network is huge, and our Eidon data set consists in a few thousands images, as pointed out in chapter 2: this has limited our flexibility during network's designing phase. As a matter of fact, if we build a more complex architecture than the one adopted and explained in section 3.5.2, the number of parameters that the network would have to learn would be too high compared to the number of data at our disposal. This restriction led us also to the necessity of cropping the original images given in input to the network, so as to reduce them to a size that it could manage. However, even if the choice of cropping the images rather than downsampling them allows us to keep the focus information and their original resolution, it is an undeniable drawback, whose price is paid in terms of results: as a matter of fact, as we can see from table 4.4, the accuracy achieved by this classifier after the transfer leaning step is 75%, which is quite low is we think to the potentiality of this deep learning-based methodology and to its complexity. Moreover, the results show also

that the algorithm tends to be more sensitive rather than specific, so the number of false positive results is higher than the false negative ones: this characteristic is opposite to the one we are seeking in a classifier, because for our application we believe it would be more cost effective reacquire an image wrongly rejected by our classifier rather than mistakenly accept and thus classify a bad quality image as of good quality. Ultimately, the computational power required and the time needed to train the network have turned out being significant even with such a little architecture, so it has been hard finding a successful trade off between these practical constraints and the necessity of obtaining an accurate solution by ensuring a proper convergence of the minimization algorithm.

These limitations, together with the general lack of interpretability of this approach, made us incline towards a more simpler solution like the SVM one, whose embedding into a screening-oriented instrument would be easier and preferable, especially in the view of a real time usage.

Conclusions

The main objective of this thesis has been addressing the problem of retinal image quality assessment by adopting solutions which rely on methodologies very different from each other, in order to compare them and determine which one should be preferred in the view of an embedding in a highly automated screening system.

In particular, we started our analysis with the heuristics-based algorithm, which represents the simplest solution developed in this work, moving then to the field of supervised machine learning techniques so as to investigate the more sophisticated approaches based on SVM and CNN.

The results obtained clearly show the superiority of SVM on our data set in comparison with the other methods, both in terms of performances achieved and computational power required. As a matter of fact, the heuristics based algorithm turned out being too simple to explain the variability of our data and thus properly address such a complex problem, regardless its advantages in terms of interpretability and quick computation. The strong points of this heuristics based algorithm are instead the shortcomings of the deep learning approach, which hence represents a quite opposite solution to the problem: nevertheless, even the CNN architecture adopted does not provide the results expected. In particular, with SVM the accuracy gained is 81%, in contrast with the 75% and 65% respectively obtained with the deep learning approach and the heuristics based algorithm. In the light of these results and arguments, SVM represents our final choice, even if further refinements are required in order to make this solution even more reliable and accurate. In particular, this improvement will be achieved when more data will be available, so as the model could be trained on a more representative sample of retinal images population.

As well as regards the deep learning approach, a potential direction for a future work oriented towards an improvement of the results obtained certainly relies on the need of more data. As a matter of fact, a more large data set would allow the network a much complete and effective training than the one performed in this work, together with more flexibility in terms of designing choices. Either way, more computational power than the one we have now at our disposal would be required in order to handle an eventual increased complexity of network's architecture and much longer and inten-

sive training. Furthermore, as suggested by [19], an immediate enhancement could be obtained refining the pre processing step of this algorithm, so as to give in input to the network not some random patches extracted from a cropped version of our images, but patches centered on specific image areas containing for example anatomical landmarks, that must remain visible for quality assessment purposes. In this way, a more precise and structured information could be given to the network, which will be trained on an automatically more numerous data set. In the end, these patches would be combined during algorithm's testing phase so as to produce an unique classification for the image to which they are associated.

Once adopted these solutions and gained better performances on a much larger data set, it could be interesting developing a fourth approach, based on a combination of deep learning and SVM, so as to complement their strength and thus obtain an highly accurate solution for our problem. In particular, we could employ the rich features representations learned by the CNN architecture on a large and representative data set in order to train the SVM model: in this way, we could also have a better insight of the descriptive power of our handcrafted features in comparison with the ones deriving from a completely black box approach as the deep learning one.

Bibliography

[1] Yu H., Agurto C. et al *Automated image quality evaluation of retinal fundus photographs in diabetic retinopathy screening.* SSIAI, 2012

[2] Davis et al. *Vision-based, real-time retinal image quality assessment.* 22nd IEEE International Symposium on Computer-Based Medical Systems, 2009

[3] Cassin, B. & Solomon, S. *Dictionary of Eye Terminology.* Gainsville, Florida: Triad Publishing Company, 1990

[4] Jonas, J.B., Schneider, U. & Naumann *Count and density of human retinal photoreceptors.* Archive for Clinical and Experimental Ophthalmology 230: 505-510

[5] Costa P. et al. *EyeQual: Accurate, Explainable, Retinal Image Quality Assessment.* 16th IEEE International Conference on Machine Learning and Applications, 2017

[6] Paulus J. et al. *Automated quality assessment of retinal fundus photos.* International Journal of Computer Assisted Radiology and Surgery, vol. 5, pp. 557-564, 2010

[7] Opt. Department of F.P.R.C. Visual Sciences of the University of Wisconsin-Madison. *ARIC Grading Protocol*, 1995

[8] Dias P., Oliveira M. *Retinal image quality assessment using generic image quality indicators.* Elsevier, 2012

[9] Pires Dias et al. *Retinal image quality assessment using generic image quality indicators.* Elsevier, 2014

[10] Gonzalez RC, Woods RE *Digital Image Processing.* Addison Wesley, US, 1992

[11] Haralick R.M. et al. *Texture Features for Image Classification.* IEEE Transactions on Systems, Man and Cybernetics, vol. 3, pp. 610-621, 1973

[12] Gaurav Kumar et al. *A detailed review of Feature Extraction in Image Processing Systems.* Fourth International Conference on Advanced Computing and Communication Technologies, 2014

[13] Scholkopf B. et al. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond.* MIT Press Cambridge, MA, USA, 2001

[14] Parampal S. Grewal et al. *Deep learning in ophthalmology: a review* Canadian Journal of Ophthalmology, 2018

[15] Saha et al. *Automated Quality Assessment of Color Fundus Images for Diabetic Retinopathy Screening in Telemedicine.* Journal of Digital Imaging

[16] Bottou L., *Stochastic Gradient Descent Triks.* Springer, Berlin, Heidelberg, 2012

[17] Gardner W.A. *Learning characteristics of stochastic gradient-descent algorithms: A general study, analysis and critique* Elsevier Signal Processing, vol. 6, pp. 113-133, 1984

[18] Parikh et al. *Understanding and using sensitivity, specificity and predictive values.* Indian Journal of Ophthalmology

[19] Mahapatra D. et al, *CNN based neurobiology inspired approach for retinal image quality assessment.* Conf Proc IEEE Eng Med Biol Soc. 2016

[20] Lee SC., Wang Y. *Automatic retinal image quality assessment and enhancement.* Proceedings of SPIE Medical Imaging Processing, 3661:1581-1590. SPIE (Washington, DC 1999)

[21] Lalonde M., Gagnon L. et al *Automatic visual quality assessment in optical fundus images.* Proceedings of Vision Interface 2001, Ottawa, 259-264

[22] Fleming AD et al. *Automated Assessment of Diabetic Retinal Image Quality Based on Clarity and Field Definition.* Investigative Ophthalmology and Visual Science, vol. 47, pp. 1120-1125, 2006