Università degli Studi di Milano Bicocca

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di laurea in Informatica

HUMAN SKIN DETECTION IN COLOR IMAGES

Relatore: Prof. Raimondo Schettini

Co-relatore: Dr. Mirko Agarla

Relazione della prova finale di:

Michele Pozzi

Matricola 845727

Anno Accademico 2020-2021

Abstract

Skin detection is the process of discriminating skin and non-skin pixels in an arbitrary image and represents an intermediate step in several image processing tasks, such as facial analysis and biomedical segmentation. Different approaches have been presented in the literature, but a comparison is difficult to perform due to multiple datasets and varying performance measurements. In this work, the datasets and the state-of-the-art approaches are reviewed and categorized using a new proposed taxonomy. Three different representative skin detector methods of the state of the art are selected and thoroughly analyzed. This approaches are then evaluated on three different state of the art datasets and skin tones sub-datasets using multiple metrics. The evaluation is performed on single and cross dataset scenario to highlight key differences between methods, reporting also the inference time. Finally, the results are organized into multiple tables, using the related figures as an assistance tool to support the discussion. Experimental results demonstrate the strength and weaknesses of each approach, and the need to involve multiple metrics for a fair assessment of the method's aspects.

Acknowledgments

The journey of completing my Bachelor's degree and writing this thesis has been demanding in many respects.
Nevertheless, it has been one of the most enriching experiences I've ever had and now that it comes to an end, I want to thank the people who supported me and helped me through it.

I would like to thank Prof. Raimondo Schettini for giving me the opportunity of working on this project.
This thesis has allowed me to explore fields that otherwise would have been difficult to encounter and that have fascinated me greatly.

I would like to extend my deepest gratitude to Dr. Mirko Agarla whose invaluable assistance and patience helped me in every step of the way.
I must also thank him for the much needed critical advice on the writing of this thesis.

I am grateful to my family for supporting me throughout the entire duration of my studies.

Contents

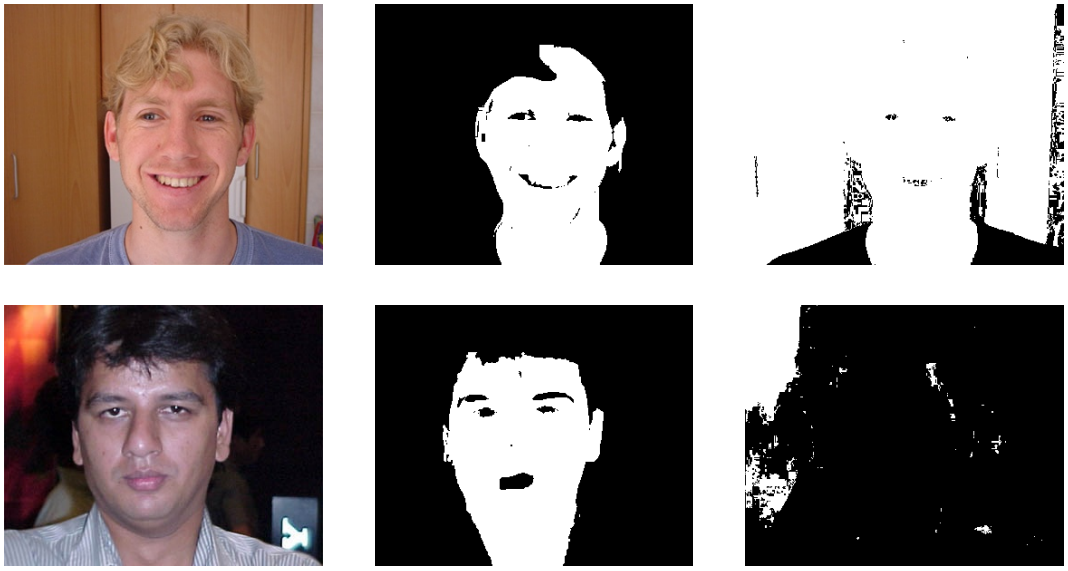
1 Introduction

The presence of human skin in media gives meaningful information. Skin features are important cues that can be used to infer a variety of aspects related to a person, such as attractiveness, age, and health [1, 2]. Skin detection is the process of discriminating skin and non-skin pixels in an arbitrary image or video, as illustrated in Figure 1.1. It is primarily used as an intermediate step for more complex tasks such as facial analysis [3, 4], gesture analysis [5], video surveillance [6], privacy protection [7], adult image detection [8, 9, 10], region-of-interest detection [11], and advertisements [12]. In the biomedical field, skin detection plays an important role in lesions segmentation and cutaneous diseases classification [13, 14]. Detecting skin-colored pixels has proven quite challenging for various reasons. Materials with a similar color to the skin, such as wood, copper, leather, and sand, can be incorrectly labeled as skin pixels. Moreover, the appearance of skin in an image depends on many variables, such as the illumination conditions, the camera color science, and motion blur [15]. Finally, skin tones vary dramatically within and across individuals. Some challenging aspects are depicted in Figure 1.2.

(a) (b)

Figure 1.1: Skin detection: (a) original image; (b) detected skin pixels. The original image is part of the Pratheepan dataset [16].

This thesis presents a review of the skin detection datasets and state-of-the-art approaches. Image databases and state-of-the-art approaches are retrieved, and a new taxonomy is proposed. Three different computational approaches are selected, thoroughly



(a) (b) (c)

Figure 1.2: Challenges of skin detection, visualized through: (a) the input image, (b) the ground truth, (c) the prediction obtained from a thresholding method [17]. The first row depicts an image containing materials with skin-like colors. The second row shows how the colors of an image can be affected by the illumination conditions.

analyzed, and implemented to compare their strength, weaknesses, generalization capability, and performance. Before assessing the methods, a validation phase takes place; then the methods are evaluated in single and cross dataset settings over three datasets and three skin tones sub-datasets, using multiple metrics. Finally, inference times are compared.

The main contributions of this work are the following:

- A comparative review of common datasets used in skin detection and discussion on some major data limitations.
- An analysis of the state-of-the-art approaches at skin detection, with the proposal for a new taxonomy.
- An analysis of some of the evaluation metrics used in binary classification problems.
- An in-depth analysis of the three representative state-of-the-art approaches.
- An evaluation of the selected approaches on three public datasets in single-dataset and cross-dataset settings.

- An evaluation of the selected approaches on three skin tones sub-datasets in single-dataset and cross-dataset settings.
- An evaluation of the inferences times of the selected approaches.

The thesis is organized as follows: the Chapter 2 gives an overview of the datasets used in skin detection including a discussion on the limitations and a detailed description of the three chosen datasets. Chapter 3 contains an in-depth analysis of the selected approaches, describing the architecture, functioning, and operations performed by each method. Chapter 4 contains a detailed explanation of the experiments setup and the evaluation process. Then, the evaluation results are presented and extensively commented. Finally, Chapter 5 draws conclusions and gives an outlook on possible future work.

1.1 State of the art

From a classification point of view, skin-color detection can be seen as a two-class problem: skin-pixel vs. non-skin-pixel classification. The problem has been approached in different ways, which aren't uniquely categorizable. Researchers have categorized the methods mainly in two groups: segmentation-based [18] and classification-based [17]. With the advent of deep learning, the former categorization becomes more ambiguous. In fact, Machine Learning methods include both approaches based on the analysis of individual pixels, such as traditional methods, and approaches that use the pixel neighbors information, such as CNNs.
A new classification-based taxonomy based is proposed, Figure 1.3. It categorizes skin detector methods into the following groups: rule-based, machine learning and hybrid.
Rule-based methods use plain rules to classify each pixel as either skin or non-skin. Thresholding and fuzzy logic systems are part of this group.
Machine learning approaches construct models from a training set of data to use in classification. They describe a broad range of techniques, which can be subdivided into statistical, deep learning and ensemble categories.
Hybrid approaches make use of a cluster of different techniques working together to perform the classification. A common choice is to stack a region-based algorithm on top of the result of other classification methodologies.

This section describes some common skin detection approaches. A chronological organization of the review with regards to the proposed taxonomy is reported in Table 1.1.
Jones and Rehg 2002 [8] is a machine learning statistical method based on the construction of a skin-color histogram model from an image database. Then, a Bayesian classifier uses the model to make predictions on the desired images. The histogram model is built from an image database by analyzing the frequency that every pixel (R,G,B) combination has to be either skin or non-skin. The Bayes rule utilizes a threshold to classify pixels

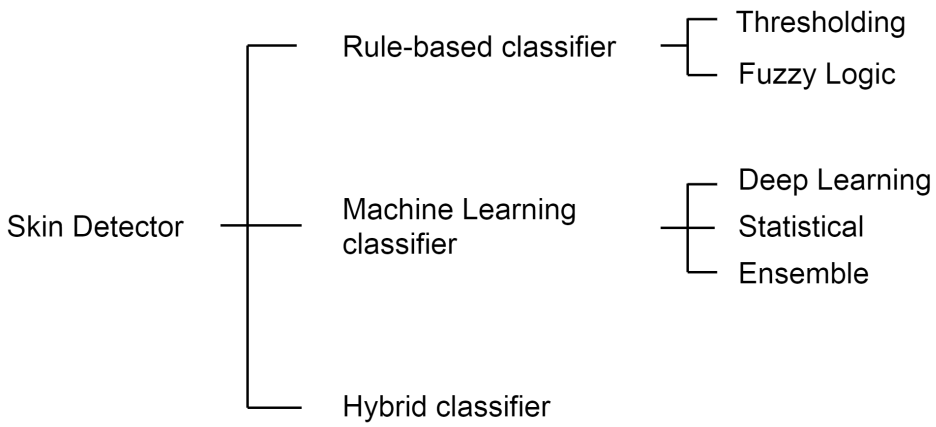


Figure 1.3: Taxonomy of skin detectors.

based on their probability of being skin pixels.

Kovac *et al.* 2003 [19] is a rule-based thresholding method that uses the union of a pair of rules to segment skin regions. The first rule specializes in finding skin at uniform daylight illumination. The second rule specializes in finding skin under flashlight or lateral daylight illumination.

Chen and Wang 2007 [20] is a hybrid approach consisting of four main stages: image segmentation, key skin region extraction, similarity measurement, and skin region classification. In the first stage, an unsupervised segmentation of color-texture regions in images is applied to segment the image into homogeneous regions. In the second stage, a skin extractor is used for extracting the key skin regions. Firstly, the candidates of key skin regions are chosen by a rule-based skin classifier. Then, the confidence values of the candidate regions are calculated as the proportion of the detected skin pixel count to the total pixel count of the region. The region with the largest confidence value will be the key skin region. In the third stage, the similarity measure between the key skin region and all other regions is calculated to merge more possibly skin regions. In the last stage, an algorithm is used to determine whether a region should be classified as a skin region or a non-skin region. The algorithm calculates the pixels overlap ratio of the two 2D skin-color histograms representing a region and the key skin region, and then utilizes a threshold to classify the region.

Khan *et al.* 2010 [21] is a machine learning ensemble approach that utilizes a random forest, an ensemble of tree predictors, to classify skin regions. The input is presented to each of the trees in the forest. Each tree gives an independent classification prediction and "votes" for that class. The forest chooses the classification having the most votes. An image database is used to train the forest trees.

Iraji and Yavari 2011 [22] is a rule-based fuzzy logic approach that defines a set of rules in the YCbCr color space to distinguish skin and non-skin pixels. There may be pixels

whose color acts both as skin color and as non-skin color. Thanks to fuzzy logic with a Mamdani inference system (based on the implication rules), it is possible to get a unique membership value for each pixel, making the classification straightforward.

Kawulok *et al.* 2014 [23] is a hybrid method that combines spatial analysis on top of a machine learning classifier. First, the input image is converted into a skin probability map using a statistical histogram skin color model and a Bayesian classifier. Then, the probability map is processed to find seed pixels. The initial seed pixels are expanded using distance transform to include more skin pixels. Subsequently, a local skin color model is trained from the obtained result and used to detect further skin pixels. Finally, the distance transform is applied one more time.

Brancati *et al.* 2017 [17] is a rule-based thresholding approach that works in the YCbCr color space. Dynamic correlation rules are used to evaluate the combinations of chrominance values to identify the skin pixels in the YCb and YCr subspaces. The correlation rules depend on the shape and size of dynamically generated skin color clusters computed in the YCb and YCr subspaces.

He *et al.* 2019 [24] is a deep learning method based on a dual-task CNN for joint detection of skin and body. The dual-task network contains a shared encoder but two decoders for skin and body separately. For each decoder, its output also serves as a guide for its counterpart, making both decoders mutually guided.

Tarasiewicz *et al.* 2020 [25] proposed Skinny, a deep learning approach based on an encoder-decoder CNN architecture called U-Net. Inception and dense block are inserted into a modified U-Net architecture to benefit from a wider spatial context.

Name	Year	Category	Subcategory
Tarasiewicz *et al.* [25]	2020	Machine Learning	Deep Learning: CNN
He *et al.* [24]	2019	Machine Learning	Deep Learning: CNN
Brancati *et al.* [17]	2017	Rule-based	Thresholding
Kawulok *et al.* [23]	2014	Hybrid	Machine Learning + Spatial Analysis
Iraji and Yavari [22]	2011	Rule-based	Fuzzy Logic
Khan *et al.* [21]	2010	Machine Learning	Ensemble: RandomForest
Chen and Wang [20]	2007	Hybrid	Rule-based + Region-growing
Kovac *et al.* [19]	2003	Rule-based	Thresholding
Jones and Rehg [8]	2002	Machine Learning	Statistical: Bayes

Table 1.1: Common approaches to Skin Detection, sorted by year in descending order.

2 Datasets

The importance of databases can be demonstrated by looking at the challenges in the field of Artificial Intelligence. For years, the research on Artificial Intelligence has focused on the same concept: a better algorithm would make better decisions, regardless of the data. However, even the best algorithm would not work well if the learned data didn't reflect the real world.
A good dataset is one that fits the desired purpose. However, there are some general considerations that indicate good quality in a dataset. Completeness, balanced classes, good organization, and quality labeling define a dataset of high quality.

In skin detection, a dataset usually consists of two types of pictures: the original images and the labeled images. Labeled images are the ground truths and can either be binary masks or segmentation masks, where only the skin pixels are present. Some datasets focus on the skin of a specific body part, such as the face or the abdomen.

2.1 Datasets Overview

A summary of common skin detection datasets is presented in Table 2.1, starting from the oldest to the newest. Only public datasets featuring images and including ground truths are considered.
TDSD [10] is the acronym of Test Database for Skin Detection, which is a database featuring 555 full-body skin images. Its ground truths are segmentation masks. It is also referred to as IBTD.
ECU [26] is a dataset created at the Edith Cowan University and represents the largest analyzed dataset, consisting of 3998 pictures. It has been categorized as a full-body dataset, but most of its content is half-body shots. It can also be referred to as Face and Skin Detection Database (FSD).
Schmugge [27] is a facial dataset that includes 845 images taken from different databases. It provides several labeled information about each image and ternary ground truths.
Pratheepan [16] is composed of 78 pictures randomly sampled from the web, precisely annotated[28]. It stores the pictures containing a single subject with simple backgrounds and images containing multiple subjects with complex backgrounds in different folders.
VPU [29], as for Video Processing & Understanding Lab, consists of 285 images taken from five different public datasets for human activity recognition. The size of the pictures is constant between the images of the same origin. The dataset provides native train

and test splits. It can also be referred to as VDM.
SFA [30] is the acronym of Skin of FERET and AR Database and consists of 1118 semi-passport pictures with a very plain background, and skin and non-skin samples (ignored in this work). Its ground truths are segmentation masks.
HGR [23] is a Hand Gesture Recognition Database that organizes 1558 hand gesture images in three sub-datasets. Two sub-datasets include size-fixed very high-resolution images together with downscaled alternatives (used in this work).
abd-skin [31] is a database composed of 1400 size-fixed abdominal pictures accurately selected to represent different ethnic groups and body mass indices. It has native test and train splits.

Name	Year	No. of Images	Shot Type	Skin Tones[1]
abd-skin [31]	2019	1400	abdomen	african, indian, hispanic, caucasian, asian
HGR [23]	2014	1558	hand	-
SFA [30]	2013	1118	face	asian, caucasian, negro
VPU [29]	2013	285	full body	-
Pratheepan [16]	2012	78	full body	-
Schmugge [27]	2007	845	face	skintones labels: light, medium dark
ECU [26]	2005	3998	full body	whitish, brownish, yellowish, and darkish
TDSD [10]	2004	555	full body	different ethnic groups

Table 2.1: Common Datasets used in Skin Detection

[1]The "Skin tones" column reports either the ethnic diversity cited in the corresponding papers or the labels utilized for the skin tone values, in case they are present.

A few notable public datasets are missing from the analysis for the following reasons.
Compaq [8] is the first large skin dataset consisting of 4675 labeled pictures. Ground truths are annotated with a semi-automatic process, hence the accuracy is not high [32]. Moreover, it is no longer available for public use [33]. Despite the mentioned limitations, it is still used in the skin detection domain [17, 34].
LVS [8] is an image database containing 2118 labeled frames. Labeled Video Sentences (LVS) contains original images and ground truths of skin and non-skin regions. The ground truth masks do not include ambiguous skin regions. The ground truths are quite imprecise because features like eyes and mouth are counted as skin pixels, as seen in Figure 2.1.
Some databases have been discarded because they do not use images: **Feeval** [9] uses video sequences; **UCI** [35] contains a list of skin and non-skin pixels in the BGR color space.

(a) (b) (c)

Figure 2.1: LVS [8] dataset example: (a) original image, (b) skin ground truth, (c) non-skin ground truth. Facial features like the eyes and mouth are treated as skin.

2.2 Issues

Image databases are essential for developing skin detectors. Over the years, new databases keep getting published, but there are still some limitations on their reliability.
The major limitations are described below, and some of them are illustrated in Figure 2.2.

- The number of pictures is sometimes not sufficiently high.
- The image quality is at times very low, and a sufficient intra-class variation is not present.
- The classes are often unbalanced, with one being much larger than the other, which may cause some metrics to give overoptimistic estimations [36].
- The ground truth labeling sometimes is performed using semi-automatic techniques, which give imprecise results [32]. Moreover, some skin regions are of dubious classification, especially the boundaries of the skin and around features such as eyes and mouth.
- Additional data is often cited in the original papers of the datasets, but not provided alongside it. Data about the lighting conditions, background complexity, number of subjects, indoor or outdoor scenery of an image may be extremely useful in some applications.
- Compression artifacts generated during the storage of ground truth masks represent another issue. The artifacts are inconvenient for either binary and segmentation masks and should be avoided by using lossless formats.
- Different skin tones are rarely evenly represented, and especially are not directly labeled. Furthermore, the categorization of skin tones does not follow a standard system and is questionable.
- Most image databases do not provide native training and testing splits, which confuses the evaluations in the literature.

(a) (b) (c)

Figure 2.2: TDSD [10] bad annotation examples: (a) original image No. 0487; (b) skin segmentation ground truth No. 0487; (c) compression artifacts. Test Database for Skin Detection (TDSD) contains both good and bad annotation examples. Ruiz-del-Solar and Verschae proposed three sub-dataset splits based on the annotation quality [33]. Despite the issues, the dataset is important because, being born as a test dataset, it contains more uncorrelated images than those available in video datasets [37].

2.3 Chosen Datasets

Accordingly to the issues previously mentioned, and considering the popularity, diversity, and size of the databases, the selected datasets for this work are ECU [26], HGR [23] and Schmugge [27]. A more detailed description of each one follows.

ECU [26] is a dataset created at the Edith Cowan University that aims to support research on skin segmentation and face detection. It contains 3998 pictures differing in terms of the depicted skin regions, skin tone types, lighting, and background, with both indoor and outdoor environments. The authors did a rough categorization of the skin tones: 1665 images represent whitish and pinkish skin types; 1402 images represent yellowish and light brownish skin types; 965 images characterize reddish, darkish, and dark brownish skin types; the remaining ones generically represent other skin types.

HGR [23] is a Hand Gesture Recognition Database of 1558 images. It organizes into three different sub-datasets: HGR1, HGR2A, and HGR2B. HGR1 is a set of 899 pictures of various sizes and taken in uncontrolled light and background environments. HGR2A contains 85 images of the same dimension taken in uniform lighting and both controlled and uncontrolled backgrounds. HGR2B features 574 constant-sized pictures taken with a controlled background and in uniform lighting. The skin tone diversity is very low as the images represent only a limited number of subjects. Regarding the HGR2A and HGR2B sub-datasets, this work utilizes the downscaled versions of the pic-

tures because high-resolution images would automatically be downscaled in approaches like neural networks.

Schmugge [27] takes the name from its creator and consists of 845 images taken from different sources: the pictures representing skin pixels are collected from the AR face dataset [38], the UOPB dataset [39], and the University of Chile dataset [33]. The first two include frontal facial images with varying illumination conditions and a very plain white background. Therefore, for these images, the non-skin pixels are not included. The Chile dataset is composed of websites and digitized news clips and represents a variety of scenes, thus its non-skin pixels are included. Other non-skin pixels are collected by randomly sampling the University of Washington content-based image retrieval database [40]. Unlike other datasets, the ground truth follows a ternary representation: pixels are labeled as skin, non-skin, and "don't care". The "don't care" label is assigned to pixels that are too ambiguous or tedious to label as either skin or non-skin. This ternary division permits better management of pixels classification. The dataset provides different files featuring additional data for each image, such as the skin tone, the light type, and the original database the picture is from. The ternary annotation system used in the dataset is shown in Figure 2.3.

Figure 2.3: Schmugge [27] samples, RGB image on the left column and groundtruth on the right column. In annotations, skin pixels are black while non-skin pixels are white. Difficult and tedious regions to mark ("don't care") are colored in gray. The background in the top two images is marked in gray, indicating that those pixels did not participate in the evaluation. Adapted from Schmugge *et al.* 2007 [27]

3 Investigated Methods

As seen in 1.1, there are several approaches to skin detection, and it can be confusing to make a choice. All the different categories of methods are characterized by certain strength and weaknesses, therefore the choice to implement an approach should be made carefully.
Rule-based techniques represent a simple method to rapidly separate objects from their surroundings, don't require training, and generally are easy to implement. However, color is the only feature they consider, and therefore the classification on backgrounds with skin-like colors, such as wood or clay, could be difficult.
Machine learning approaches are suitable when there is training data available. There are several techniques with different features in this category. Some uses only pixels color has the main drive for classification, while others manage to consider multiple features.
Hybrid methods are suitable when a single classification technique does not produce the desired results.

The chosen methods belong to the following categories: thresholding, traditional machine learning, deep learning.
Thresholding has been chosen because of the simplicity of the approach, which may demonstrate how powerful simple rules can be.
Machine learning and deep learning have been chosen for a comparison of their classification ability: it might be interesting to see how CNNs can learn about semantics from images [41] and the comparison with respect to the traditional models.

3.1 Thresholding

Thresholding methods are based on the idea that human skin can be grouped in clusters within a color space. Therefore, the main process is to define the boundaries of the clusters. Color images are segmented by designating separate thresholds for each color component, as seen in Figure 3.1. The pixels falling within the range of these thresholds are classified as skin pixels.

There are static and dynamic thresholding methods. **Static thresholding** consists of simple fixed rules to define the cluster boundaries. In **Dynamic thresholding** the rules defining the boundaries depend on some variables.

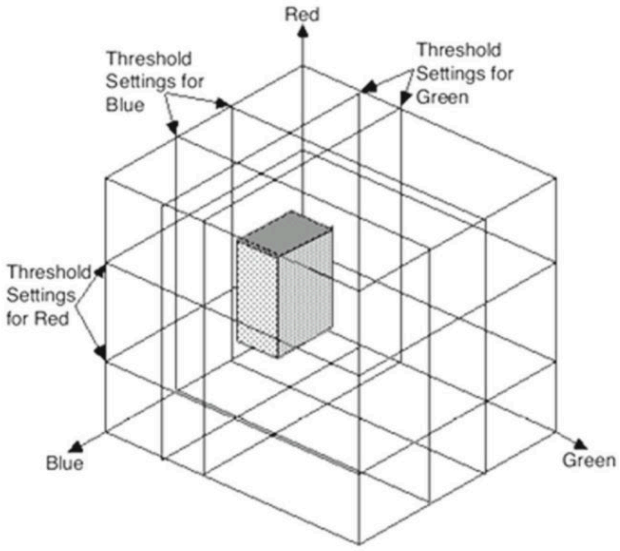


Figure 3.1: Separate threshold settings for each color component. The shaded area is the Boolean AND of the three threshold settings. Adapted from John C. Russ 2007 [42]

In literature, there isn't an agreement on the best color space to use. Nonetheless, to select the most suitable color space, its features should be taken into account. For example, RGB describes a high correlation between color channels. An interesting technique to increase the detector accuracy is to combine channels of different color spaces [43]. Several color spaces have been tested over the years: RGB [19, 44, 6], RGB-H-CbCr [43], HSV [45, 4, 46], YUV [47, 48], YCbCr [45, 48, 17]. Despite their simplicity, thresholding methods are relevant because of their low computational cost. Their efficiency makes hardware implementations suitable, for example on Field Programmable Gate Array (FPGA) [6]. An example of threshold rules in the RGB color space is presented below (taken from Kovac *et al*. 2003 [19]):

```
# The skin colour at uniform daylight illumination
R > 95 AND G > 40 AND B > 20 AND
max{R, G, B} - min{R, G, B} > 15 AND
|R - G| > 15 AND R>G AND R>B

OR

# The skin colour under flashlight or (light) daylight
# lateral illumination
R > 220 AND G > 210 AND B > 170 AND
|R - G| <= 15 AND R>B AND G>B
```

3.1.1 Implementation

The chosen implementation[1] is a dynamic thresholding approach [17] published in 2017. An overview of the algorithm is presented below:

Step 1: RGB to YCbCr conversion

Step 2: Computation of Cr_{max} and Cb_{min}

Step 3: Pixel-wise computation of correlation rules parameters

Step 4: Pixel-wise correlation rules check

The method works in the YCbCr color space, so the first thing it does is the conversion of an RGB input image by using the ITU-R BT.601-5 conversion [49]. The skin pixels clusters assume a trapezoidal shape in the YCb and YCr color subspaces, as seen in Figure 3.2. Moreover, the shape and size of the trapezium vary according to many factors, such as the illumination conditions.

In high illumination conditions, the base of the trapezium results larger. Besides, the chrominance components of a skin pixel P with coordinates (P_Y, P_{Cb}, P_{Cr}) in the YCbCr space exhibit the following behavior: the further is the (P_Y, P_{Cr}) point from the longer base of the trapezium in the YCr subspace, the further is the (P_Y, P_{Cb}) point from the longer base of the trapezium in the YCb subspace, and vice versa.

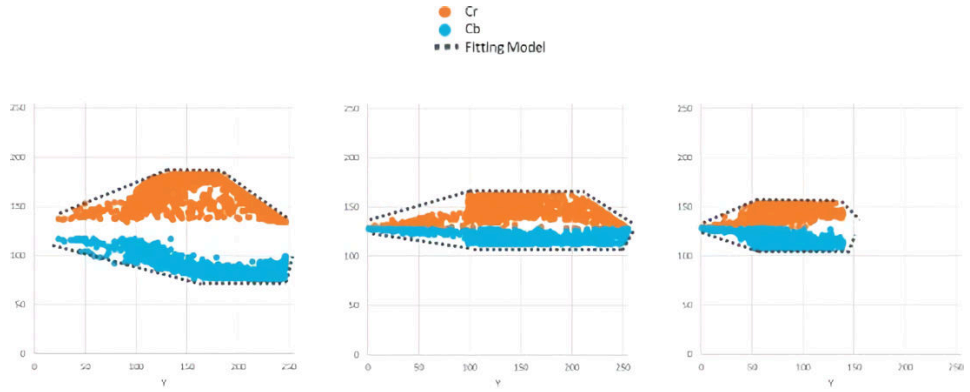


Figure 3.2: Fitting model for the YCb and YCr subspaces, in different conditions of illumination: (a) indoors with artificial light (b) outdoors with sunlight and (c) with low lighting. Adapted from Brancati *et al.* 2017 [17].

[1]Source code available at https://github.com/nadiabrancati/skin_detection/

The aforementioned observations are the base of the method: it tries to define image-specific trapeziums in the YCb and YCr color subspaces and then verifies that the correlation rules between the two subspaces reflect the inversely proportional behavior of the chrominance components.

The first step is the computation of the height of the trapeziums in the YCb and YCr subspaces (the trapeziums are referred to as T_{YCb} and T_{YCr} in the following), which will be useful later on. Referring to Figure 3.3 for a visual representation, it can be noticed that by varying the Y value from its minimum value 0, to its maximum value 255, the coordinates of points belonging to the longer bases of T_{YCr} and T_{YCb} are given by (Y,Cr_{min}) and (Y,Cb_{max}) in the YCr and YCb subspaces. The coordinates of points belonging to the remaining sides of the trapezium are given by [Y,$H_{Cr}(Y)$] and [Y, $H_{Cb}(Y)$] with:

$$
\begin{aligned}
H_{Cr}(Y) &= \begin{cases} Cr_{\min} + h_{Cr}\left(\frac{Y-Y_{\min}}{Y_0-Y_{\min}}\right) & Y \in [Y_{\min}, Y_0] \\ Cr_{\max} & Y \in [Y_0, Y_1] \\ Cr_{\min} + h_{Cr}\left(\frac{Y-Y_{\max}}{Y_1-Y_{\max}}\right) & Y \in [Y_1, Y_{\max}] \end{cases} \\
H_{Cb}(Y) &= \begin{cases} Cb_{\min} + h_{Cb}\left(\frac{Y-Y_2}{Y_{\min}-Y_2}\right) & Y \in [Y_{\min}, Y_2] \\ Cb_{\min} & Y \in [Y_2, Y_3] \\ Cb_{\min} + h_{Cb}\left(\frac{Y-Y_3}{Y_{\max}-Y_3}\right) & Y \in [Y_3, Y_{\max}] \end{cases}
\end{aligned}
\tag{3.1}
$$

where $h_{Cr} = Cr_{max} - Cr_{min}$ and $h_{Cb} = Cb_{max} - Cb_{min}$ represent the heights of T_{YCr} and T_{YCb}, respectively.

The Cr_{max} and Cb_{min} values are computed by taking into account the histogram of the pixels with the following values: $Cr \in [Cr_{min}, 183]$ and $Cb \in [77, Cb_{max}]$, looking for the maximum of Cr and the minimum of Cb that are associated with at least 10% of the pixels in the image. The Y_0 and Y_1 values are respectively set as the 5th percentile and the 95th percentile of the luminance values associated with the pixels of the image with $Cr = Cr_{max}$. The same procedure is followed to find the Y_2 and Y_3 values, considering the pixels with $Cb = Cb_{min}$. This process is illustrated in Figure 3.4.

The correlation rules between the chromatic components P_{Cr} and P_{Cb} of a pixel P are defined by the computation of some parameters, as follows:

$$P_{Cr} - P_{Cb} \geq I_P \tag{3.2}$$

$$|P_{Cb} - P_{Cb_s}| \leq J_P \tag{3.3}$$

where I_P is the minimum difference between the values P_{Cr} and P_{Cb}, P_{Cb_s} is an estimated value of P_{Cb}, and J_P is the maximum distance between the points (P_Y, P_{Cb}) and (P_Y, P_{Cb_s}).

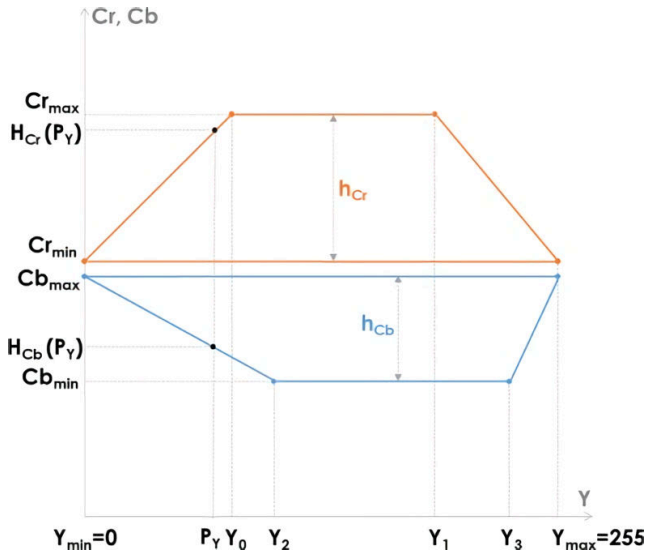


Figure 3.3: Graphical representation of T_{YCr} and T_{YCb}. Adapted from Brancati *et al.* 2017 [17].

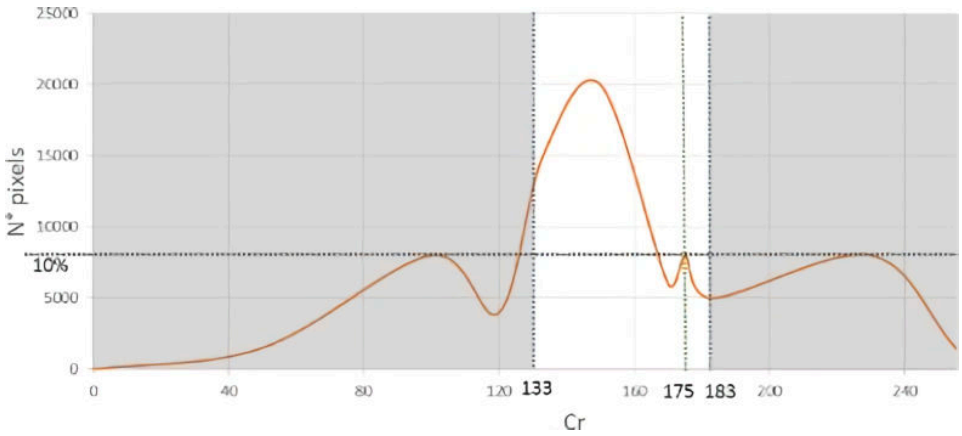


Figure 3.4: Computation of Cr_{max} on the histogram of Cr values. Adapted from Brancati *et al.* 2017 [17].

A pixel P is classified as a skin pixel if both the conditions hold. The first rule indicates that the chrominance components should be sufficiently far from each other. The second represents the range of values delimiting the P_{Cb_s} value, to which the P_{Cb} should belong. Figure 3.5 gives a visual representation of the operations to perform to compute the required parameters.

$$\begin{aligned} P_{Cb_s} &= Cb_{max} - dP_{Cb_s} \\ P_{Cr} &= Cr_{min} + dP_{Cr} \end{aligned} \tag{3.4}$$

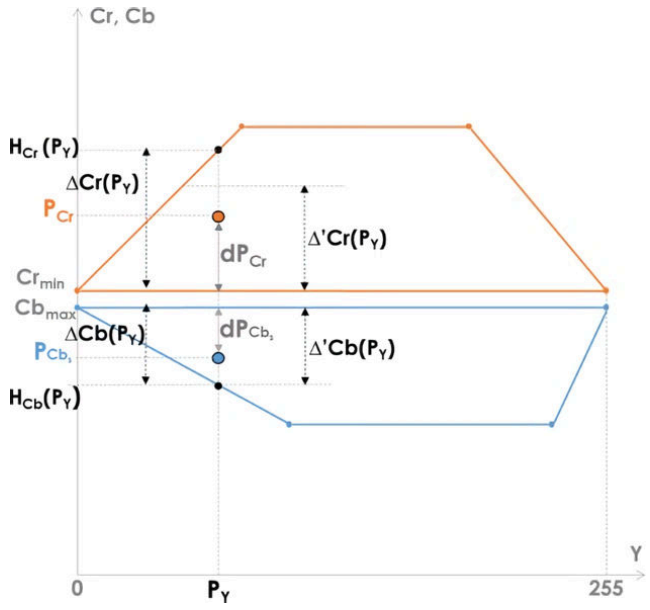


Figure 3.5: Computation of the correlation rules parameters. Adapted from Brancati *et al.* 2017 [17].

with dP_{Cb_s} and dP_{Cr} as the distances respectively between the points (P_Y, P_{Cb_s}) and (P_Y, P_{Cr}), and the longer bases of the trapezium in the corresponding subspaces. It is possible to compute dP_{Cb_s} with respect to dP_{Cr} on the basis of the inversely proportional behavior of the chrominance components observed within the trapezium:

$$dP_{Cb_s} = \alpha \cdot dP_{Cr} \tag{3.5}$$

where α takes into account the different shapes of the trapezium, being computed as the ratio between the normalized heights of the two trapeziums in correspondence with the current luminance value P_Y as follows:

$$\begin{aligned} \Delta_{Cr}(P_Y) &= H_{Cr}(P_Y) - Cr_{min} \\ \Delta_{Cb}(P_Y) &= Cb_{max} - H_{Cb}(P_Y) \end{aligned} \tag{3.6}$$

where $\Delta_{Cr}(P_Y)$ and $\Delta_{Cb}(P_Y)$ represent the distances between the points $(P_Y, H_{Cr}(P_Y))$ and $(P_Y, H_{Cb}(P_Y))$ and the longer base of their respective trapezium.

Now, the values of $\Delta_{Cr}(P_Y)$ and of $\Delta_{Cb}(P_Y)$ are normalized with respect to the difference in the size of the trapeziums:

$$\Delta'_{Cr}(P_Y) = \begin{cases} \Delta_{Cr}(P_Y) \cdot \frac{A_{T_{YCb}}}{A_{T_{YCr}}} & \text{if } A_{T_{YCr}} \geq A_{T_{YCb}} \\ \Delta_{Cr}(P_Y) & \text{otherwise} \end{cases}$$

$$\Delta'_{Cb}(P_Y) = \begin{cases} \Delta_{Cb}(P_Y) & \text{if } A_{T_{YCr}} \geq A_{T_{YCb}} \\ \Delta_{Cb}(P_Y) \cdot \frac{A_{T_{YCr}}}{A_{T_{YCb}}} & \text{otherwise} \end{cases} \tag{3.7}$$

where $A_{T_{YCr}}$ and $A_{T_{YCb}}$ are the areas of the trapeziums T_{YCr} and T_{YCb}, respectively. Then, the value α is given by:

$$\alpha = \frac{\Delta'_{Cb}(P_Y)}{\Delta'_{Cr}(P_Y)} \tag{3.8}$$

Finally, the remaining parameters I_P and J_P are given by:

$$I_P = sf \cdot [(\Delta'_{Cr}(Y) - dP_{Cr}) + (\Delta'_{Cb}(Y) - dP_{Cb_s})]$$

$$J_P = dP_{Cb_s} \cdot \frac{dP_{Cb_s} + dP_{Cr}}{\Delta'_{Cb}(Y) + \Delta'_{Cr}(Y)} \tag{3.9}$$

$$sf = \frac{\min((Y_1 - Y_0), (Y_3 - Y_2))}{\max((Y_1 - Y_0), (Y_3 - Y_2))}$$

3.2 Statistical

Large amount of data can support simple and computationally efficient learning algorithms [8]. Statistical methods aim at creating a model to understand how the data are related in order to make predictions (an example can be seen in Figure 3.6). When performing classification, these models not only allow to predict the class label, but also to obtain a probability of the respective label. This probability gives the confidence on the prediction.

Given a probability, different rules can be used to perform the classifications, with one of the most common being the Bayes rule. In binary classification problems, a single threshold is involved to choose at which probability one class should be classified over another. The thresholding rule is not unique, multiple rules have been used in the literature [8, 50].

In skin detection, one of the most popular work of statistical modeling has been published in 2002 [8]: a Bayesian classifier has been used to perform classifications starting from the probabilities given by statistical models. In the same work, skin and non-skin gaussian mixture models using a parallel implementation of the standard EM algo-

rithm [51] and 16 gaussians in each model have been trained. A comparison with histogram models of size 32 is presented. The results describe slightly better performance in the case of the histogram models, which also took only a matter of minutes to train, as opposed to about 24 hours required by the gaussian mixture models. A mixture model is also slower to use during classification since all the gaussians must be evaluated in computing the probability of a single color value. Contrarily, a histogram model translates to a fast classifier since only two table lookups are required to compute the probability of skin. However, from a storage point of view, the gaussian mixture models are much more compact: 896 bytes in contrast to 262 Kbytes of the histogram models.
Gomez *et al.* 2002 [52] presented a work in which two 3-dimensional histogram models are constructed to represent the skin and non-skin classes, and the probability is computed with a rule different than the Bayes theorem.

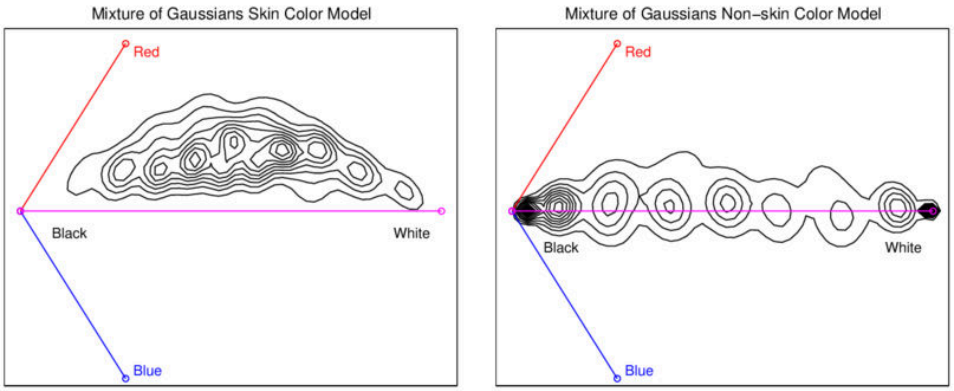


Figure 3.6: Mixture of gaussians are an example of statistical models. Adapted from Jones and Rehg 2002 [8]

3.2.1 Implementation

The chosen implementation[2] uses three-dimensional histograms to model the data and probability calculus to perform the classification [53]. The training data is used to construct two histogram models representing the probabilities of skin and non-skin classes, with RGB as the color space and a histogram size of 256 bins per color component. The models are saved as a lookup table. The thresholding value utilized to label skin pixels is 0.555555.
An overview of the algorithm is presented below:

[2] Source code available at https://github.com/Chinmoy007/Skin-detection

Training:

Step 1: Initialize two 3-dimensional data structures

Step 2: Pick an image and the corresponding mask from the training dataset

Step 3: Loop every (R,G,B) pixel of the image

Step 4: Pick the corresponding pixel from the mask: if it is a skin pixel, increase the value at position [R, G, B] of the data structure representing skin, otherwise increase the value in the other structure

Step 5: Return to the first step until the training images have all been processed

Predicting:

Step 1: Loop every (R,G,B) pixel from input image

Step 2: Calculate the probability of that (R,G,B) color combination of being skin

Step 3: If skin probability > Θ, the pixel is classified as skin

The key step in skin pixel classification is the computation of $P(skin \mid rgb)$, which is the probability of a given rgb pixel to belong to the skin class, and it is given by the following rule:

$$P(skin \mid rgb) = \frac{s[rgb]}{s[rgb] + n[rgb]} \tag{3.10}$$

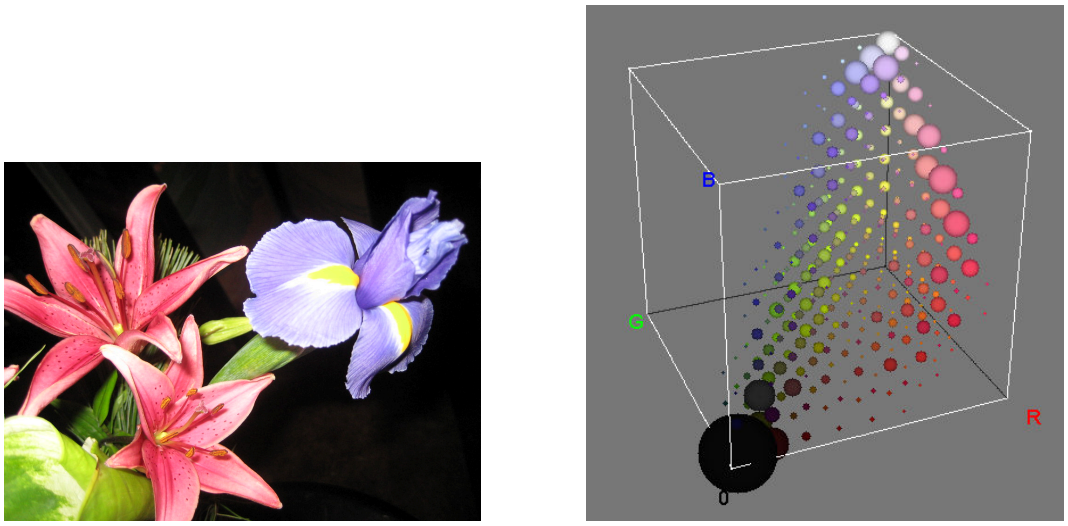
where $s[rgb]$ is the pixel count contained in bin rgb of the skin histogram and $n[rgb]$ is the equivalent count from the non-skin histogram.
A particular RGB value is labeled skin if:

$$P(skin \mid rgb) \geq \Theta \tag{3.11}$$

where $0 \leq \Theta \leq 1$ is a threshold value that can be adjusted to trade-off between true positives and false positives.
It is important to note that using color spaces other than RGB (such as YCBCr or HSV) will not improve the performance of the skin detector [8]. The Detector performance depends entirely on the amount of overlap between the skin and non-skin samples. Colors that occur in both classes with comparable frequencies cannot be classified reliably.

There are different ways to build the skin and non-skin color models. One of the most common is to utilize **histogram models**: the skin and non-skin pixels contained in the training set images are placed into the skin and non-skin histograms, respectively.

Before constructing an histogram model, two requirements are needed: a color space and the size of the histogram, which is measured by the number of bins per color channel. The fact that the performance of the classifier is independent of the color space, and RGB being the most common and natural color space used in image representation, makes RGB the favorite choice. Color images usually store 24 bits for each pixel, as there are three color components represented by 8 bits each. Each color component is thereby able to represent $2^8 = 256$ levels of intensity.



(a) (b)

Figure 3.7: 3D histogram representation: (a) original photo by Jose and Roxanne licensed under CC BY 2.0; (b) visualization made utilizing the "3D Color Inspector" plugin of ImageJ.[3]

When building the histogram, the number of bins is important. Too few bins could result in poor accuracy, while too many bins could lead to over-fitting [8]. Reducing the number of bins helps to "generalise" and compact the histogram [52]. By using fewer bins, a set of points of similar color is represented by a single color (this technique is called color quantization. An example of a 3D histogram using the technique is depicted in Figure 3.7). By using a 24-bit pixel representation, the histogram model would have a size of 256 bins per color channel, which correspond to more than 16 million (256^3) bins, each mapped to a specific (R,G,B) color triple. Jones and Rehg (2002) [8] found the best histogram size in their experiments to be 32. They also reported that, with a histogram size of 256, 77% of the possible 24-bit RGB colors are never encountered, and thus the histogram is mostly empty.
Using a histogram size of 256 in the RGB color space means that each of the three his-

[3]Plugin available at https://imagej.nih.gov/ij/plugins/color-inspector.html

togram dimensions is divided into 256 bins. And each bin stores an integer counting the number of times that color value occurred in the entire database of images.

3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of Neural Network commonly applied to analyze visual scenes. Their structure is inspired by the human brain, mimicking the way that biological neurons signal to one another.

3.3.1 Neural Networks

A **biological neuron** has specialized protrusions called dendrites and axons. Dendrites bring information to the cell body, and axons take information away from the cell body. Information flows from one neuron to another across a small gap between them named synapse. The dendrites carry the signal to the cell body where they all get summed; if the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon.

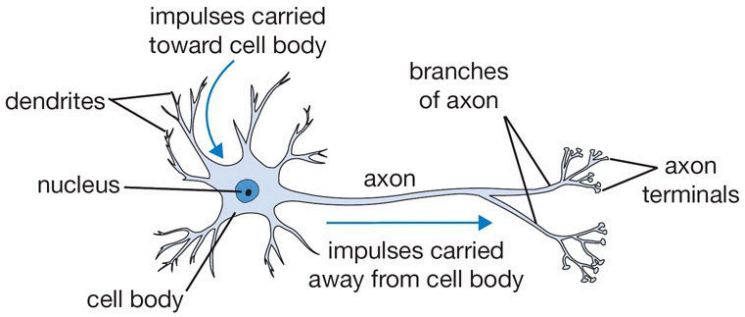


Figure 3.8: Biological neuron.[4]

In the **computational model of a neuron**, also named Perceptron, the signals that travel along the axons x_i interact multiplicatively $(w_i x_i)$ with the dendrites of the other neuron based on the synaptic strength w_i at that synapse. The dendrites bring the signals to the cell body, where they all get summed. The idea is that the synaptic strengths (the weights w) are learnable and control the strength of influence of one neuron on another. The neuron then performs a transformation to the input through the activation function $f(x)$ and generates the output y. The activation function is typically a non-linear transformation that is applied to the input data. The bias value b allows the activation function to be shifted to the left or right, to better fit the data. The weights and biases represent the parameters of the network.

A **Neural Network** is a group of multiple neurons connected together, as shown in Figure 3.10. Neurons are typically organized into different layers. Neurons of one layer

[4] The figure is adapted from https://cs231n.github.io/neural-networks-1/

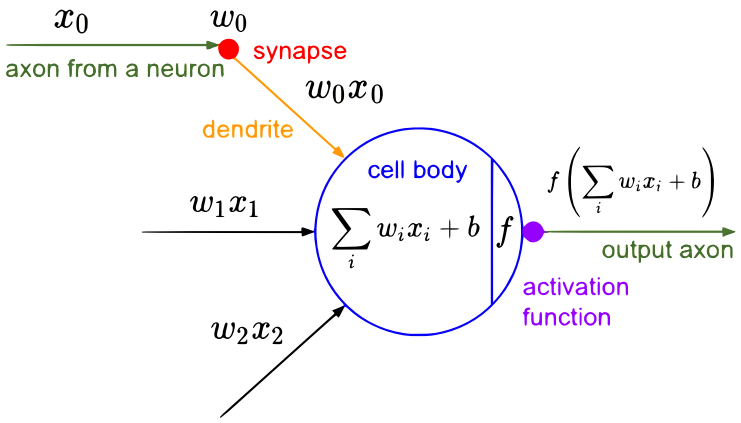


Figure 3.9: Neuron computational model.

connect only to neurons of the immediately preceding and immediately following layers, therefore the output of a layer becomes the input of the next layer. The first layer is the input layer, where each neuron represents a feature in the dataset, the last layer is the output layer, and all the other layers are the hidden layers. Although a single output is depicted in the figure, the output layer can have different sizes. It can vary from one output for a single class classification to thousands of pixels of an image classification map.
Once the dataset and problem are defined, the main steps to train a Neural Network are the following:

Step 1: Construct the network architecture and initialize with random weights

Step 2: Do a forward pass (Forward propagation)

Step 3: Calculate the total error, which needs to be minimized

Step 4: Back propagate the error and Update weights

Step 5: Repeat the steps(2-4) for No. of epochs/until error is minimum.

In **forward propagation**, the input data is fed to the network in the forward direction: each hidden layer gets the data, perform calculation and pass the result to the next layer. Finally, the output layer calculates the output of the model.
The forward propagation for the network in Figure 3.10 is described below. The input is represented by X, which can be imagined as a matrix of 2 rows and 1 column. The next layer is represented by z_{11}, z_{12}, z_{13}, which are the values of the intermediate neurons calculated from the weight, bias, and neuron values of the previous layer. This layer can be imagined as a matrix of 3 rows and 1 column. In this case, there are two input nodes

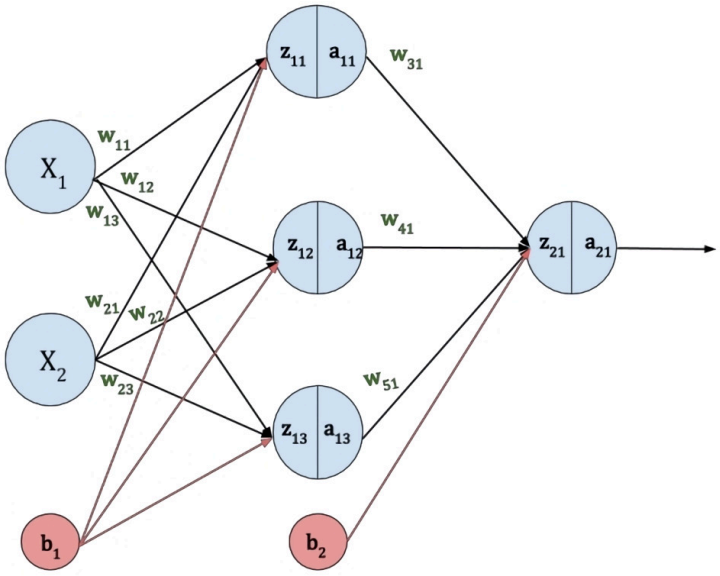


Figure 3.10: A two-layer Neural Network architecture. Conventionally, the input layer is not counted for defining the number of layers of a Neural Network. There are two input features: x_1 and x_2, a hidden layer with three neurons, and an output layer with one neuron. Each neuron has assigned the weight parameter W_{ij}. The b_1 and b_2 are the bias parameters for the input layer and hidden layer, respectively.

and three output nodes that the parameters must fit. $W^{(\text{layer})}_{\text{pln, nln}}$ is the parameter to be optimized, in which pln and nln represent the previous layer input neuron and the next layer output neuron, respectively. The bias parameter for the input layer is represented by b_1.

$$\begin{bmatrix} z_{11} \\ z_{12} \\ z_{13} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_1 \\ b_1 \end{bmatrix} \tag{3.12}$$
$$Z^{[1]} = W^{[1]} X + b^{[1]}$$

An activation function $\sigma(Z)$ will be applied to the intermediate neuron values to learn the non-linear patterns between inputs and target output variables. The a_{11}, a_{12}, and a_{13} values are the output of the activation function that is applied to z_{11}, z_{12} and z_{13}, respectively.

$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} = \sigma \begin{bmatrix} z_{11} \\ z_{12} \\ z_{13} \end{bmatrix} \tag{3.13}$$
$$A^{[1]} = \sigma\big(Z^{[1]}\big)$$

At the next step, $A^{[1]}$ becomes the input layer and z_{21} becomes the output layer. There are three input nodes and an output node that the parameters must fit, so $W^{[2]}$ is a matrix with one row and three columns. The bias parameter for the hidden layer is represented by b_2.

$$\begin{gathered}[z_{21}] = \begin{bmatrix} w_{31} & w_{41} & w_{51} \end{bmatrix} \begin{bmatrix} a11 \\ a12 \\ a13 \end{bmatrix} + [b_2] \\ Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}\end{gathered} \tag{3.14}$$

The activation function $\sigma(Z)$ is applied to the obtained values, and the algorithmic step represented by this layer finishes.

$$\begin{aligned}[a_{21}] &= \sigma[z_{21}] \\ A^{[2]} &= \sigma\big(Z^{[2]}\big)\end{aligned} \tag{3.15}$$

The output of the neural network is calculated, hence the forward propagation algorithm ends.

Activation functions are an important part of Neural Networks. Without them, the networks are just the weighted sum of their inputs plus a bias term, unable to learn any complex and nonlinear function. Activation functions are means to introduce non-linearity to the model. There are different classes of activation functions available, with common ones being the Sigmoid, the ReLU, the Tanh, and the Linear activation functions. The mentioned activation functions are illustrated in Figure 3.11.

The next step in a Neural Network is to **compute the loss (error)**. Once the output of the network is obtained, it is compared to the desired output value (the so-called ground truth), and a loss function computes the error signal, which measures how well the network predicts outputs.
A popular loss function is the Cross-Entropy Loss, defined as follows:

$$\begin{aligned}J &= -\frac{1}{m} \sum_{i=1}^{m} L\big(a^{[2](i)}, y^{(i)}\big) \\ L\big(a^{[2]}, y\big) &= -y \log a^{[2]} - (1-y) \log\big(1 - a^{[2]}\big)\end{aligned} \tag{3.16}$$

where L is the loss function and J is the cost function: the average loss over the entire training dataset.
The goal is to then find a set of weights and biases that minimizes the error. However, it is impossible to compute the error signal for internal neurons directly because the output values of these neurons are unknown. The idea is to propagate the error signal back to all neurons by computing the gradient of the loss function with respect to each weight. The derivative of the loss function on each parameter gives information about how changing

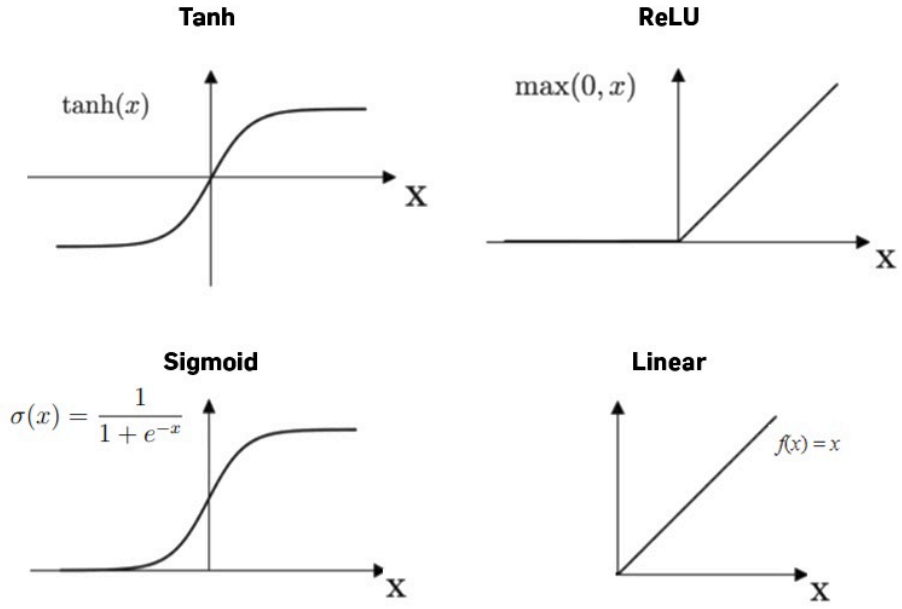


Figure 3.11: Common activation functions. The Tanh function forces the values to be between -1 and 1. The Sigmoid function forces the values to be between 0 and 1. The ReLU function cuts values below zero. The Linear activation function returns the input.

the parameter impacts the function value. The algorithm that computes the gradient is called **backpropagation**.

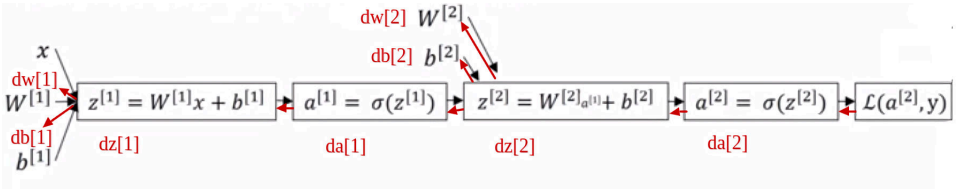


Figure 3.12: The red line represents the back-propagation process. The $da[2]$, $dz[2]$, $dw[2]$, $db[2]$, $da[1]$, $dz[1]$, $dw[1]$ and $db[1]$ are the partial derivative of the loss function with respect to $a[2]$, $z[2]$, $w[2]$, $b[2]$, $a[1]$, $z[1]$, $w[1]$ and $b[1]$ respectively.

While performing the derivative calculation, it is also necessary to calculate the derivative of the activation function. Below is presented the derivative of the Sigmoid Activation function:

$$
\begin{aligned}
\sigma(x) &= \frac{1}{1+e^{-x}} = \left(1+e^{-x}\right)^{-1} \\
\sigma'(x) &= \left(1+e^{-x}\right)^{-2} e^{-x} \\
&= \frac{e^{-x}}{\left(1+e^{-x}\right)^2} \\
&= \frac{1}{(1+e^{-x})} * \frac{1+e^{-x}-1}{(1+e^{-x})} \\
&= \frac{1}{(1+e^{-x})} * \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}
\tag{3.17}
$$

The computation of the partial derivative of weight and bias parameters is done as follows:

$$
\begin{aligned}
da^{[2]} &= \frac{\partial L}{\partial a^{[2]}} & da^{[1]} &= \frac{\partial L}{\partial a^{[1]}} \\
dz^{[2]} &= \frac{\partial L}{\partial z^{[2]}} & dz^{[1]} &= \frac{\partial L}{\partial z^{[1]}} \\
dw^{[2]} &= \frac{\partial L}{\partial w^{[2]}} & dw^{[1]} &= \frac{\partial L}{\partial w^{[1]}} \\
db^{[2]} &= \frac{\partial L}{\partial b^{[2]}} & db^{[1]} &= \frac{\partial L}{\partial b^{[1]}}
\end{aligned}
\tag{3.18}
$$

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified to minimize the error. The **weight updating** is commonly performed by a gradient descent algorithm, which takes advantage of the already calculated gradients. The weight and bias parameters are updated by subtracting the partial derivative of the loss function with respect to those parameters. A visualization of the gradient descent algorithm is displayed in Figure 3.13. The step size can be modified accordingly thanks to the learning rate coefficient α, which controls how much to update the parameter.

$$
\begin{aligned}
W^{[1]} &= W^{[1]} - \alpha \frac{\partial L}{\partial W^{[1]}} \\
b^{[1]} &= b^{[1]} - \alpha \frac{\partial L}{\partial b^{[1]}} \\
W^{[2]} &= W^{[2]} - \alpha \frac{\partial L}{\partial W^{[2]}} \\
b^{[2]} &= b^{[2]} - \alpha \frac{\partial L}{\partial b^{[2]}}
\end{aligned}
\tag{3.19}
$$

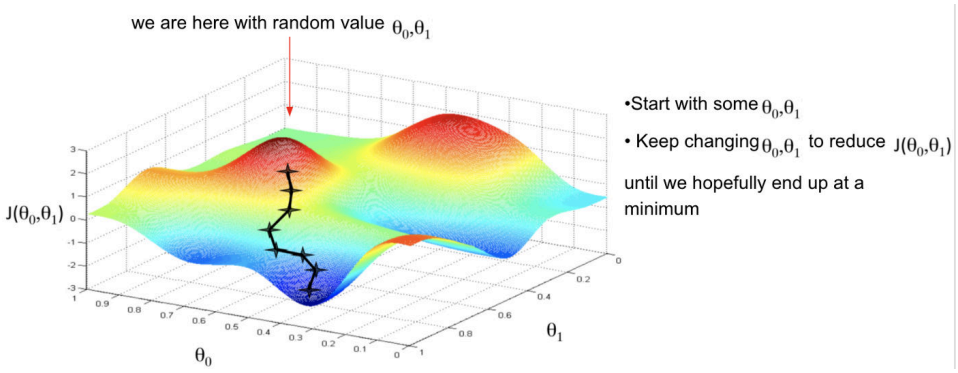


Figure 3.13: Visualization of the gradient descent algorithm. θ_0 and θ_1 represent the parameters to update: the bias and the weight values, respectively.

Convolutions in Neural Networks

A Convolutional Neural Network (CNN) has the same architecture as the usual Neural Network but arranges its neurons in three dimensions (width, height, depth). Every layer of a CNN transforms the 3D input volume to a 3D output volume of neuron activations. There are three typologies of layers utilized in a CNN architecture: the convolutional layer, which increases the efficiency of the forward function and vastly reduces the number of parameters, the pooling layer, and the fully-connected layer, which is the regular layer present in Neural Networks. Typically convolutional layers are followed by an activation function, such as a ReLU. The convolutional and pooling layers are used for feature extraction, while the fully connected layers are used for classification. An architecture overview is displayed in Figure 3.14.

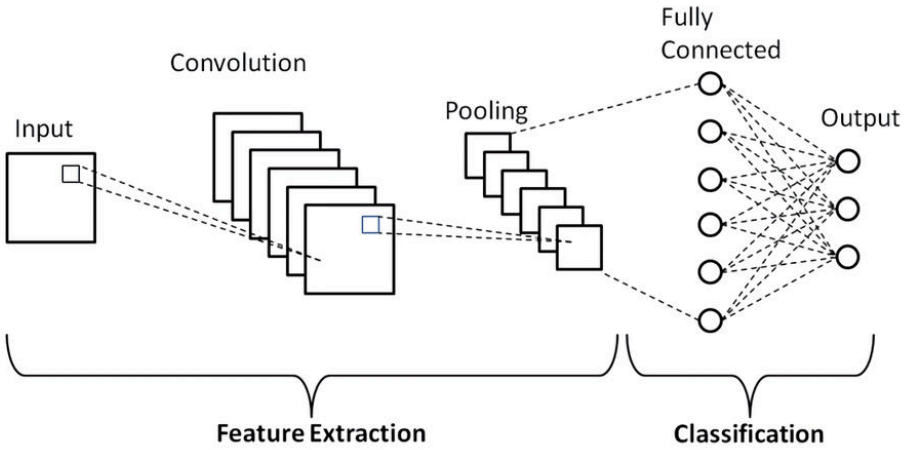


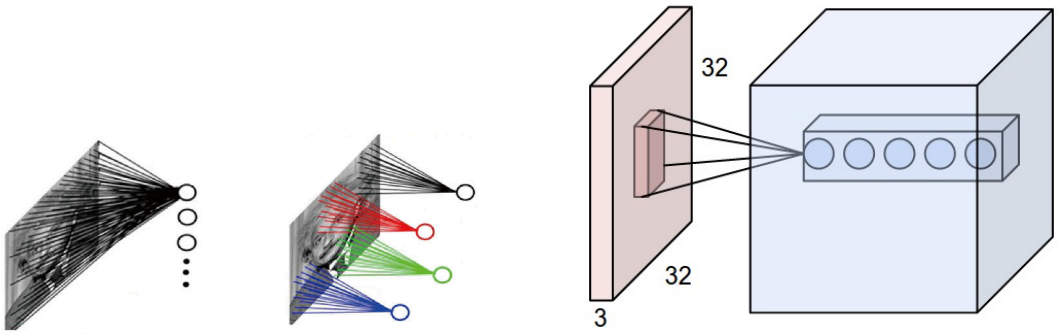
Figure 3.14: Overview of a CNN architecture.

The **Convolutional layer**'s parameters consist of a set of learnable filters. Every filter is small spatially (along width and height) but extends through the full depth of the input volume. A typical filter on the first layer of a Convolutional Neural Network might have size 5x5x3 because colored images have depth 3, the color channels. During the forward pass, each filter is slid across the width and height of the input volume. The dot products between the entries of the filter and the input at any position are computed. As the filter slides over the width and height of the input volume, it produces a 2-dimensional activation map that represents the responses of that filter at every spatial position. In this way, the network will learn filters that activate when they see some type of visual features, such as an edge. Inside a convolutional layer, there is a set of filters: each of them extracts a particular feature and produces a separate 2-dimensional activation map. These activation maps are stacked along the depth dimension and produce the output volume.

The first advantage of a convolutional layer is the **local connectivity** (depicted in Figure 3.15a): each neuron of the layer is connected only to a local region of the input volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron (equivalently this is the filter size). The connections are local in 2D space (along width and height) but always full along with the entire depth of the input volume.

The second advantage is **parameter sharing**. It is possible to dramatically reduce the number of parameters by making one reasonable assumption: if one feature is useful to compute at some spatial position (x,y), then it should also be useful to compute at a different position (x2,y2). A practical application of this reasoning can be found in Figure 3.16. The parameter sharing is done by slicing the depth of the volume size into 2-dimensional slices, called depth slices. The neurons in each depth slice share the same weights and bias. For example, a volume of size 55x55x96 has 96 depth slices, each of size 55x55; by using a filter size of 11x11x3 on the volume, the convolutional layer will have only 96 unique sets of weights for a total of $96 * 11 * 11 * 3 = 34\,848$ unique weights, or 34 944 parameters (+96 biases). Without this technique, the parameters would have been much more: $96 * 55 * 55 = 290\,400$ neurons, each using $11 * 11 * 3 = 363$ weights and 1 bias, for a total of $290\,400 * 364 = 105\,705\,600$ parameters. During backpropagation, every neuron in the volume will compute the gradient for its weights, but these gradients will be added up across each depth slice, and only update a single set of weights per slice.

It is common to periodically insert a **Pooling layer**, which performs a form of non-linear down-sampling, in-between successive Convolutional layers in a CNN architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. There are several non-linear functions to implement pooling, where max pooling is the most common (depicted in Figure 3.17. It partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum. It is important to note that pooling loses the precise spatial relationships between high-level parts (such as nose and mouth in a face image).



(a) (b)

Figure 3.15: Some Convolutional layer features. (a) Global (left) and local (right) perception. (b) The red volume represents an example input, a 32x32x3 image, and the blue volume is an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (all color channels). Note, there are multiple neurons (5 in this example) along with the depth, all looking at the same region in the input: these neurons share the same receptive field. Despite this, they do not share the same weights, because they are associated with 5 different filters.

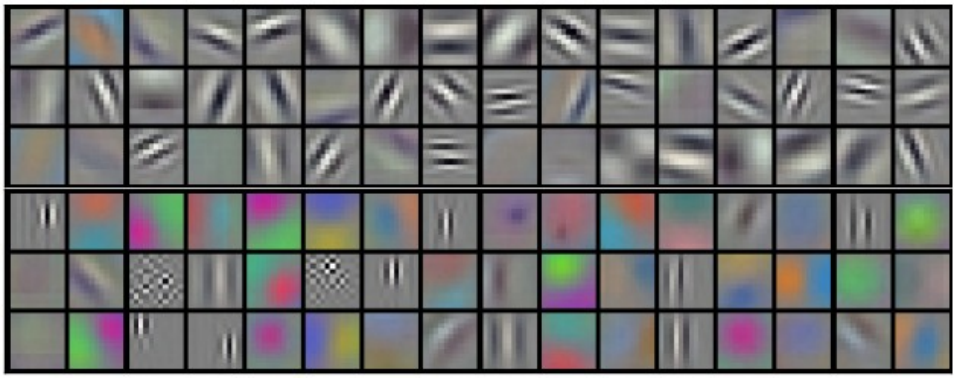


Figure 3.16: Example filters adapted from Krizhevsky *et al.* 2012 [54]. Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the 55*55 neurons in one depth slice. Notice that the parameter sharing assumption is relatively reasonable: if detecting a horizontal edge is important at some location in the image, it should intuitively be useful at some other location as well. There is therefore no need to relearn to detect a horizontal edge at every one of the 55*55 distinct locations in the Conv layer output volume.

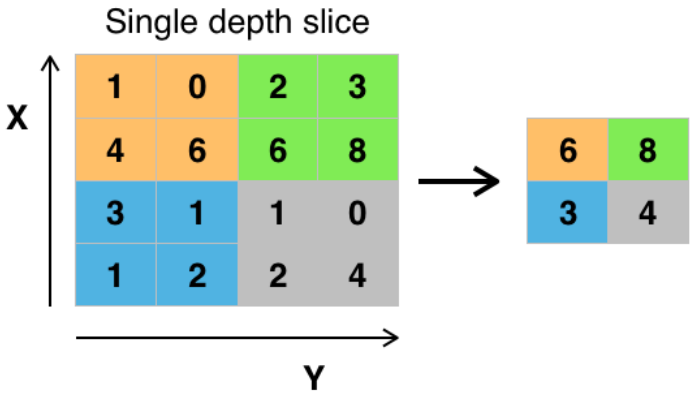


Figure 3.17: Max pooling with a 2x2 filter and stride = 2: the filter moves two pixels right for each horizontal movement and two pixels down for each vertical movement.

3.3.2 Implementation

The chosen implementation[5] consists of a modified U-Net [14] incorporating dense blocks and inception modules to benefit from a wider spatial context. The network is named Skinny [25] and is depicted in Figure 3.18. An additional deep level is appended to the original U-Net model, to better capture large-scale contextual features in the deepest part of the network. The features extracted in the contracting path propagate to the corresponding expansive levels through the dense blocks. The original U-Net convolutional layers are replaced with the inception modules: before each max-pooling layer, in the contracting path, and after concatenating features, in the expanding path. Thanks to these architectural choices, Skinny benefits from a wider pixel context.

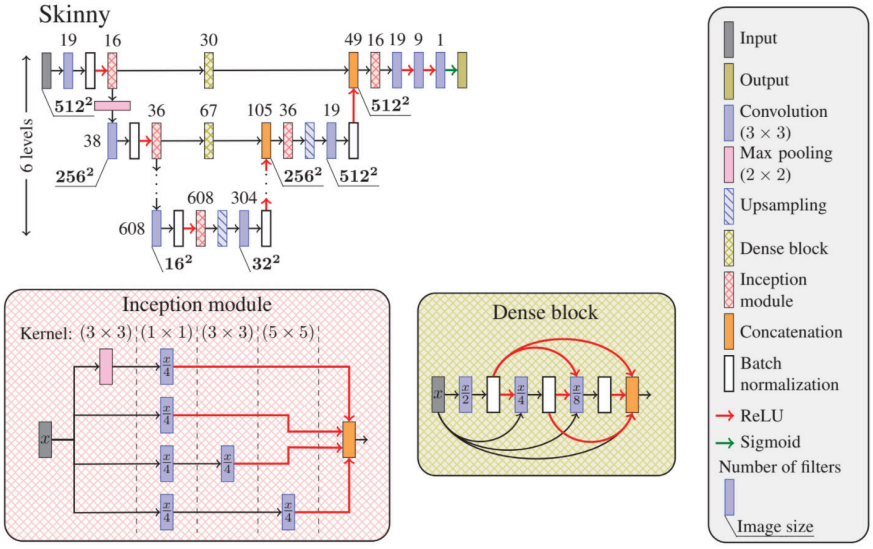


Figure 3.18: The architecture of Skinny. Adapted from Tarasiewicz *et al.* 2020 [25]

In image segmentation, it would be useful for the output of a Neural Network to be directly a classification map. It can be achieved by decapitating the fully connected layers of a CNN network, converting it into a **Fully Convolutional Network (FCN)** [55]. In fact, fully connected layers can be viewed as convolutions with kernels that cover their entire input regions. An example of a fully convolutional network is the **U-Net** [14] (called in this way because of its U shape, which can be seen in Figure 3.19), a famous network used for semantic segmentation. The architecture of an FCN can be seen as a union of two networks: an encoder, which takes the input and output a feature map, and a decoder, which takes the feature vector from the encoder and gives a classification map in output.

[5] Source code available at `https://github.com/ttarasiewicz/Skinny`

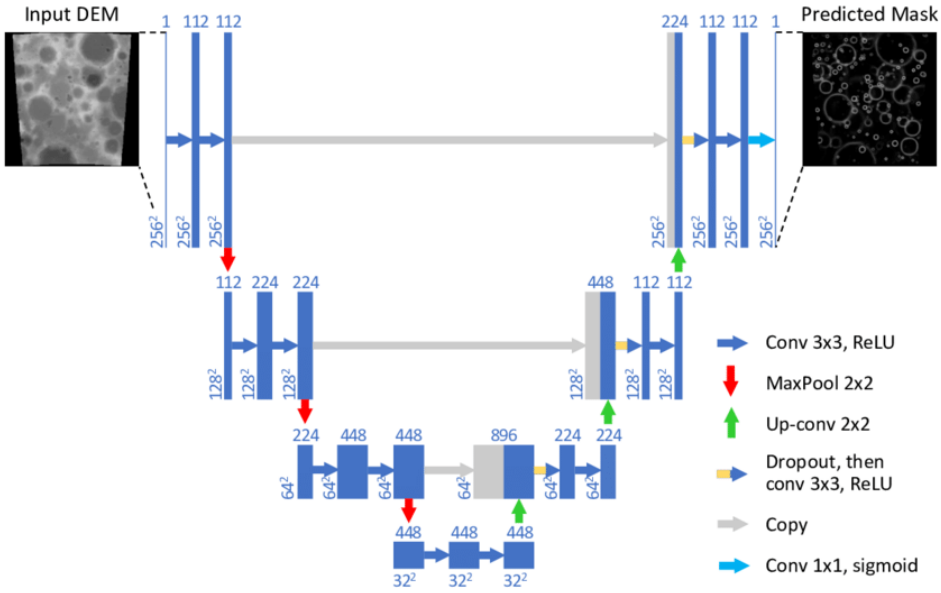


Figure 3.19: U-Net is a Fully Convolutional Network that utilizes the encoder-decoder architecture. The encoder consists of the contracting pathway of the network, while the decoder consists of the expanding pathway. Adapted from Silburt *et al.* 2019 [56]

The encoder performs the feature extraction task via multiple down-sampling operations, in the same way as the first part of a CNN. The decoder must then up-sample these feature maps multiple times until the achievement of the desired classification map, which size is determined by the architecture. The upsampling operation is possible thanks to the skip connections. **Skip connections** are used to skip features from the contracting path to the expanding path in order to recover spatial information lost during downsampling, making fully convolutional methods suitable for semantic segmentation. The **Up-sampling layer** consists of a transposed convolution, an operation that goes in the opposite direction to a convolution, as shown in Figure 3.20.

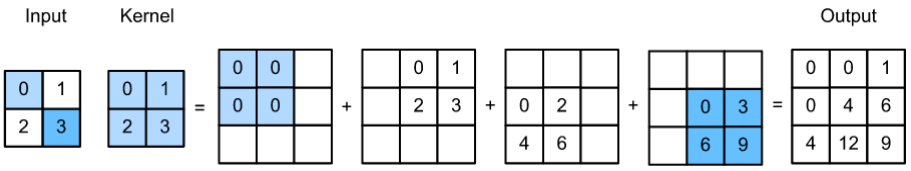
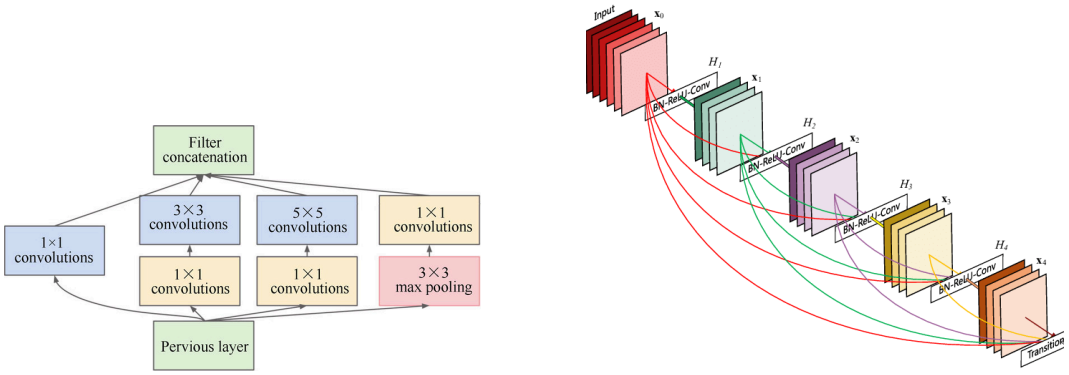


Figure 3.20: Transposed convolution. Each element of the input feature map is taken and multiplied with every element of the kernel. The outputs of these operations are then summed.

The main advantage of the chosen implementation compared to a regular U-Net is the addition of inception modules and dense blocks. Salient parts in the image can have extremely large variations in size. For example, an image can represent a dog nearby or far away, and the size of the part of the image occupied by the dog varies. Because of the huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough. A larger kernel is preferred for information distributed more globally, and a smaller kernel is preferred for information distributed more locally. The solution is to have filters with multiple sizes operate on the same level. The **inception module** [57] (displayed in Figure 3.21a) performs this operation.

Dense blocks [58] strengthen feature propagation and reuse. A dense block comprises n dense layers. These dense layers are connected using a dense circuitry such that each dense layer receives feature maps from all preceding layers and passes its feature maps to all subsequent layers. The dimensions of the features (width, height) stay the same in a dense block, but the number of filters changes between them. A dense layer can be imagined as a convolutional layer, followed by a batch normalization layer, and the activation function.

The goal of **Batch Normalization** [59] is to achieve a stable distribution of activation values throughout training. The distribution of the inputs to layers somewhere down in the network may change after each mini-batch of input images, as the weights refresh. This can make the learning algorithm always pursue a moving target. In a neural network, batch normalization is achieved through a normalization step that fixes the means and variances of each layer's inputs. Ideally, the normalization would be conducted over the entire training set. However, to use this step jointly with optimization methods, it is impractical to use global information. Thus, normalization is restrained to each mini-batch in the training process.



(a) (b)

Figure 3.21: (a) Inception module. 1x1 convolutions reduce the dimension along the direction of the number of channels, making the process less computationally expensive. Adapted from Szegedy *et al.* 2015 [57]. (b) A 5-layer dense block with a growth rate of k = 4. Every layer has access to its preceding feature maps, and therefore, to the collective knowledge. Each layer adds then new information to this collective knowledge, in concrete k feature maps of information. Adapted from Huang *et al.* 2017 [58].

4 Results

4.1 Metrics

Metrics give accurate measurements about how a process is functioning and provide a measure to suggest the improvements. Developing and understanding standardized metrics is crucial to guide decision-making and prevent the need to make hasty decisions [60].

As stated in 1.1, skin detection is a two-class problem. To evaluate a binary classifier, the ideal data (the ground truth) is compared to the predicted data given by the method. In the case of skin detection, data is generally represented by binary masks. The primitive metrics to define are the ones inside a confusion matrix, depicted in Figure 4.1. A **False Positive** is an error in binary classification in which a test result incorrectly indicates the presence of a condition such as a disease when the disease is not present. A **False Negative** represents the opposite case: the test result incorrectly fails to indicate the presence of a condition when it is present. These are the two kinds of errors in a binary test, in contrast to the two kinds of correct results (a **True Positive** and a **True Negative**). It is important to note that the gravity of an error depends on the contest. An example of different error seriousness is illustrated in Figure 4.2. In image segmentation, False Positives may represent a lesser gravity since a follow-up processing of the image could fix the issues, while the data of false negatives is lost.

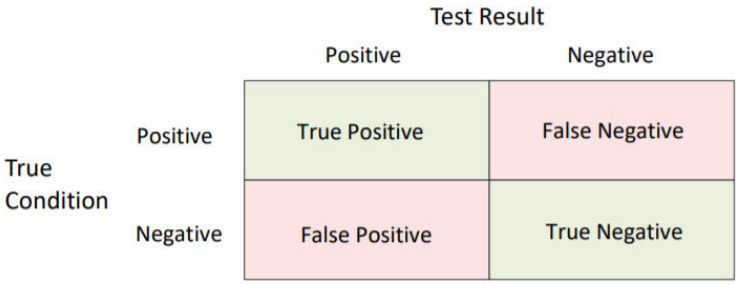


Figure 4.1: A confusion matrix.

Starting from the primitive metrics, it is possible to compute more complex evaluation metrics, such as Recall, Specificity, and Precision.

Recall, also called True Positive Rate or Sensitivity, represents how many relevant items are selected.

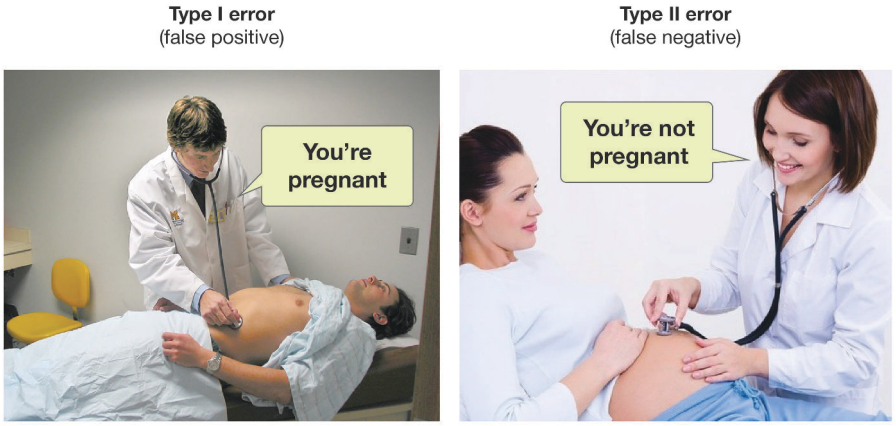


Figure 4.2: The gravity of an error depends on the contest in which it happens. False Negatives and False Positives can have different gravity.

Specificity, also called False Positive Rate, represents how many negative elements are truly negative.
Precision represents how many selected items are relevant.
These metrics are calculated as follows:

$$\begin{aligned} Recall &= \frac{TP}{TP + FN} \\ Specificity &= \frac{TN}{TN + FP} \\ Precision &= \frac{TP}{TP + FP} \end{aligned} \tag{4.1}$$

where TP, TN, FP, FN are respectively True Positives, True Negatives, False Positives, and False Negatives.
Recall, Specificity, and Precision may not be enough to evaluate a classifier, so they are often combined into other metrics, such as F_1-Score, IoU, and D_{prs}.
F_1-Score, also called F-Measure, Sørensen-Dice coefficient, or Dice similarity coefficient, is a measure of a test's accuracy. The F_1 in its name means that the β value of the generic F-Score is set to 1. The F_1 score is the harmonic mean of precision and recall. The more generic F_β score applies additional weights, valuing one of precision or recall more than the other.
Intersection Over Union (IoU), also named Jaccard index, measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the

union of the sample sets. The sample sets are represented by the ground truth and the prediction given by a skin detector.
D_{prs} [61] is a metric that focuses on segmentation algorithms. The already mentioned metrics are expressed as a compromise between two of the three aspects that are important for a quality assessment: Precision, Recall, and Specificity [61]. D_{prs} takes into account all of the three aspects. It measures the Euclidean distance between the segmentation, represented by the point (PR, RE, SP), and the ground truth, the ideal point $(1, 1, 1)$, hence lower values are better in this case.
These metrics are calculated as follows:

$$\begin{aligned} F_1 &= \frac{2TP}{2TP + FP + FN} \\ IoU &= \frac{TP}{TP + FP + FN} \\ D_{prs} &= \sqrt{(1 - PR)^2 + (1 - RE)^2 + (1 - SP)^2} \end{aligned} \tag{4.2}$$

where PR, RE, SP are Precision, Recall, and Specificity, respectively.

It is possible to note that F_1-Score and IoU have a similar formula: the only difference is that F_1 weights the True Positives higher. When taking the average score of each metric over a set of inferences, IoU tends to penalize single instances of bad classification more than the F_1 score [1]. Suppose for example that the vast majority of the inferences are moderately better with classifier A than B, but some of them are significantly worse using classifier A. It may be the case then that the F_1 metric favors classifier A while the IoU metric favors classifier B. For a better interpretation of the average scores between multiple instances, the difference **F_1-IoU** is taken into account. The higher the difference, the more is the bad inferences in the set, while a lower difference means that the prediction set is more balanced. All the described metrics range from 0 to 1, except for primitives, which are whole numbers, and D_{prs}, which ranges from 0 to $\sqrt{3}$.

It is also important to note that F_1 and IoU do not take into account the True Negatives. In the case of skin detection the performances are generally measured on the skin class, hence this limitation is negligible. However, if True Negatives are important for an evaluation, more balanced metrics should be taken into account, such as the Matthews correlation coefficient (MCC) [36].

[1] A detailed explanation can be found at https://stats.stackexchange.com/a/276144

4.2 Experimental setup

Before the evaluation process on the chosen datasets, the selected methods have been validated on the datasets splits used in their original papers. In this way, it has been possible to check their proper functioning. The statistical method [53] has not been validated because it does not refer to a paper and evaluations are not reported. The thresholding method [17] uses a metrics averaging of this type: the Recall, Precision, and Specificity measures are calculated as average scores over the set of instances, then the obtained scores are used to calculate F_1-Score and D_{prs}. The U-Net approach [25], instead, calculates the F_1-Score directly as the average score over all the set of instances. The validation results are shown in Table 4.1 and Table 4.2.

	F_1-Score			D_{prs}		
	HGR[1]	ECU	Pratheepan	HGR[1]	ECU	Pratheepan
Original	0.8252	0.6550	0.6592	0.2667	0.5043	0.5149
Implementation	0.8257	0.6586	0, 6630	0.2660	0.5006	0.5096
Change	0.0005	0.0036	0.0038	0.0007	0.0037	0.0053

Table 4.1: Brancati *et al.* 2017 [17] validation data.
Each dataset was used in its entirety to perform the testing.
[1]HGR consists of: HGR1, HGR2A-downscaled, HGR2B-downscaled.

	F_1-Score	
	HGR[1]	ECU[2]
Original	0.9494	0.9230
Implementation	0.9308	0.9133
Change	0.0186	0.0097

Table 4.2: Tarasiewicz *et al.* 2020 [25] validation data.
[1]HGR consists of: HGR1, HGR2A-downscaled, HGR2B-downscaled.
[2]ECU was split accordingly to the original work of the method.
The model was trained on the ECU splits; HGR has not been used for training.
The testing was performed on the test set of ECU and the entirety of HGR.

After the validation phase, the setup of the experiments begins with the definition of the employed datasets:

- ECU: 3998 images.
- HGR: 1558 images from HGR1, HGR2A-downscaled, HGR2B-downscaled sub-datasets.

- Schmugge: 840 images. The pictures have been processed by reading the `.config.SkinImManager` file provided by the dataset. Five pictures have been ignored because of faulty ground truths: `aa50.gt.d3`, `dd71.gt.d3`, `hh54.gt.d3`, and `aa69.gt.d3` that has a duplicated entry in the file. The rule used to manage the ternary ground truths has been considering whatever is not the background as skin.

Skin tones sub-datasets are defined by taking advantage of the additional labels in the Schmugge dataset. The resulting sub-datasets represent light, medium, and dark skin tones and consist of 409, 101, and 27 pictures, respectively. Not all Schmugge dataset is used for skin tones, because the database contains whole images of non-skin pixels, as described in section 2.3.

None of the datasets provide native training and testing splits. The train and validation sets have been merged in methods that do not use the validation set. For ECU, the splits used are the ones mentioned in Tarasiewicz *et al.* 2020 [25]. HGR, Schmugge, and skin tone splits are randomly defined by keeping the following proportions: 70% train, 15% validation, and 15% test. Since the proportions would reduce the dark sub-dataset to only 19 training images, the data has been augmented in this case, only for the training split. The purpose is to get at least 100 images for the evaluations, but also to generate images that look natural; 104 was the final size of the training set. The following augmentation operations provided by the Albumentations library [62] are performed in the following order:

- `HorizontalFlip(p=1)` on original images.
- `Rotate(limit=15,p=0.8)` on the original images plus the ones obtained by the horizontal flipping.
- `RandomCrop(H*0.8,W*0.8,p=0.8)` on the original images plus the ones obtained by the horizontal flipping plus the ones obtained by the rotation.

where `H` and `W` are the height and width of the processed image, respectively, and `p` and `limit` are the probability that the transformation happens, and the rotation limits in degrees (the angle is between `-limit` and `+limit`), respectively.

Once the datasets have been defined, the settings of the methods can be adjusted. The thresholding [17] and statistical [53] approaches have been left to their default implementations settings described in subsection 3.1.1 and subsection 3.2.1, respectively. The U-Net approach [25] uses only the complete model named "Skinny" with the following settings:

- Maximum number of epochs = 200
- Batch size = 3
- Initial learning rate = 10^{-4}

- Minimum learning rate = 10^{-6}
- Reduce learning rate on plateau patience = 5
- Early stopping patience = 10 for HGR, ECU, and Schmugge models.
  Early stopping patience = 50 for light, medium, dark models (by having only 4 validation images in the dark sub-dataset, the model has a harder time taking the right path).
- Adam [63] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$
- The loss function is the average of Binary Cross-Entropy and the Dice coefficient.
- The original preprocessing: the images with over (512×512) pixels are downscaled preserving the aspect ratio so that the number of pixels does not exceed (512×512). Moreover, Skinny applies a padding operation after the possible downscale to make the width and height of images multiple of 32, as seen in Figure 4.3.
- The models have been trained by monitoring the F_1 over the validation set and saving on new max values.

The last phase of the setup is defining the evaluation process. Some methods did not provide a binary prediction, but a prediction with grayscale values. All the predictions have been binarized by rounding each pixel value to either be white or black. Initially, the metrics are measured for all the instances, then the average and population standard deviation for each metric are computed.

Two different settings of dataset evaluation are performed: single and cross evaluation. The first measures how well a method performs with respect to the same dataset, the second measures how well a method generalizes with respect to other datasets.
In the **single evaluation** of a method over a dataset, the method is eventually trained on the training set, in the case of a trainable method, and then predictions are performed on the test set.
In the **cross evaluation**, only the trainable approaches are analyzed. The models are the same of the single evaluation: each model is trained on the training set of a dataset. The evaluation of a method is performed over the whole dataset by the models that have not been trained on the concerning dataset.

(a) (b) (c) (d)

Figure 4.3: Skinny [25] applies padding to make the width and height of an image a multiple of 32. If the dimensions of the image are already a multiple of 32, a padding of 32 is still applied. (a) original image; (b) ground truth; (c) preprocessed original image; (d) preprocessed ground truth

4.3 Performance on single databases

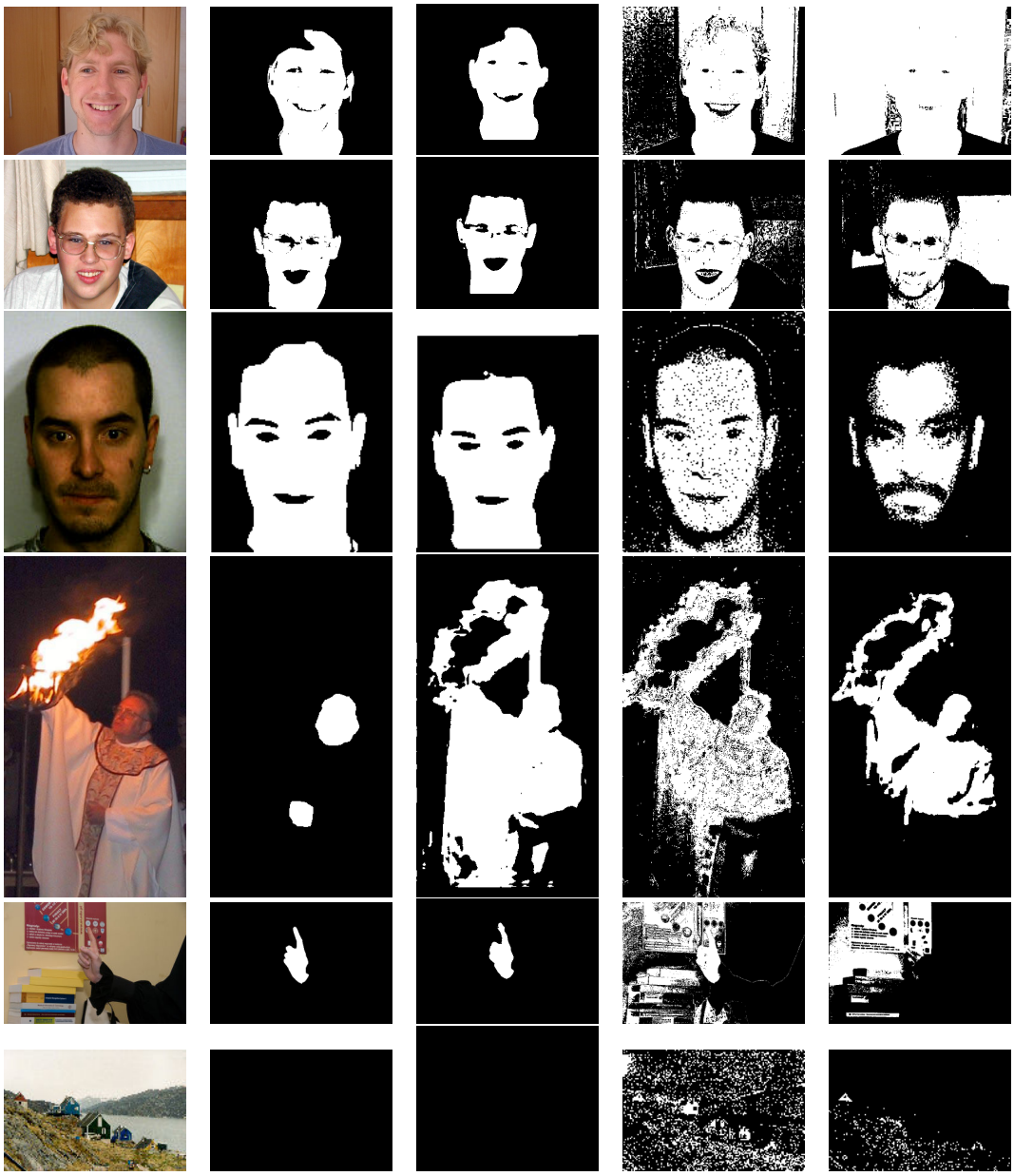
In the single evaluation of the datasets, Table 4.3, the deep-learning approach beats its competitors in all the measurements, while the statistical approach comes always second. The Schmugge dataset describes a higher difficulty of classification that can be attributed to the variety of the lighting conditions, subjects, and environments present in its pictures. The variety of the database can also be described by the high standard deviation measurements obtained. Furthermore, by considering the ambiguous regions as skin pixels, the performance of the classifier may be affected. The high D_{prs} scores obtained by the statistical and thresholding methods may indicate the presence of many True Negatives, as the metric is the only one between the chosen metrics that incorporates them. HGR seems to be the easier dataset to classify, which can be due to the relatively low diversity of subjects and backgrounds. In fact, learning approaches tend to have very high measurements. Tarasiewicz *et al.* 2020 [25] seems to perform very well on this dataset, achieving very low standard deviation scores. In the ECU dataset, the results of the histogram and thresholding approaches are relatively close, while the CNN outperforms them by far.

Some interesting instances can be seen in Figure 4.4. The first row shows the problem that having a background with similar colors to the skin represents in the color-based methods. The statistical method seems to act relatively better in this scenario compared to Brancati *et al.* 2017 [17], as seen in the second row. In the third row, the statistical approach classify some background and hair pixels as skin, while the rule-based method has many False Negatives, but the head remain still recognizable. The fourth row represent a very hard image to classify. Interestingly, the U-Net describes a very bad classifications, with a tremendous number of False Positives. The thresholding approach is the most restrictive on False Positives in this instance. The next-to-last image has an inconvenient background with skin-like colors and only Tarasiewicz *et al.* 2020 [25] tend to have a good classification. The statistical method manages to classify the skin pixels well, but has a really high number of False Positives. The last image is a landscape image without

skin pixels and once again the color-based approaches describe many False Positives, with the machine learning one having lots of them.

	Method\Database	ECU	HGR	Schmugge
$F_1 \uparrow$	Tarasiewicz *et al.* [25]	**0.9133 ± 0.08**	**0.9848 ± 0.02**	**0.6121 ± 0.45**
	Acharjee [53]	0.6980 ± 0.22	0.9000 ± 0.15	0.5098 ± 0.39
	Brancati *et al.* [17]	0.6356 ± 0.24	0.7362 ± 0.27	0.4280 ± 0.34
$IoU \uparrow$	Tarasiewicz *et al.* [25]	**0.8489 ± 0.12**	**0.9705 ± 0.03**	**0.5850 ± 0.44**
	Acharjee [53]	0.5751 ± 0.23	0.8434 ± 0.19	0.4303 ± 0.34
	Brancati *et al.* [17]	0.5088 ± 0.25	0.6467 ± 0.30	0.3323 ± 0.28
$D_{prs} \downarrow$	Tarasiewicz *et al.* [25]	**0.1333 ± 0.12**	**0.0251 ± 0.03**	**0.5520 ± 0.64**
	Acharjee [53]	0.4226 ± 0.27	0.1524 ± 0.19	0.7120 ± 0.54
	Brancati *et al.* [17]	0.5340 ± 0.32	0.3936 ± 0.36	0.8148 ± 0.48

Table 4.3: Single evaluation on datasets.
For each dataset: methods are eventually trained on the training set, in the case of trainable methods, and then predictions are performed on the test set.
For example, with ECU as the dataset, it means that a method is trained using the training set of ECU, if the method is trainable, and then tested on the test set of ECU.



(a) (b) (c) (d) (e)

Figure 4.4: Skin detection results in datasets single evaluation: (a) the input image; (b) the ground truth; (c) Tarasiewicz *et al.* 2020 [25]; (d) Acharjee [53]; (e) Brancati *et al.* 2017 [17]. In all the following results, Tarasiewicz *et al.* 2020 [25] predictions have different dimensions than other images due to the network preprocessing.

4.4 Performance across databases

In the cross evaluation of the datasets (present at Table 4.4), the deep learning approach still dominates, but there are some interesting exceptions. In fact, using HGR as the training set and predicting over Schmugge, Acharjee 2018 [53] outperforms Tarasiewicz *et al.* 2020 [25], especially in the F_1 score. This means that, while the statistical method generally performs better than Tarasiewicz *et al.* 2020 [25], it also includes a lot of False Positives, as the F_1-IoU and the D_{prs} metrics indicate. The latter is particularly bad in both cases, evidencing a big distance between the ideal ground truths and the predictions. In the case where ECU is used to train and Schmugge to predict, and in the opposite case, Tarasiewicz *et al.* 2020 [25] beats the statistical method discretely. In the latter case, the U-Net describes a slightly worse F_1-IoU, suggesting the presence of False Positives and False Negatives. Tarasiewicz *et al.* 2020 [25] outperforms the other machine learning approach by far in the remaining cases. The U-Net exceeds an F_1 score of 80 in the case of Schmugge as the training set and HGR as the prediction set despite the size of the training set, which is not huge.

	Training	ECU		HGR		SCHMUGGE	
	Testing	HGR	SCHMUGGE	ECU	SCHMUGGE	ECU	HGR
$F_1 \uparrow$	Taras. *et al.* [25]	**0.9308 ± 0.11**	**0.4625 ± 0.41**	**0.7252 ± 0.20**	0.2918 ± 0.31	**0.6133 ± 0.21**	**0.8106 ± 0.19**
	Acharjee [53]	0.5577 ± 0.29	0.3319 ± 0.28	0.4279 ± 0.19	**0.4000 ± 0.32**	0.4638 ± 0.23	0.5060 ± 0.25
$IoU \uparrow$	Taras. *et al.* [25]	**0.8851 ± 0.15**	**0.3986 ± 0.37**	**0.6038 ± 0.22**	0.2168 ± 0.25	**0.4754 ± 0.22**	**0.7191 ± 0.23**
	Acharjee [53]	0.4393 ± 0.27	0.2346 ± 0.21	0.2929 ± 0.17	**0.2981 ± 0.24**	0.3318 ± 0.20	0.3752 ± 0.22
$D_{prs} \downarrow$	Taras. *et al.* [25]	**0.1098 ± 0.15**	**0.7570 ± 0.56**	**0.3913 ± 0.26**	**0.9695 ± 0.44**	**0.5537 ± 0.27**	**0.2846 ± 0.27**
	Acharjee [53]	0.5701 ± 0.29	1.0477 ± 0.35	0.8830 ± 0.23	1.0219 ± 0.42	0.7542 ± 0.30	0.6523 ± 0.27
$F_1 - IoU \downarrow$	Taras. *et al.* [25]	**0.0457**	**0.0639**	**0.1214**	**0.0750**	0.1379	**0.0915**
	Acharjee [53]	0.1184	0.0973	0.1350	0.1019	**0.1320**	0.1308

Table 4.4: Cross evaluation on datasets.
For each dataset: methods are trained on the training set and then predictions are performed on all the images of every other datasets.
For example, with ECU as the training dataset and HGR as the testing dataset, it means that a method is trained using the training set of ECU, and then tested on all the HGR dataset.

Some interesting instances are shown in Figure 4.5. The expression "HGR on ECU" describes the situation in which the evaluation is performed by using HGR as the training set and ECU as the test set. In cross evaluations, this type of expression is going to be used a lot to avoid text redundancy. The first row (HGR on ECU) shows a really poor classification by the statistical approach. In the second row (ECU on Schmugge) it can be noticed how Acharjee 2018 [53] tends to exaggerate at classifying skin pixels in some cases, confirming the above intuitions on the statistical method having a lot of False Positives. The third row (ECU on Schmugge) describes a tragic classification by Tarasiewicz *et al.* 2020 [25], that might be caused by the presence of tricky lighting conditions and shadows

on the skin regions. The statistical approach performs badly too, but still has a lot better classification. In this case, the ground truth shows how incorporating the ambiguous regions provided by the Schmugge database makes the skin regions quite pixels-hungry on the border with non-skin regions. The fourth image (Schmugge on ECU) represents a bad classification in both approaches, where there is especially an outstanding number of False Positives. The next-to-last image (HGR on Schmugge) is part of the datasets combination in which Acharjee 2018 [53] outperforms the deep learning one. The statistical method reports a lot of False Positives, but also a lot of True Positives, which Tarasiewicz *et al.* 2020 [25] struggles to identify. The last picture (HGR on Schmugge) is also part of the same datasets combination and describes a similar situation: the U-Net fails at labeling several skin pixels, especially on very lit regions, while the statistical method overdoes it. This image represents the high complexity and diversity of the Schmugge content.

Figure 4.5: Skin detection results in datasets cross evaluation: (a) the input image; (b) the ground truth; (c) Tarasiewicz *et al.* 2020 [25]; (d) Acharjee [53]

4.5 Performance on single Skin tones

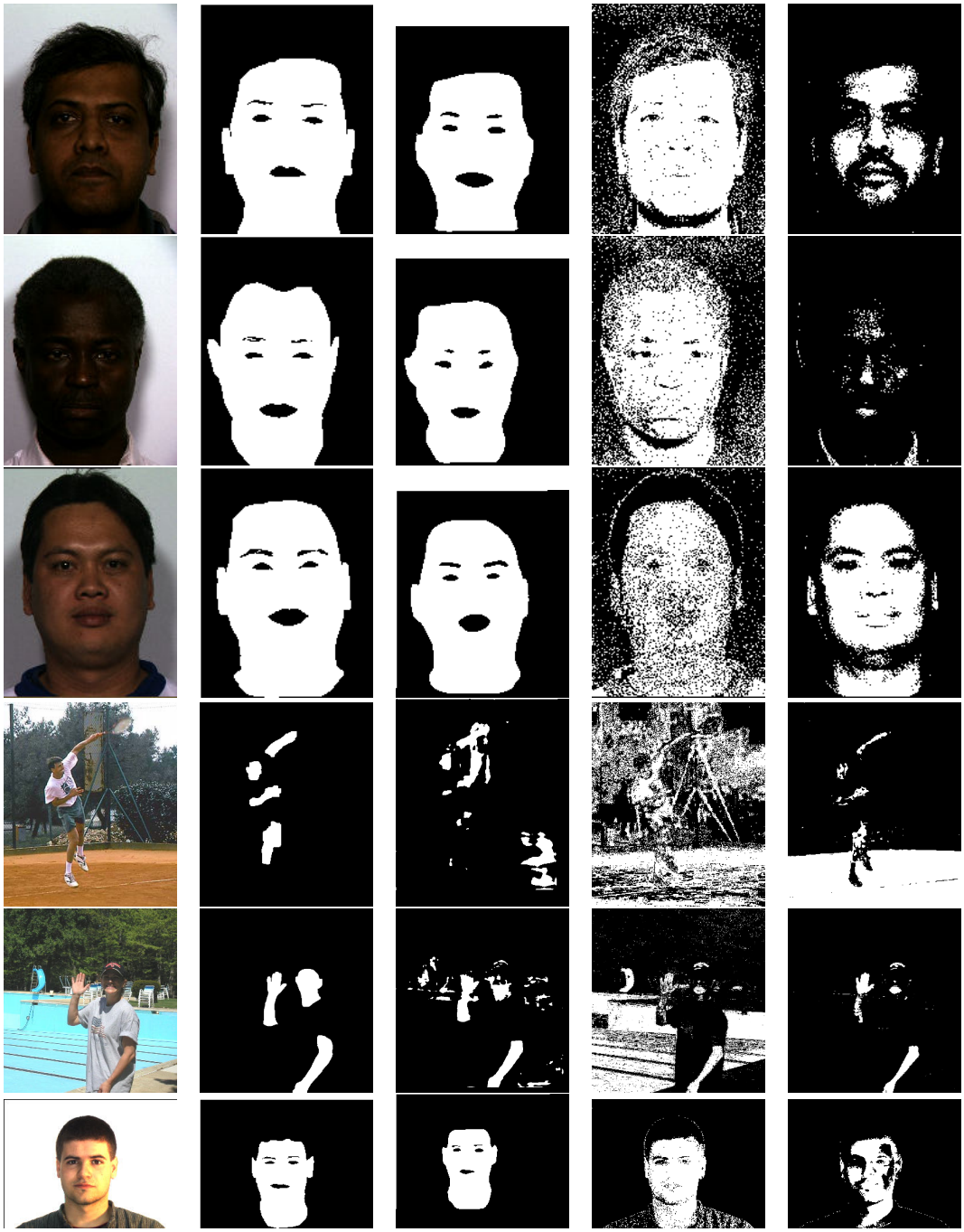
In the single evaluation of the skin tones sub-datasets (present at Table 4.5), the U-Net outperforms the competitors in all cases, and the thresholding approach comes last in all cases. The learning approaches describe a very low standard deviation on the darker skin tones, indicating that the diversity of the images might not be very high. Tarasiewicz *et al.* 2020 [25] reports outstanding scores, having almost ground truth-like predictions, as described by the D_{prs} measure. Brancati *et al.* 2017 [17] instead describes horrible results, which may indicate that the skin clustering rules are leaving out the darker skin pixels. The machine learning methods have the highest difficulty at classifying the medium skin tones, which pictures include a lot of difficult scenarios, such as clay terrains that are skin-like colored. Despite the troubles, all the approaches report generally good scores on both medium and light skin tones, especially the machine learning methods.

	Method\Database	DARK	MEDIUM	LIGHT
$F_1 \uparrow$	Tarasiewicz *et al.* [25]	$\mathbf{0.9529 \pm 0.00}$	$\mathbf{0.9260 \pm 0.15}$	$\mathbf{0.9387 \pm 0.12}$
	Acharjee [53]	0.8123 ± 0.02	0.7634 ± 0.19	0.8001 ± 0.15
	Brancati *et al.* [17]	0.2620 ± 0.14	0.6316 ± 0.20	0.6705 ± 0.14
$IoU \uparrow$	Tarasiewicz *et al.* [25]	$\mathbf{0.9100 \pm 0.01}$	$\mathbf{0.8883 \pm 0.18}$	$\mathbf{0.9006 \pm 0.14}$
	Acharjee [53]	0.6844 ± 0.03	0.6432 ± 0.17	0.6870 ± 0.16
	Brancati *et al.* [17]	0.1587 ± 0.10	0.4889 ± 0.19	0.5190 ± 0.14
$D_{prs} \downarrow$	Tarasiewicz *et al.* [25]	$\mathbf{0.0720 \pm 0.01}$	$\mathbf{0.1078 \pm 0.21}$	$\mathbf{0.0926 \pm 0.15}$
	Acharjee [53]	0.3406 ± 0.05	0.3452 ± 0.23	0.3054 ± 0.20
	Brancati *et al.* [17]	0.8548 ± 0.12	0.5155 ± 0.24	0.4787 ± 0.17

Table 4.5: Single evaluation on skin tones.
For each dataset: methods are eventually trained on the training set, in the case of trainable methods, and then predictions are performed on the test set.
For example, with DARK as the dataset, it means that a method is trained using the training set of DARK, if the method is trainable, and then tested on the test set of DARK.

Some interesting instances can be seen in Figure 4.6. The first two rows depict darker skin tones. In both examples, it is possible to notice a pattern in the classification of each approach: Tarasiewicz *et al.* 2020 [25] produces almost ground truth-like predictions; the statistical method tends to exaggerate on classifying skin pixels, but has an excellent number of True Positives; Brancati *et al.* 2017 [17] seems to fail at classifying the darkest skin tones. The next two rows represent medium skin tones. The following row reports similar results to the dark skin tones rows, but in this case the thresholding approach has good results. It is noticeable that, while the statistical method tends to include many

False Positives, the thresholding one is a lot more conservative, marking only the inner regions of the face, which are sufficient for describing the face shape. The last row of medium skin tones represents a tricky background with a clay terrain. Tarasiewicz *et al.* 2020 [25] produces a very good prediction, while the other approaches include many False Positives. Acharjee 2018 [53] reports a tremendous number of False Positives, while Brancati *et al.* 2017 [17] is deceived by the clay terrain and ruins its otherwise excellent classification. The last two rows feature people with light skin tones. In the starting row, the U-Net and the rule-based approaches have very good predictions, with Tarasiewicz *et al.* 2020 [25] incorporating more False Positives, and Brancati *et al.* 2017 [17] including more False Negatives. The statistical approach reports once again a huge number of False Positives. The last row depicts good results on all methods and shows some limitations that Brancati *et al.* 2017 [17] seems to have on tricky lighting conditions.



(a) (b) (c) (d) (e)

Figure 4.6: Skin detection results in skin tones single evaluation: (a) the input image; (b) the ground truth; (c) Tarasiewicz *et al.* 2020 [25]; (d) Acharjee [53]; (e) Brancati *et al.* 2017 [17].

4.6 Performance across Skin tones

The cross evaluation of the skin tones sub-datasets, Table 4.6), describes some interesting situations. As usual, the U-Net outperforms the other machine learning approach in most situations, but it is outperformed a pair of times. In fact, using the darker skin tones as training and predicting on the medium skin tones makes Acharjee 2018 [53] the seemingly better classifier. It describes best scores in all metrics but the difference between F_1 and D_{prs}, with also a lower population standard deviation. The dark sub-dataset was the smallest one and therefore has been data-augmented. However, the augmentation has been performed with light transformations that do not detach from the original images too much. Hence, these scores may indicate that Acharjee 2018 [53] performs better with a smaller training set. Anyway, Tarasiewicz *et al.* 2020 [25] describes only slightly worse results and reports good classifications in this scenario too. The higher difference between F_1 and D_{prs} may indicate the usual greedy that the statistical method demonstrates on False Positives. The dark on light case is particularly interesting because the statistical approach wins on the F_1-Score, but is outperformed in all the others. Winning on the F_1-Score and losing on the IoU means that the statistical approach picks more True Positives than the deep learning one. Tarasiewicz *et al.* 2020 [25] also describes more unstable results as the population standard deviation is higher. The F_1 and IoU scores are relatively close between the approaches, but the D_{prs} is far better with Tarasiewicz *et al.* 2020 [25]. This behavior may suggest, along with the F_1 and IoU considerations, and the fact that their difference is higher on the statistical method, the presence of many False Positives and/or False Negatives, which however seems unlikely given the previous observations, on the statistical method predictions. All the other situations have the U-Net winning on the scores by a great degree. In the medium on dark situation, Acharjee 2018 [53] describes a very high D_{prs}, even higher than the light on dark case, where the F_1 and IoU scores are worse. This is a clear indicator that the Specificity is driving the prediction away from the ideal ground truth, so it may be an indicator of a very small number of True Negatives. Tarasiewicz *et al.* 2020 [25] presents the highest score in the light on medium case, which may be a signal of the need of having bigger training data, but on the other hand, it performs far worse on the light on dark case, indicating that a bigger training data might not be sufficient. In fact, the U-Net describes the best average score by training on the medium skin tones, which represent a smaller dataset than the lighter skin tones, but a midpoint between the colors of darker and lighter skin tones. This may be a representation of how the Neural Network adapts and learns to classify new skin tones.

Some interesting instances are reported in Figure 4.7. The first two rows represent dark on medium cases. The results seem to confirm the earlier guesses on the statistical method incorporating a lot of False Positives. Tarasiewicz *et al.* 2020 [25] depicts a terrible classification, with a lot of False Positives and False Negatives. In the second row, the tricky lighting causes troubles to the U-Net, which provides another terrible predic-

tion, while the statistical method has a far better prediction, despite including a lot of False Positives too. The third row and the fourth row represent dark on light situations. Tarasiewicz *et al.* 2020 [25] produces an awful prediction even in these cases, featuring particularly a lot of False Negatives. Acharjee 2018 [53] does the opposite and once again exaggerates as usual with the classification of skin pixels. The original images of these two rows are also featured in Figure 4.5, where they reported similar issues, so it seems that both approaches struggle to adapt to skin pixels different from the ones of the training set in these types of images. The smallest image depicts a quite extreme and unnatural level of lighting, hence the issues are easily justifiable. The other image seems to depict a more natural skin tone, but it must be taken into account that it may still be far from the skin tones of the training set, which may be too light or dark. The fifth row features a medium on dark case, where the hypothesis of having very few True Negatives, driving the D_{prs} measure high, seems confirmed. Tarasiewicz *et al.* 2020 [25] on the other hand performs a quite good classification, marking almost correctly most of the skin regions. The last row represents a light on dark case on the same original picture of the previous row, with the purpose of a quick comparison. In this case, the statistical approach does a much better job, especially at predicting non-skin pixels, which may indicate that the light sub-dataset contains more images featuring sky and water labeled as non-skin pixels. Tarasiewicz *et al.* 2020 [25] performs quite similarly to the previous instance.

	Training	DARK		MEDIUM		LIGHT	
	Testing	MEDIUM	LIGHT	DARK	LIGHT	DARK	MEDIUM
$F_1 \uparrow$	Taras. *et al.* [25]	0.7300 ± 0.25	0.7262 ± 0.26	$\mathbf{0.8447 \pm 0.13}$	$\mathbf{0.8904 \pm 0.14}$	$\mathbf{0.7660 \pm 0.17}$	$\mathbf{0.9229 \pm 0.11}$
	Acharjee [53]	$\mathbf{0.7928 \pm 0.11}$	$\mathbf{0.7577 \pm 0.12}$	0.5628 ± 0.14	0.7032 ± 0.14	0.5293 ± 0.20	0.7853 ± 0.11
$IoU \uparrow$	Taras. *et al.* [25]	0.6279 ± 0.27	$\mathbf{0.6276 \pm 0.28}$	$\mathbf{0.7486 \pm 0.15}$	$\mathbf{0.8214 \pm 0.16}$	$\mathbf{0.6496 \pm 0.21}$	$\mathbf{0.8705 \pm 0.13}$
	Acharjee [53]	$\mathbf{0.6668 \pm 0.11}$	0.6229 ± 0.13	0.4042 ± 0.13	0.5571 ± 0.14	0.3852 ± 0.19	0.6574 ± 0.12
$D_{prs} \downarrow$	Taras. *et al.* [25]	0.3805 ± 0.33	$\mathbf{0.3934 \pm 0.34}$	$\mathbf{0.2326 \pm 0.17}$	$\mathbf{0.1692 \pm 0.18}$	$\mathbf{0.3402 \pm 0.21}$	$\mathbf{0.1192 \pm 0.16}$
	Acharjee [53]	$\mathbf{0.3481 \pm 0.16}$	0.4679 ± 0.18	0.6802 ± 0.20	0.5376 ± 0.23	0.6361 ± 0.22	0.3199 ± 0.16
$F_1 - IoU \downarrow$	Taras. *et al.* [25]	**0.1021**	**0.0986**	**0.0961**	**0.0690**	**0.1164**	**0.0524**
	Acharjee [53]	0.1260	0.1348	0.1586	0.1461	0.1441	0.1279

Table 4.6: Cross evaluation on skin tones.
For each dataset: methods are trained on the training set and then predictions are performed on all the images of every other datasets.
For example, with DARK as the training dataset and LIGHT as the testing dataset, it means that a method is trained using the training set of DARK, and then tested on all the LIGHT dataset.

(a) (b) (c) (d)

Figure 4.7: Skin detection results in skin tones cross evaluation: (a) the input image; (b) the ground truth; (c) Tarasiewicz *et al.* 2020 [25]; (d) Acharjee [53].

4.7 Inference time

The inference time evaluation follows these rules:

- Image loading into memory is excluded.
- Image saving to disk is excluded.
- The measurement starts when the algorithm starts.
- Pre-processing and post-processing, if present, are included in the measured execution time.

It has been defined a set of 14 images of the same dimensions to use as the performance evaluation set: it is composed of the first 14 images of the ECU dataset of size 352×288. One image at a time has been processed by the methods and the resulting execution time has been saved. The set of pictures has been processed 5 times and, each time, the average measurement time has been calculated. Finally, the five average values have been averaged into a single value and the standard deviation has been computed.
The inference time measurements have all been performed on an i7 4770k processor running on Pop!_OS 20.10 x86_64 with 16 GB of RAM. The rule-based [17] method uses OpenCV 2.4.9 and is compiled with g++ version 10.2.0-13ubuntu1. The other methods use Python 3.8.6 64bit. Tarasiewicz *et al.* 2020 [25] uses Tensorflow 2.5.0 for inference time measurements, but the models has been trained using Google Colab with Tensorflow 2.4.1 and Python 3.7.10.

The inference time evaluation (found in Table 4.7) reports a very simple situation. The thresholding approach is much faster than the other approaches and also expresses a null standard deviation, which highlights the impartiality that the algorithm describes with respect to images of different content. The statistical approach comes second and reports a really low standard deviation score too. Despite coming second, its score is far away from the first place. At the last place lies Tarasiewicz *et al.* 2020 [25], with a score of almost one second per image and the biggest standard deviation score, which may indicate that its inference times depend slightly on the pixels of an image. The scores indicate that the only suitable approach to real-time skin detection on a CPU may be Brancati *et al.* 2017 [17], which is very fast and could describe the same behavior on less powerful hardware.

	Inference time (seconds)
Tarasiewicz *et al.* [25]	0.826581 ± 0.043
Acharjee [53]	$\underline{0.457534 \pm 0.002}$
Brancati *et al.* [17]	$\mathbf{0.007717 \pm 0.000}$

Table 4.7: Inference time performance measurements.

5 Conclusion

In this thesis the significance and limitations of skin detection have been addressed. A review of public datasets available in the domain and an analysis of state-of-the-art approaches has been presented, including a new proposed taxonomy. Three different state-of-the-art methods have been thoroughly examined implemented and validated in respect to the original papers, when possible. An evaluation of the chosen approaches in different settings has been presented, alongside a discussion on the metrics used in the domain. Finally, the results have been thoroughly discussed through data and figures.

5.1 Discussion

The analysis of the evaluation results has pointed out the strengths and weaknesses of each of the chosen approaches.
The thresholding method reported the worst classification scores but described inference times dozens of times faster on the CPU. Moreover, its prediction time resulted independent of image pixels. The prediction is inaccurate when the images have dark skin tones and backgrounds with skin-like colors.
The statistical approach reported better results, but was prone to several False Positives. Being color-based, it also shared some limitations with the thresholding approach: materials with skin-like colors have represented a challenge to classify.
The U-Net approach has always reported the best scores. It demonstrated how deep learning is able to find features other than color to be able to classify skin pixels even in tricky situations. It struggled to generalize in two cases: when the training dataset was very small and when the training data was too different from the test data.
The involvement of more balanced metrics has proven to be essential to clarify situations where other metrics described over-optimistic results.

5.2 Future Work

Skin detection approaches have evolved in multiple paths, covering aspects such as classification performance and computational efficiency. However, skin detection datasets still represent a major limitation in the development and evaluation of skin detectors. New datasets are still highly sought after and could represent a significant boost in the skin detection domain.

In future evaluations, it should be taken into account the metric behavior and its limitations. Metrics that describe a more stable behavior on unbalanced datasets can be involved to represent results in a more complete way.

Finally, the development of skin detectors could benefit from the progress that image segmentation is having with deep learning, especially in the medical field, which often features binary classification problems.

For achieving better classifications, Transformers could be considered, as they have proven to be really solid in Natural Language Processing [64], and are starting to gain traction in the image segmentation tasks, so much that some U-Net-like architectures have been recently designed [65, 66].

Regarding performance and computational powers, skin detectors could venture into mobile deep learning development. In fact, mobile phones, are starting to become a solid platform for U-Nets [67].

Bibliography

1. B. Fink, K. Grammer, and P. J. Matts. "Visible skin color distribution plays a role in the perception of age, attractiveness, and health in female faces". *Evolution and Human Behavior* 27:6, 2006, pp. 433–442.
2. S. E. Choi, Y. J. Lee, S. J. Lee, K. R. Park, and J. Kim. "Age estimation using a hierarchical classifier based on global and local facial features". *Pattern recognition* 44:6, 2011, pp. 1262–1281.
3. G. A. Ramirez, O. Fuentes, S. L. Crites Jr, M. Jimenez, and J. Ordonez. "Color analysis of facial skin: Detection of emotional state". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 468–473.
4. S. Baskan, M. M. Bulut, and V. Atalay. "Projection based method for segmentation of human face and its evaluation". *Pattern Recognition Letters* 23:14, 2002, pp. 1623–1629.
5. J. Singha, A. Roy, and R. H. Laskar. "Dynamic hand gesture recognition using vision-based approach for human–computer interaction". *Neural Computing and Applications* 29:4, 2018, pp. 1129–1141.
6. Y.-H. Chen, K.-T. Hu, and S.-J. Ruan. "Statistical skin color detection method without color transformation for real-time surveillance systems". *Engineering Applications of Artificial Intelligence* 25:7, 2012, pp. 1331–1337.
7. A. Shifa, M. B. Imtiaz, M. N. Asghar, and M. Fleury. "Skin detection and lightweight encryption for privacy protection in real-time surveillance applications". *Image and Vision Computing* 94, 2020, p. 103859.
8. M. J. Jones and J. M. Rehg. "Statistical color models with application to skin detection". *International journal of computer vision* 46:1, 2002, pp. 81–96.
9. J. Stöttinger, A. Hanbury, C. Liensberger, and R. Khan. "Skin paths for contextual flagging adult videos". In: *International symposium on visual computing*. Springer. 2009, pp. 303–314.
10. Q. Zhu, C.-T. Wu, K.-T. Cheng, and Y.-L. Wu. "An adaptive skin model and its application to objectionable image filtering". In: *Proceedings of the 12th annual ACM international conference on Multimedia*. 2004, pp. 56–63.

11. Y. Liu, Z. G. Li, and Y. C. Soh. "Region-of-interest based resource allocation for conversational video communication of H. 264/AVC". *IEEE transactions on circuits and systems for video technology* 18:1, 2008, pp. 134–139.
12. C.-C. Low, L.-Y. Ong, V.-C. Koo, and M.-C. Leow. "Multi-audience tracking with RGB-D camera on digital signage". *Heliyon* 6:9, 2020, e05107.
13. T.-T. Do, Y. Zhou, H. Zheng, N.-M. Cheung, and D. Koh. "Early melanoma diagnosis with mobile imaging". In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2014, pp. 6752–6757.
14. O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
15. B. D. Zarit, B. J. Super, and F. K. Quek. "Comparison of five color models in skin pixel classification". In: *Proceedings International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems. In Conjunction with ICCV'99 (Cat. No. PR00378)*. IEEE. 1999, pp. 58–63.
16. W. R. Tan, C. S. Chan, P. Yogarajah, and J. Condell. "A fusion approach for efficient human skin detection". *IEEE Transactions on Industrial Informatics* 8:1, 2011, pp. 138–147.
17. N. Brancati, G. De Pietro, M. Frucci, and L. Gallo. "Human skin detection through correlation rules between the YCb and YCr subspaces based on dynamic color clustering". *Computer Vision and Image Understanding* 155, 2017, pp. 33–42.
18. S. Naji, H. A. Jalab, and S. A. Kareem. "A survey on skin detection in colored images". *Artificial Intelligence Review* 52:2, 2019, pp. 1041–1087.
19. J. Kovac, P. Peer, and F. Solina. *Human skin color clustering for face detection*. Vol. 2. IEEE, 2003.
20. W.-C. Chen and M.-S. Wang. "Region-based and content adaptive skin detection in color images". *International journal of pattern recognition and artificial intelligence* 21:05, 2007, pp. 831–853.
21. R. Khan, A. Hanbury, and J. Stoettinger. "Skin detection: A random forest approach". In: *2010 IEEE International Conference on Image Processing*. IEEE. 2010, pp. 4613–4616.
22. M. S. Iraji and A. Yavari. "Skin color segmentation in fuzzy YCBCR color space with the mamdani inference". *American journal of scientific research* 2011:7, 2011, pp. 131–137.

23. M. Kawulok, J. Kawulok, J. Nalepa, and B. Smolka. "Self-adaptive algorithm for segmenting skin regions". *EURASIP Journal on Advances in Signal Processing* 2014:170, 2014, pp. 1–22. ISSN: 1687-6180. DOI: 10.1186/1687-6180-2014-170. URL: http://asp.eurasipjournals.com/content/2014/1/170.

24. Y. He, J. Shi, C. Wang, H. Huang, J. Liu, G. Li, R. Liu, and J. Wang. "Semi-supervised skin detection by network with mutual guidance". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2111–2120.

25. T. Tarasiewicz, J. Nalepa, and M. Kawulok. "Skinny: A Lightweight U-net for Skin Detection and Segmentation". In: *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2020, pp. 2386–2390.

26. S. L. Phung, A. Bouzerdoum, and D. Chai. "Skin segmentation using color pixel classification: analysis and comparison". *IEEE transactions on pattern analysis and machine intelligence* 27:1, 2005, pp. 148–154.

27. S. J. Schmugge, S. Jayaram, M. C. Shin, and L. V. Tsap. "Objective evaluation of approaches of skin detection using ROC analysis". *Computer vision and image understanding* 108:1-2, 2007, pp. 41–51.

28. M. Z. Osman, M. A. Maarof, and M. F. Rohani. "Improved dynamic threshold method for skin colour detection using multi-colour space". *Am. J. Appl. Sci* 13, 2016, pp. 135–144.

29. J. C. Sanmiguel and S. Suja. "Skin detection by dual maximization of detectors agreement for video monitoring". *Pattern Recognition Letters* 34:16, 2013, pp. 2102–2109.

30. J. P. B. Casati, D. R. Moraes, and E. L. L. Rodrigues. "SFA: A human skin image database based on FERET and AR facial images". In: *IX workshop de Visao Computational, Rio de Janeiro*. 2013.

31. A. Topiwala, L. Al-Zogbi, T. Fleiter, and A. Krieger. "Adaptation and evaluation of deep learning techniques for skin segmentation on novel abdominal dataset". In: *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)*. IEEE. 2019, pp. 752–759.

32. M. R. Mahmoodi, S. M. Sayedi, F. Karimi, Z. Fahimi, V. Rezai, and Z. Mannani. "SDD: A skin detection dataset for training and assessment of human skin classifiers". In: *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*. IEEE. 2015, pp. 71–77.

33. J. Ruiz-del-Solar and R. Verschae. "SKINDIFF-Robust and fast skin segmentation". *Department of Electrical Engineering, Universidad de Chile*, 2006.

34. A. Dourado, F. Guth, T. E. de Campos, and L. Weigang. "Domain adaptation for holistic skin detection". *arXiv preprint arXiv:1903.06969*, 2019.

35. R. B. Bhatt, G. Sharma, A. Dhall, and S. Chaudhury. "Efficient skin region segmentation using low complexity fuzzy decision tree model". In: *2009 Annual IEEE India Conference*. IEEE. 2009, pp. 1–4.

36. D. Chicco and G. Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". *BMC genomics* 21:1, 2020, pp. 1–13.

37. S. Bianco, F. Gasparini, and R. Schettini. "Adaptive skin classification using face and body detection". *IEEE Transactions on Image Processing* 24:12, 2015, pp. 4756–4765.

38. A. Martinez and R. Benavente. "The AR Face Database: CVC Technical Report, 24", 1998.

39. E. A. Marszalec, J. B. Martinkauppi, M. N. Soriano, and M. Pietikaeinen. "Physics-based face database for color research". *Journal of Electronic Imaging* 9:1, 2000, pp. 32–38.

40. A. P. Berman and L. G. Shapiro. "A flexible image database system for content-based retrieval". *Computer Vision and Image Understanding* 75:1-2, 1999, pp. 175–195.

41. D. Dharmaretnam and A. Fyshe. "The emergence of semantics in neural network representations of visual information". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, pp. 776–780.

42. J. C. Russ. *The image processing handbook.* English. Boca Raton, FL: CRC Press, 2007, p. 817. ISBN: 978-0-8493-7254-4.

43. N. A. bin Abdul Rahman, K. C. Wei, and J. See. "Rgb-h-cbcr skin colour model for human face detection". *Faculty of Information Technology, Multimedia University* 4, 2007.

44. A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt. "A skin tone detection algorithm for an adaptive approach to steganography". *Signal Processing* 89:12, 2009, pp. 2465–2478.

45. C. Garcia and G. Tziritas. "Face detection using quantized skin color regions merging and wavelet packet analysis". *IEEE Transactions on multimedia* 1:3, 1999, pp. 264–277.

46. H.-C. Do, J.-Y. You, and S.-I. Chien. "Skin color detection through estimation and conversion of illuminant color under various illuminations". *IEEE Transactions on Consumer Electronics* 53:3, 2007, pp. 1103–1108.

47. H. Yao and W. Gao. "Face detection and location based on skin chrominance and lip chrominance transformation from color images". *Pattern recognition* 34:8, 2001, pp. 1555–1564.

48. P. Vadakkepat, P. Lim, L. C. De Silva, L. Jing, and L. L. Ling. "Multimodal approach to human-face detection and tracking". *IEEE transactions on industrial electronics* 55:3, 2008, pp. 1385–1393.

49. R. I.-R. BT et al. "Studio encoding parameters of digital television for standard 4: 3 and wide-screen 16: 9 aspect ratios", 2011.

50. S. L. Phung, D. Chai, and A. Bouzerdoum. "A universal and robust human skin color model using neural networks". In: *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*. Vol. 4. IEEE. 2001, pp. 2844–2849.

51. R. A. Redner and H. F. Walker. "Mixture densities, maximum likelihood and the EM algorithm". *SIAM review* 26:2, 1984, pp. 195–239.

52. G. Gomez. "On selecting colour components for skin detection". In: *Object recognition supported by user interaction for service robots*. Vol. 2. IEEE. 2002, pp. 961–964.

53. C. Acharjee. *Skin-detection*. `https://github.com/Chinmoy007/Skin-detection`. 2018 (accessed: 14 May 2021).

54. A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems* 25, 2012, pp. 1097–1105.

55. J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

56. A. Silburt, M. Ali-Dib, C. Zhu, A. Jackson, D. Valencia, Y. Kissin, D. Tamayo, and K. Menou. "Lunar crater identification via deep learning". *Icarus* 317, 2019, pp. 27–38.

57. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

58. G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

59. S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

60. J. M. Austin and A. Kachalia. "The Need for Standardized Metrics to Drive Decision-making During the COVID-19 Pandemic". *Journal of hospital medicine* 16:1, 2021, pp. 56–58.

61. K. Intawong, M. Scuturici, and S. Miguet. "A new pixel-based quality measure for segmentation algorithms integrating precision, recall and specificity". In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2013, pp. 188–195.

62. A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. "Albumentations: fast and flexible image augmentations". *Information* 11:2, 2020, p. 125.

63. D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*, 2014.

64. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

65. H. Cao, Y. Wang, J. Chen, D. Jiang, X. Zhang, Q. Tian, and M. Wang. "Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation". *arXiv preprint arXiv:2105.05537*, 2021.

66. J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou. "Transunet: Transformers make strong encoders for medical image segmentation". *arXiv preprint arXiv:2102.04306*, 2021.

67. A. Ignatov, K. Byeoung-su, R. Timofte, and A. Pouget. "Fast camera image denoising on mobile gpus with deep learning, mobile ai 2021 challenge: Report". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2515–2524.