A machine learning approach for jaundice detection using color corrected smartphone images

Håvard Hagen Falk
Oliver Damsgaard Jensen

Master of Science in Computer Science
Submission date: June 2018
Supervisor: Frank Lindseth, IDI
Co-supervisor: Mahdi Amani, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

Neonatal jaundice and the associated yellow skin discoloration is caused by hyperbilirubinemia, a condition in which there is too much bilirubin in the blood. In 2010, the failure to treat jaundice resulted in 114,000 avoidable infant deaths and 75,000 children with brain dysfunction worldwide. Studies have found hyperbilirubinemia to be one of the top three causes of death among newborns in sub-Saharan Africa, and affordable jaundice detection technologies are urgently needed.

Transcutaneous bilirubinometers are robust screening methods commonly used by the medical community, but the price tag limits their widespread availability. Using smartphone cameras to capture skin induced color values, we create a low-cost alternative to expensive bilirubinometers. Through computer vision and deep learning, the final fully-connected feed-forward neural network achieves competitive results compared to Dräger JM-103, a state-of-the-art bilirubinometer. In the critical cases where treatment is required, the neural network even outperforms Dräger JM-103, resulting in 75% fewer false negative predictions.

Contrary to the transcutaneous bilirubinometers, the proposed solution relies on images, where light sources and camera sensors drastically impact the skin information. To make images illumination and camera independent, we propose a novel approach for color correction, the Gaussian process regression (GPR), a machine learning model that adapts to environmental variables. The results show that the GPR achieves equal results to state-of-the-art color correction techniques, while also creating more general models.

Sammendrag

Spedbarns gulsott og den gule misfargingen av huden skyldes hyperbilirubinemi, en tilstand der blodet inneholder økte mengder av fargestoffet bilirubin. I 2010 resulterte manglende behandling av gulsott i 114 000 unødvendige spedbarns dødsfall og 75 000 barn med hjerneskader totalt i verden. Studier viser at hyperbilirubinemi er en av de tre største dødsårsakene blant nyfødte i Afrika syd for Sahara, og at det er et akutt behov for rimelige teknologier som kan oppdage gulsott.

Transkutane bilirubinometre er en pålitelig screening metode som ofte er brukt av det medisinske miljøet, men som dessverre er lite tilgjengelige grunnet høy pris. Ved å bruke smarttelefonkameraer til å ta bilder av hud, har vi laget et billig alternativ til de dyre bilirubinometrene. Vi tar i bruk datasyn og maskinlæring i form av et dypt nevralt nettverk og oppnår konkurransedyktige resultater sammenliknet med en Dräger JM-103, et toppmoderne bilirubinometer. I kritiske tilfeller hvor behandling er nødvendig, overgår det nevrale nettverket Dräger JM-103, noe som resulterer i 75% færre falskt negative prediksjoner.

I motsetning til de transkutane bilirubinometrene, er den foreslåtte løsningen avhengig av bilder, hvor lyskilder og kamerasensorer drastisk påvirker hudinformasjonen. For å få lys og kamera uavhengige bilder, foreslår vi en ny tilnærming til fargekorrigering Gaussian process regression (GPR), en maskininnlæringsmodell som tilpasser seg bildetakningsforholdene. Resultatene viser at GPR oppnår like gode resultater som dagens beste fargekorrigeringsteknikker, samtidig som den lager mer generelle fargekorrigeringsmodeller.

Preface

This thesis was written over the course of the spring semester 2018, for the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU).

Foremost, we would like to express our sincere gratitude to our supervisor, associate professor Frank Lindseth, for giving us the opportunity to work on this exciting project. His help and guidance over the past months gave us motivation to finish the thesis and steered us in the right the direction.

We would also like to thank our co-supervisors Mahdi Amadi for all the encouragement, insightful comments, and hard questions. Without his passionate participation and input, the thesis would not be as detailed as it is today. It was a pleasure working with you.

We thank Anders Aune and Gunnar Vartdal from Picterus AS for allowing us to use their jaundice dataset and to work on this project. It has truly been a great experience and a lot of fun. Thank you for providing us with all necessary information and letting us tap into your medical expertise.

Lastly, we owe a deep thanks to Gunnar Vartdal from Picterus AS for the idea of using the Gaussian process regression for color correction.

Trondheim, June 7. 2018

Håvard Hagen FALK and **Oliver Damsgaard JENSEN**

Contents

List of Tables

List of Figures

Abbreviations

AI artificial intelligence. 23, 31, 34

ANN artificial neural network. 22, 23, 30, 31, 34, 54, 55

CNN convolutional neural network. 34, 35, 52, 57, 71, 77, 78, 87, 92, 93, 97

GP Gaussian process. 23–27, 45, 46

GPR Gaussian process regression. i, iii, 3, 42, 45, 46, 48, 50, 62, 65, 67–69, 81, 85–87, 95

LCC linear color correction. 20–22, 43, 44, 48, 62, 65, 67, 69, 81–83, 86

PCC polynomial color correction. 21, 22, 42–44, 85

PCCF polynomial color correction framework. 42, 48, 62, 65–70, 83, 84, 86, 87

RBF radial-basis function. 25, 27, 28, 53, 71, 85, 91

ReLu Rectified Linear Unit. 32, 35

RPCC root-polynomial color correction. 22, 43, 44, 83, 96

RPCCF root-polynomial color correction framework. 42, 44, 48, 62, 65, 67–70, 83–87, 96

RQ rational quadratic. 25, 27, 46

SGD Stochastic Gradient Decent. 32, 33

sRGB standard RGB. 14, 18

SVM Support Vector Machine. 22, 23, 27–29, 52, 90

SVR support vector regression. 29, 30, 52–54, 71, 72, 75, 87, 90–92

tanh The hyperbolic tangent. 32

TcB transcutaneous bilirubin. 2, 3, 10, 11, 41, 52, 72, 74, 93, 95

TSB total serum bilirubin. 2, 9–12, 40, 41, 49–52, 71, 72, 74, 87–90, 92, 93, 95, 96

Chapter 1

Introduction

1.1 Motivation

Neonatal jaundice refers to the yellow skin discoloration caused by bilirubin, a waste product formed during the breakdown of heme (i.e., the oxygen-carrying components in blood). The condition is developed by 60 - 80% of all newborns, usually within the first 48 hours of birth. In most cases, jaundice does not require treatment and newborns rid themselves of bilirubin after one or two weeks. If, however, the condition is severe and goes untreated, permanent damage can be inflicted to the brain, and may result in brain dysfunction or even death. To prevent this; systematic monitoring of all newborns is employed during birth hospitalization in most industrialized countries.

A population study performed by K Bhutani et al. (2013), across 184 countries, estimated the scope and consequences of jaundice for 2010. They concluded that failure to treat jaundice resulted in 114,000 avoidable infant deaths and 75,000 children growing up with brain dysfunction (kernicterus). Additionally, Slusher et al. (2011) have found jaundice to be one of the top three leading causes of death among newborns in sub-Saharan Africa. They state that appropriate jaundice detection technologies are urgently needed and that health care providers worldwide recognize jaundice as a "silent" cause of significant neonatal morbidity and mortality.

The yellow skin discoloration associated with jaundice is attributed to excess bilirubin. Visual examinations have, for a long time, been used to screen patients and are still important in clinical situations. Visual features are, however, subjective to the observer, and objectively quantifiable methods to support optical diagnosis are frequently used in the medical community. A blood sample can be analyzed to precisely determine the exact bilirubin concentration of a patient, total serum bilirubin (TSB) for short, but requires equipment and is expensive.

Bilirubin has a yellow color due to its absorption of green and blue light. These absorption properties are, today, used to estimate bilirubin concentration via skin-reflectance measurements. The measurements, called transcutaneous bilirubin (TcB), are performed by bilirubinometers, apparatus that emit light waves onto skin and, with sensors, record the amount of reflected light. Studies show that TcB correlates linearly with TSB and is therefore often used as a screening process, reducing the invasive evaluation method of drawing blood.

Bilirubinometers are, unfortunately, very expensive and the high cost limits their widespread use in outpatient settings in low- and middle-income countries. In this thesis, we explore the use of computer vision and machine learning to create a low-cost alternative to the expensive state-of-the-art transcutaneous measuring techniques. Our approach requires only a smartphone and an inexpensive SkinChecker (color checker optimized for skin) used for color correction purposes.

1.2 Project Goals and Research Questions

The primary goal of this thesis is to explore bilirubin prediction, using images captured by smartphone cameras. Illumination and camera sensors are variables that add inconsistency to images, and color correction must be performed prior to the prediction. To reach our goal; the following research questions are set forth:

RQ 1. What is the most accurate and robust technique to perform human skin color correction for scientific use?

- **RQ 1.1.** Which color correction technique is most robust to varying illumination?
- **RQ 1.2.** Which CIE illuminant is most suitable for general color correction?
- **RQ 1.3.** How well can a color correction technique (optimized on a color checker) generalize its prediction capability to a more diverse range of colors?
- **RQ 1.4.** Is there a single best color correction model regardless of CIE RGB color subspaces[1] CIE reproduced illumination?
- **RQ 1.5.** Can the machine learning approach Gaussian Process Regression achieve a more robust color correction than state-of-the-art least squares regression approaches?

[1]Color subspace (or color sub model) describes a range of colors inside a subset of the CIE RGB color model. See section 2.4 for more

RQ 2. Is it possible to predict bilirubin concentration using skin color imaged by smartphone cameras?

- **RQ 2.1.** Can computer vision and deep learning be used to achieve competitive results to state-of-the-art TcB measurement approaches (bilirubinometers)?
- **RQ 2.2.** Given color correction, are image-based bilirubin prediction models flexible enough to be used in outpatient and home environments?
- **RQ 2.3.** Can image-based bilirubin prediction serve as a standalone diagnostic tool for severe hyperbilirubinemia?
- **RQ 2.4.** Does spatially adjacent color features in human skin provide additional information about the underlying bilirubin concentration?

1.3 Contribution

With an evident need for globally affordable jaundice detection technologies, our main contribution is an inexpensive, smartphone-based bilirubin prediction model that enables widespread jaundice screening in outpatient and home settings. Today, physicians in low- and middle-income countries struggle to detect jaundice as subjective visual examinations are the only form of medical evaluation. The idea is to provide a solution that first and foremost caters to health care workers in clinics where advanced equipment is unavailable due to cost. Additionally, we provide an easy-to-use tool for parents to monitor their newborns at home. Overall we contribute with a solution that can significantly reduce brain dysfunction and mortalities related to jaundice.

In the process of building the jaundice detection pipeline, problems related to image inconsistencies were encountered. To overcome these issues, we propose a novel color correction solution, using Gaussian process regression (GPR), a machine learning approach that adapts extremely well to unseen colors. A paper related to our novel color correction approach is currently being written.

Lastly, for bilirubin prediction, we believe the deep learning based approach merits a standalone paper. The paper will go into further detail on bilirubin prediction, network architectures, and regression on images from smartphone cameras.

1.4 Thesis Structure

In **Introduction** the motivation behind the thesis, our contributions, and the thesis research questions are presented. The **background** chapter provides the basic theory for the methods used in this thesis and a literature review of related work. First jaundice, its relation to human skin, and jaundice detection technologies are presented. An explanation of color theory and color correction approaches are then covered. Lastly, a brief introduction to machine learning used for regression is reviewed.

The proposed solutions are presented in **Methods**, a two-fold chapter. First the correction solutions are introduced. Then, machine learning regression techniques are described, along with the jaundice dataset used to build and evaluate the bilirubin prediction models.

In **Results** the performance of color correction and bilirubin prediction models are presented. Color correction solutions are evaluated in terms of $\Delta E{*}_{a,b}$ using color checkers as ground truth, and bilirubin prediction models are compared to a Dräger JM-103.

The **Discussion** chapter presents a discussion of the results and reflects upon the strengths and weaknesses of the proposed solutions. Based on the results and the following discussion, a final **Conclusion** for color correction and smartphone-based bilirubin prediction is made. Lastly, future work is reviewed.

Chapter 2

Background

2.1 Optical Properties of Human Skin

To make an optical diagnosis; it is essential to understand how alterations in the information source (e.g., skin) affect visual features. The visual properties of human skin and its perceived color is attributed to underlying substances' interaction with light. To understand the cause of jaundice skin discoloration, we must first understand human skin, and how each component's concentration alters the perceived color of skin. This section will summarize light absorption, the optical properties of human skin, and their connection to jaundice.

2.1.1 Light Absorption

Light is electromagnetic radiation, and how it interacts with physical objects depend on the frequency of light waves and the nature of the molecules constructing the object. Electrons inside an atom have a natural frequency at which they tend to vibrate. If a light wave of a given frequency hits electrons that have equal frequencies, the electrons will absorb the energy of the wave.

The absorbed energy is often manifested as heat, but more importantly, light absorption prevents incoming light from being reflected. The intensity of absorption as a function of frequency, is what we refer to as an absorption spectrum (Fig. 2.2). Different material will absorb light at different wavelengths due to their underlying molecular composition. The absorption spectrum can, therefore, be seen as a fingerprint for a molecule, uniquely identifying it. It is this property that allows us to estimate concentrations based on reflected light.

We can use absorption spectrums to estimate the concentration of a given substance. By emitting light onto a surface and recording the amount of reflected light, we learn the aggregate absorption effect of the surface. The results can be cross-referenced with the absorption spectrum of the target substance to find its concentration.

An equally important consequence of light absorption is the perceived color of an object. An object is visible to an observer due to the light that is reflected off the object. Because of light absorption, molecules only reflect a subset of incoming frequencies. The residual light after absorption aggregates to the object's perceived color. The molecules constructing an object thus defines the object's perceived color and indicates its molecular composition.

2.1.2 Human Skin Physiology

Skin is a layered organ, protecting the human organism against environmental stress, such as heat, radiation, and infections. It contains light-absorbing substances, and skin's visual appearance is attributed to both its reflection and absorption properties.

Skin consists of three main layers; epidermis, dermis, and the hypodermis (Fig. 2.1). The outermost layer (epidermis), is usually $50-80$ µm thick and contains melanin, the pigment responsible for human genetic skin coloration. The underlying layer (dermis), has a more complex structure, usually $1-4$ mm thick, and consists of connective tissue and blood vessels. At the deepest layer, the hypodermis, made of fat and connective tissue, provides insulation and protection against mechanical injury.

2.1.3 Bilirubin

Bilirubin is a waste product formed during a natural process that breaks down the oxygen-carrying components in blood, hemoglobin, and myoglobin. (McDonagh and Lightner (1985)). The hemoglobin breakdown occurs mainly in the liver (Tenhunen et al. (1969), and the resulting bilirubin waste product is toxic to certain neurons. Bilirubin exhibits an absorption peak at 460 nm, giving the compound a yellow/orange color (Anderson and Parrish (1981)), shown in figure 2.2.

[1]https://www.dreamstime.com/royalty-free-stock-image-layers-human-\
skin-melanocyte-melanin-epidermis-melanocytes-produce-pigment-which-\
can-then-transfer-to-other-image37423406

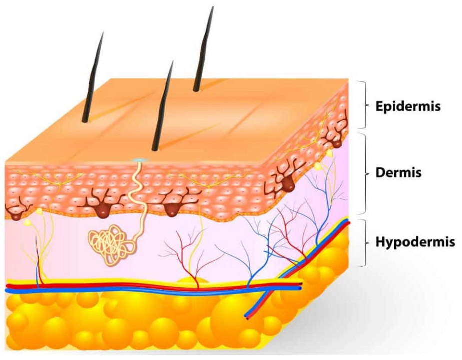


Figure 2.1: A diagram of human skin. The diagram shows the three separate layers that construct the skin organ; epdermis, dermis, and hypodermis. Adapted from Designua[1].

2.1.4 Melanin

Melanin is a broad term for a group of natural pigments and is one of the most influential factors when it comes to human skin color. The amount, shape, and size of melanin, found in the epidermis, is genetically determined and dictates skin colors associated with race (e.g., brown, white, yellow). Melanin affects how we perceive skin color by absorbing light at various wavelengths, and has a broad absorption band ranging from the ultraviolet into the near-infrared parts of the spectrum (Kollias and Baqer (1987)). Its concentration, which alters skin's perceived color, can be assessed by measuring reflectance at 568, 660, and 880 nm (Masuda et al. (2009)), shown in figure 2.2.

2.1.5 Blood

The absorption properties of blood are mostly attributed to hemoglobin absorption. Hemoglobin is a protein found in red blood cells, whose task is to carry oxygen from the lungs to the rest of the body. Hemoglobin on its own, deoxyhemoglobin (Hb), has a dark red color (Anderson and Parrish (1981)), but if bound to oxygen, oxyhemoglobin (Hb02), becomes bright red (Zijlstra et al. (2000)). Within the range of visible light, Hb exhibits two absorption maxima (400 nm and 550 nm), while Hb02 shows three (400 nm, 548 nm, and 576 nm)(Fig. 2.2). The two substances contribute to the rosy complexion associated with good health in light-skinned people.

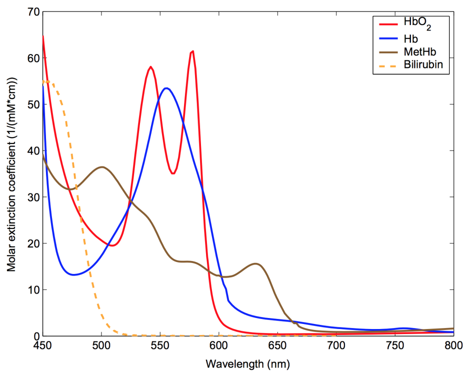


Figure 2.2: Extinction coefficients for hemoglobin, methemoglobin, and bilirubin. Reprinted from Randeberg (2005).

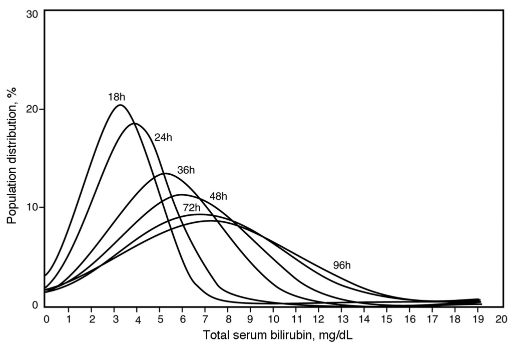
2.2 Jaundice

The concentration of bilirubin in human blood is usually restrained by the liver, whose task is to filter out waste products from the bloodstream. In the liver bilirubin is conjugated by enzymes, making it soluble in water and removed from the body through bile.

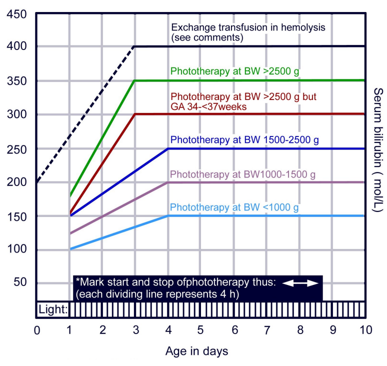
If the liver, for any reason, cannot conjugate bilirubin efficiently, unconjugated bilirubin starts to accumulate in the blood. Hyperbilirubinemia is the term used to describe a condition where bilirubin concentration exceeds normal levels (generally below 25 µmol/L for an adult). If untreated, the excess bilirubin seeps out of the blood vessels into fatty body tissue, often discoloring skin and the white part of the eye (sclera). It is this discoloration, caused by pigments from bilirubin, that is referred to as jaundice.

Starting from the chest; the bilirubin discoloration spreads and makes its way to outer parts of the body (e.g., feet, hands, head). If hyperbilirubinemia reach a certain level, unconjugated bilirubin breaches the blood-brain barrier and allows bilirubin to accumulate inside the brain. The bilirubin breach can inflict permanent brain damage and may result in cerebral palsy, deafness, brain damage or even death.

Infants are especially exposed to high concentrations of bilirubin. While growing in the mother's womb, the placenta removes unconjugated bilirubin from the fetus. Because of this, the bilirubin conjugating enzymes are actively shut off during the pregnancy. After the umbilical cord is cut, the task of filtering bilirubin from the blood rests on the baby's liver. The enzymes, however, need time to become fully active, and



(a)



(b)

Figure 2.3a: Shows the distribution of TSB at selected time intervals after birth. Reprinted from (Maisels and Kring (2006)). **Figure 2.3b:** A jaundice treatment determination chart: The chart shows weight and age dependent bounds for the treatment of hyperbilirubinemia. It is a guideline for management of neonatal jaundice currently used in most pediatric departments in Norway.

unconjugated bilirubin starts to accumulate. The distribution of bilirubin concentration at selected time intervals after birth is shown in figure 2.3a.

Of all newborns, 60 - 80% suffer from some degree of hyperbilirubinemia due to low enzyme activities in the liver (Avery GB (1994)). Usually, the conjugating enzymes become functional within a couple of days, and the neonate cures itself. In cases where the enzymes slumber for too long, treatment of jaundice will commence if the concentration of bilirubin (µmol/L) reaches a certain threshold, dependent on the neonate's age and body weight (Fig. 2.3b). 5 - 10% of neonates receive treatment for jaundice, usually treated through light therapy (McDonagh and Lightner (1985)). However, severe cases will require a blood transfusion.

2.3 Measuring Bilirubin

Hyperbilirubinemia is caused by high concentrations of bilirubin in the bloodstream, and can, therefore, be precisely determined by a blood test. The blood test measures the amount of µmol bilirubin per liter blood ($\mu mol/L$) and is the only way to know a patient's *true* bilirubin concentration. The resulting concentration found by the blood test is called total serum bilirubin (TSB), and is today, used to determine whether a patient should receive treatment. However, performing a blood test on all neonates is too expensive, and there is a widespread consensus that invasive methods, such as blood tests, are unnecessary.

2.3.1 Transcutaneous Bilirubin

The yellow discoloration caused by bilirubin is a good indicator of the underlying bilirubin concentration and is used by doctors as a screening method[2]. Advancements in technology have enabled physicians to exploit the light absorption properties of bilirubin through transcutaneous[3] methods quantitatively asses the severity of hyperbilirubinemia. The transcutaneous methods rely on absorption spectroscopy explained in Section 2.1.1, and hardware that measure the reflection properties of a material as a function of wavelengths. The transcutaneous apparatus referred to as transcutaneous bilirubinometers, are expensive (the state-of-the-art Dräger JM-105 (newly replaced JM-103) costs 49 900 NOK eks.mva[4]).

The result from a transcutaneous bilirubinometer is, however, not a precise measurement of a patient's bilirubin concentration. It is rather an estimated TSB value based on differences in optical densities for light at different wavelengths and is called transcutaneous bilirubin (TcB).

Since transcutaneous bilirubinometers measure the bilirubin in extravascular tissue and not the blood, several studies have gone to great lengths in analyzing the correlation between TcB and TSB. Researchers agree there is a linear correlation between TSB and differences in light absorbance. However, literature disagrees as to how reliable these estimations are. Rubaltelli et al. (2001) argue that transcutaneous measurements can be used not only as a screening device but also as a reliable substitute, thereby replacing invasive TSB sampling altogether. On the other hand, Maisels (2015) concludes that TcB measurements are not good enough to substitute TSB, but that it does tell us (1) when to worry about an infant and (2) when to obtain TSB from a blood sample.

Several bilirubinometers are being produced, and the available meters can be divided into two categories:

1. Multi-wavelength Spectral Reflectance meters (Bilicheck)
2. Two-wavelength (460 nm, 540 nm) Spectral Reflectance meters (Dräger JM-103)

In a bilirubinometer comparison survey, Francesco Raimondi and Capasso (2012) conclude that while targeting significantly more wavelengths, Bilicheck and JM-103

[2]Screening, in medicine, is a strategy used to identify the possible presence of an as-yet-undiagnosed disease

[3]Transcutaneous - Applied, or measured across the depth of skin. [Oxford Dictionary]

[4]Price collected from the Dräger dealer in Norway

Apparatus	Caucasian $(n = 503)$	Black $(n = 253)$	Hispanic/Asian $(n = 93)$
JM-103	0.949	0.822	0.926

Table 2.1: The Däger JM-103 TcB measurements on Caucasian, Black and Hispanic/Asian neonates. The numerical values represent the Pearson correlation coefficient between TSB and the JM-103's measured TcB. The numbers are collected from Maisels et al. (2004).

are equally reliable screening tools for hyperbilirubinemia. Due to the negligible difference and for simplicity, this thesis will focus on the dual-wave JM-103 bilirubinometer.

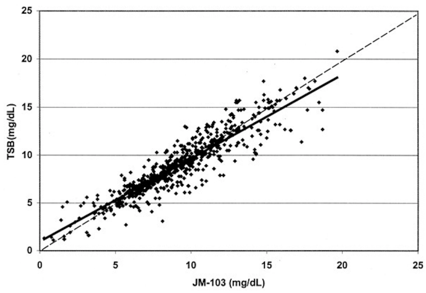
Dräger JM-103

The Dräger JM-103 is a state-of-the-art transcutaneous bilirubinometer relying on a two-wavelength (460 nm, 540 nm) spectral reflectance meter. In 'Hyperbilirubinemia and Transcutaneous Bilirubinometry', El-Beshbishi et al. (2009) go into detail on the measurement principle of JM-103. In short, the JM-103 determines TcB by measuring the difference in optical density for light in the blue (450 nm) and green (550 nm) wavelength regions. Note that these wavelengths correspond to the bilirubin and hemoglobin absorption peaks in figure 2.2.

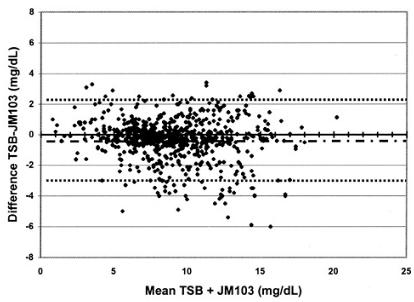
When assessing the performance of a transcutaneous bilirubinometer, two metrics are widely used; Pearson's correlation coefficient and TSB - TcB difference. Table 2.1 shows results from a study conducted by Maisels et al. (2004) on 849 children, grouped by skin color. Plotting the predicted TcB vs. measured TSB (Fig. 2.4a) visualizes the correlation between the TSB and TcB. The plot highlights bilirubin concentration intervals where the transcutaneous bilirubinometer performs poorly and vice versa. The difference plot (Fig. 2.4b), often called a Bland-Altman plot, reveals trends in the predictions, showcasing under- or over-estimations by the bilirubinometer. The dotted line shows the 95% confidence interval from the mean prediction (bold dotted line).

2.3.2 BiliCam

Another way to predict bilirubin concentration is by assessing the outer skin layer, visible to the human eye. The visual features of skin can be captured by a camera, recording the RGB values induced by the skin. The light absorbing properties of bilirubin turns skin yellow, and the yellow intensity indicates the underlying bilirubin concentration. Finding a direct relationship between skin color and bilirubin concentration can be used to predict the bilirubin concentration.



(a) Linear regression plot of JM-103 TcB versus TSB measurements. Bold line shows the regression line and the dotted line is the line of identity



(b) Difference plot (Bland-Altman) for the total population, between measured JM-103 TcB and TSB measurements. The dotted line shows the 95% confidence interval from the mean prediction (bold dotted line)

Figure 2.4: A study conducted by Dräger with 849 children. Reprinted from Maisels et al. (2004).

Taylor et al. (2017) proposes BiliCam; a smartphone application that predicts a neonate's TSB using an embedded smartphone camera. The application captures six images of a neonate, with a modified Macbeth color checker on its chest, and sends the images to a server for processing. After the images have been analyzed, a predicted bilirubin concentration value is returned to the user. The image analysis is done using machine learning and regression techniques, where they claim to achieve an overall prediction correlation of $r = 0.91$.

Their evaluation technique, however, is misleading and facilitates unrealistic results. Using supervised learning they forget to hold out a test set, implying the set used to evaluate the final model, is also the validation set used to optimize hyperparameters. Hence, every time they tune the model, based on validation results, some information from the validation/test set is leaked to the model, often referred to as *information leakage* (Chollet, 2017, chapter 4), a fundamental principle to avoid in supervised learning. Failing to do so leads to a model that performs artificially well since it is, to some degree, optimized for the final evaluation. This is explained further in section 2.7.3.2.

The results they present are, therefore, from a research perspective, invalid and we will not use their results for future comparison.

2.4 Color & Color Spaces

Color is the brains reaction to a specific visual stimulus, where the eye's retina samples color using three broad bands, roughly corresponding to red, green, and blue light (Ford and Roberts (1998)). Specifically, the human visual system relies on three receptors, called cones, that specialize in sensing short, medium and long wavelengths.

Terms used to describe color, such as hue (i.e., color's position on a color wheel), brightness, and intensity, are subjective and make color comparison difficult. The trichromatic theory - that any color can be described by combining three linearly independent colors (e.g., red, green and blue) - is the basis on which photography and most computer color spaces operate. More formally, it is stated that any color can be matched by a linear combination of three other colors, provided that none of those three can be matched by a combination of the other two, and is Grassman's first law of color mixture (Grassmann (1853)).

A **color space** is a set all possible colors that can be made from a group of colorants (Nishad (2013)) (Fig. 2.5) with the purpose of easily describing colors. To further simplify comparison and measurement of color, the International Commission on Illumination (CIE) studied human color perception and, in 1931, developed a standard color space called the CIE XYZ color space. It was the first quantitatively defined relationship between light frequencies and the human visual system. Instead of using red, green and blue, the CIE XYZ 1931 color space defines a 3D coordinate system, where the axes X, Y, and Z are extrapolations of RGB created mathematically to avoid negative numbers.

2.4.1 XYZ 1931 Color Space

The XYZ 1931 color space was the first formally defined color space and is the root of all colorimetry. The color space encompasses all color sensations visible to a person, but are not real colors, and cannot be generated in any light spectrum. The color space is based on experiments done on the human visual system, and the three cones that sense light. Y corresponds to luminance (brightness), Z is an approximation to the receptor capturing short wavelength light (blue light), and X is a linear combination of the three receptor response curves (short, medium and long wavelengths). XYZ values are often referred to as **tristimulus** values.

2.4.2 CIE L*a*b* Color Space

The CIE L*a*b* is a color space whose goal is to be 'device-independent'. That is, it defines colors independently of how they are created or displayed. This property makes

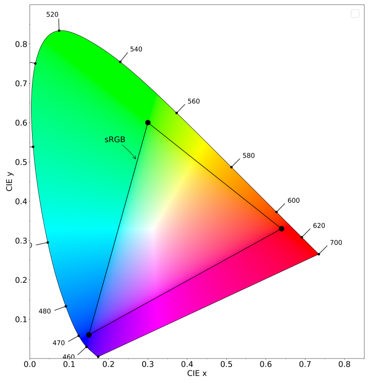


Figure 2.5: CIE xy Chromaticity diagram of all colors visible to the average human eye. The triangle defines the sRGB color space. Any color within the triangle can be represented by mixing the colors at its corners (Red, Green and Blue).

the CIE L*a*b color space very useful in research as results are comparable regardless of the image capturing device. It was developed in 1976 by the CIE, and expresses color as three numerical values; L* for lightness, a* for green-red color components and b* for blue-yellow color components. The CIE L*a*b* color space allows us to specify any color A in terms of its coordinates, and be confident that a defined color B will be equal to A, given that B is defined with the same definition as A.

2.4.3 Standard RGB (sRGB) Color Space

The standard RGB (sRGB) color space (Anderson et al. (1996)), often referred to as the sRGB color space, is a color space cooperatively created by Microsoft and HP. The color space is a color sub-space of all colors visible to the average human eye (Fig. fig:background:color:sRGB) and was developed in 1996 for the use in monitors, printers, and the internet. If no other color space is specified, it is the default color space for images.

2.4.4 Delta E (ΔE)

A standard color space allows for equal comparison. However, a unified metric for color difference is required to express the difference between two colors correctly. Having studied human's perception of color, the CIE constructed the concept of ΔE. The idea is that a color difference of $1\Delta E$ is the smallest color difference a human eye can detect (i.e., any color difference less than $1\Delta E$ is imperceptible to the human eye). As with color spaces, several ΔE versions have been proposed over time. The first ΔE formula was presented alongside the CIE L*a*b* color space in 1976 (Sharma, 2002, chapter 1), and its formula is the Euclidean distance between two colors in the CIE L*a*b* color space:

$$\Delta E*_{ab} = \sqrt{(L*_2 - L*_1)^2 + (a*_2 - a*_1)^2 + (b*_2 - b*_1)^2} \tag{2.1}$$

With this definition, it is estimated that a human will regard a $\Delta E^*_{ab} \approx 2.3$ difference as 'just noticeable'.

2.4.5 CIE Illuminant

A light source is an object that emits light. The quality and energy of the light is, however, not always consistent (e.g., sunlight varies throughout the day), and light sources are often seen as unreliable, and cannot be technically reproduced. To create light suited for colorimetric calculations, the CIE introduced the concept of standard illuminants. A standard illuminant is a theoretical source of visible light where its spectral power distribution is explicitly defined. Each illuminant has a color temperature that describes the relationship between the temperature of the source material, and the energy distribution of its emitted light. In simpler terms, high temperatures give a white glowing light, and colder temperatures give a more yellow/orange glowing light. Illuminants are divided into series (e.g., **D**-series (daylight)) describing source characteristics shown in Table 2.2.

Name	Color glow	Temperatur (K)	Description
A	🟧	2856	Incandescent / Tungsten
B	🟧	4874	Direct sunlight at noon
C	⬜	6774	Average / North sky Daylight
D50	🟧	5003	Horizon Light
D65	⬜	6504	Noon Daylight
E	🟧	5454	Equal energy
F1	⬜	6430	Daylight, Fluorescent
F8	🟧	5000	D50 simulator, Fluorescent
F11	🟧	4000	Philips TL84, Fluorescent

Table 2.2: Different CIE illuminants, where the letter defines serial code. The colors are RGB representations of each white point, calculated with luminance Y=0.54 and the standard observer. All table values are collected from Pascale (2003) and HunterLab (2005).

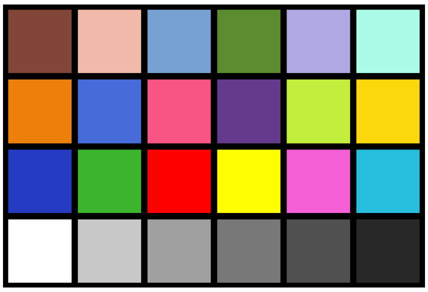
2.5 Color Checker

A color checker, also known as a calibration card, is a physical set of colors defined in the CIE L*a*b* color space. The colors are, thus, defined regardless of light source or image capturing device. The device independent property makes the color checker a perfect reference point when color correcting images (section 2.6). If a color checker is included in an image, the RGB errors at each color patch can be calculated and provides information about the RGB variations in the image, which again can be used to correct the color errors.

2.5.1 General Color Checkers

The Macbeth ColorChecker Classic by McCamy et al. (1976) (Fig.2.6a) is one of the most commonly used reference targets in photographic work. It consists of 4x6 color patches, each $50mm^2$, characterized by different spectral responses. The checker is designed to approximate colors found in nature and includes colors representing human skin, flowers, water, and sky. Six patches are different neutral gray, from black to white, where the spectral response of each patch is constant at all wavelengths and differ only in intensity. We referred to these six patches as the grayscale of the color checker. Other adaptations of McCamy's color checker have since been produced (e.g. Datacolor's **SpyderCHECKR 24**[5]).

[5]The SpyderCHECKR 24 is used in future experiments. All target values are given in Appendix A


(a) The Macbeth ColorChecker Classic 24

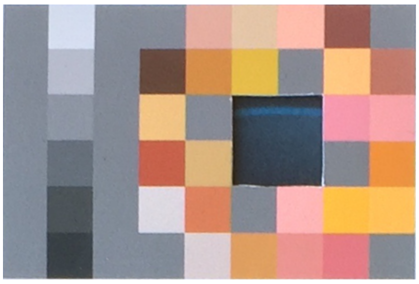

(b) The SkinChecker by Picterus AS

Figure 2.6: Figure **a** shows the Macbeth ColorChecker Classic, with 24 unique colors characterized by different spectral responses. The colors are intended to mimic those of natural objects such as human skin, foliage, and flowers. Figure **b** shows the SkinChecker with 31 unique colors. The colors are related to skin pigmentations found in various racial groups.

2.5.2 Skin Color Checker

For the detection of Jaundice, Gunnar Vartdal from Picterus AS[6] designed a custom color checker (Fig. 2.6b). We will refer to this color checker as the **SkinChecker**, and all target values are given in Appendix A. The colors on the SkinChecker are based on simulated reflection spectra of neonate's skin with varying skin parameters. These reflection spectra have been printed using spectral printing, a technique which attempts to recreate the whole reflection spectrum instead of just the RGB color values (Brito (2016)). The effect of spectral printing, as opposed to normal printing, is that the color checker colors are expected to change in the same way as the skin that is being measured when they are subjected to different illumination sources.

The color patches on SkinChecker have been chosen from reflection spectra based on numerical simulations spanning the whole range of possible skin parameters for newborns. The color selection has been done to ensure that the color checker works for neonates of all ethnicities, as well as covering cases where certain skin parameters of the newborn, such as skin thickness, is very different from the mean of that parameter. In addition, the color checker includes a plurality of gray patches. These patches can be used to correct spatial illumination variation or to investigate the white balance characteristics of the color correction.

[6]`http://www.picterus.com/`

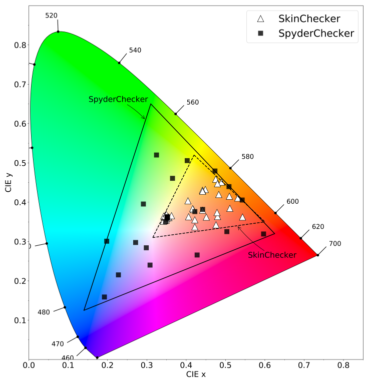


Figure 2.7: The CIE xy Chromaticity diagram with the SpyderChecker 24 and SkinChecker's colors individually positioned. The SkinChecker contains skin related colors and resides only in the red-yellow part of the color space. The SpyderChecker 24 is a diverse color checker and its colors are sparsely placed in the color space.

2.6 Color Correction

Cameras record three color responses (R G B). Variations in these color responses, caused by inaccurate sensors and illumination, result in device **dependent** RGB values. The task of correcting errors in the captured RGB values is referred to as **color calibration** or **color correction**. Without correcting the color information distorted by vision system components, important color characteristics in images may be inaccurately represented.

Color correction involves mapping device dependent RGBs to corresponding device **independent** color values (e.g., CIE L*a*b* or sRGB). To correct RGB errors, it is common practice to use a color checker as a reference target, and calculate a minimum error mapping between RGB and the color checker target values.

However finding this mapping is not always easy, and the problem of color correction arises from the fact that camera sensor sensitivities cannot always be represented as a linear combination of CIE color matching functions (Wyszecki and Stiles (1982)). This property results in camera-eye metamerism (Wandell (1995)) where certain surfaces, while clearly different to the human eye, will induce equal camera responses (i.e., the surfaces will look equal in an image). Color correction cannot resolve metamerism but aims to find the best possible mapping from device RGBs to device independent values.

Figure 2.7 shows the color diversity of a SpyderCHECKR 24 and a SkinChecker. The figure highlights the difference between general and specialized color correction, drawing triangles to visualize the RGB color sub-spaces defined by the two color checkers. For general color correction, a model must be able to reproduce a wide range of colors, as reflected by the SpyderCHECKR 24. The SkinChecker, on the other hand, does not contain either dominant green or blue, but instead contains substantially more data points in its focus area, allowing a model to optimize for more accurate color correction of human skin.

Look-up tables, least-squares linear and polynomial regression, and neural networks are some of the methods described in literature when trying to perform both general and specialized color correction.

Color correction can be done between any color spaces, but in the following section, we transform RGB to the CIE L*a*b* color space to exemplify the theory behind color correction.

2.6.1 Look-Up Tables

A trivial approach in color correction is the use of look-up tables. A look-up table is a large collection of camera RGB examples and the corresponding target values, manually created to define a mapping between the two color spaces. Color correction is performed by finding the recorded RGB value in the table and merely swapping it out with its corresponding target value. If a color value is not present in the table, an interpolation technique is implemented to find the most appropriate correction value. Look-up tables are typically used to perform color correction in color printers.

2.6.2 Least-Squares Linear Regression

A linear mapping from camera RGBs to CIE L*a*b* triplets can be achieved through a 3×3 linear transform. If we let ρ define a three element vector representing the three camera responses (R, G, B) and **q** define the three corresponding L*, a*, b* values, a simple linear transform can be written as:

$$\mathbf{q} = \mathbf{M}\rho \tag{2.2}$$

To clarify, **M** holds the coefficients (d_{ij}) of the transform that performs the actual color correction.

$$\begin{bmatrix} L* \\ a* \\ b* \end{bmatrix} = \begin{bmatrix} d_{00} & d_{01} & d_{02} \\ d_{10} & d_{11} & d_{12} \\ d_{20} & d_{21} & d_{22} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.3}$$

In the $3x3$ case, each component of the CIE L*a*b* triplet is individually calculated using the affiliated coefficients.

$$L* = f_{L*}(R, G, B) = d_{00} * R + d_{01} * G + d_{02} * B \tag{2.4}$$

$$a* = f_{a*}(R, G, B) = d_{10} * R + d_{11} * G + d_{12} * B \tag{2.5}$$

$$b* = f_{b*}(R, G, B) = d_{20} * R + d_{21} * G + d_{22} * B \tag{2.6}$$

With only one equation to satisfy, the 3×3 matrix (**M**), can perfectly map **q** $\rightarrow \rho$. However, we want to incorporate all color patches in the color checker and so we extend equation 2.2 to N color patches. If we let P be a $N \times 3$ set of raw camera RGB responses, and the set of $N \times 3$ known target values be denoted as **Q**, mathematically, the linear color correction (LCC) can be written as:

$$\mathbf{Q} = \mathbf{M}P \tag{2.7}$$

where the best mapping **M**: $RGB \rightarrow L * a * b*$ can be found by minimizing the least squares fitting :

$$\epsilon = \Sigma(\mathbf{q}_i - \mathbf{M}\rho_i)^2 \tag{2.8}$$

Equation 2.7 is an over-determined system of equations, and the mapping **M** from P to **Q** can be found through least-squares regression using the Moore-Penrose inverse (Aster et al. (2011)):

$$\mathbf{M} = \mathbf{Q}P^T(PP^T)^{-1} \tag{2.9}$$

We refer to the 3×3 linear transform as a LCC, mapping RGB to L*a*b* (Eq. 2.10).

$$(R, G, B)^T \xrightarrow{\mathbf{M}} (L*, a*, b*) \tag{2.10}$$

The LCC has proved to perform well in numerous studies, with the advantage of being independent of camera exposure (Finlayson et al. (2015)). It has, however, also been known to produce significant errors when mapping RGBs to CIE L*a*b* values.

Some studies show that LCC can, for some surfaces, generate errors up to $14\Delta E^*_{ab}$[7] (Guowei et al. (2001)). However, the same study shows that on average the 3×3 LCC transform yields an error of $2.47\Delta E^*_{ab}$ on the 8-bit Professional Color Communicator (Park and Park (1995)).

2.6.2.1 Least-Squares Polynomial Regression

A more modern approach for color correction is to assume the relationship between RGB and target values is polynomial, not linear. This assumption has lead to a more complex method for color correction, polynomial color correction (PCC), where the R, G, and B values at a pixel are extended by adding additional polynomial terms of increasing degree.

$$\rho = (R, G, B)^T \xrightarrow{\theta_k} \hat{\rho}_k = (R, G, B, ..., m)^T \tag{2.11}$$

We denote the kth polynomial extension of an RGB-vector (ρ) as $\hat{\rho}_k$. The extension operator θ_k transforms a three-element column vector to an m-element column vector with a set of added polynomial terms. For a simple RGB case i.e. $\rho = (R, G, B)^T$ the polynomial expansions of 2nd, 3rd and 4th degree are given below:

$$\hat{\rho_1} = [r, g, b, rg, rb, gb, r^2, g^2, b^2, 1] \tag{2.12}$$

$$\begin{aligned}\hat{\rho_3} = [&r, g, b, rg, rb, gb, r^2, g^2, b^2, rg^2, rb^2, gr^2, gb^2, br^2, bg^2,\\ &r^3, g^3, b^3, rgb, 1]\end{aligned} \tag{2.13}$$

$$\begin{aligned}\hat{\rho_4} = [&r, g, b, rg, rb, gb, r^2, g^2, b^2, rg^2, rb^2, gr^2, gb^2, br^2, bg^2, rgb,\\ &r^3, g^3, b^3, r^3g, r^3b, g^3r, g^3b, b^3r, b^3g, r^2g^2, r^2b^2, g^2b^2,\\ &r^2gb, g^2rb, b^2rg, r^4, g^4, b^4]\end{aligned} \tag{2.14}$$

Using the extension operator, the three RGB values recorded in a pixel are extended, represented by 9, 19, and 34 numbers respectively (if we strictly follow Equations (2.12) to (2.14)). As apposed to LCC's 3×3 matrix, PCC color correction is carried out by 9×3, 19×3, and 34×3 matrices (Eq. 2.3). If we let $\hat{P}$ denote the polynomial color response of N surfaces, then Eq. 2.9 can find the coefficients that minimize **M**, the $m \times 3$ PCC matrix.

In 'A study of digital camera colorimetric characterization based on polynomial modeling', Guowei et al. (2001) aim to discover the connection between modeling accuracy and the number of terms used in the matrices. They show that a matrix with

[7] $2.3\Delta E^*_{ab}$ is regarded as 'just noticeably different'

11×3 terms ($\hat{\rho} = [r, g, b, rg, rb, gb, r^2, g^2, b^2, rgb, 1]$) produces the best results for their two experiments with an average colour difference of around $1\Delta E^*_{ab}$.

2.6.2.2 Root-Polynomial Color Correction

If the correct polynomial fit is chosen, PCC can significantly reduce the colorometric error from LCC. However, the PCC fit depends on sensors and illumination, where exposure alters the vector of polynomial components in a non-linear way. Hence, choosing the right polynomial fit is very important in PCC. To solve the exposure sensitivity of PCC, (Finlayson et al. (2015)) present a polynomial-type regression related to the idea of fractional polynomials. Their method, named root-polynomial color correction (RPCC), takes each term in a polynomial expansion to its kth root of each k-degree term, and is designed to scale with exposure. The root-polynomial extensions for $k = 2$, $k = 3$ and $k = 4$ are defined as:

$$\bar{\rho}_2 = [r, g, b, \sqrt{rg}, \sqrt{rb}, \sqrt{gb}, 1] \tag{2.15}$$

$$\begin{aligned}\bar{\rho}_3 = [&r, g, b, \sqrt{rg}, \sqrt{rb}, \sqrt{gb}, \sqrt[3]{rg^2}, \sqrt[3]{rb^2}, \sqrt[3]{gr^2}, \sqrt[3]{gb^2},\\ &\sqrt[3]{br^2}, \sqrt[3]{bg^2}, \sqrt[3]{rgb}]\end{aligned} \tag{2.16}$$

$$\begin{aligned}\bar{\rho}_4 = [&r, g, b, \sqrt{rg}, \sqrt{rb}, \sqrt{gb},\\ &\sqrt[3]{rg^2}, \sqrt[3]{rb^2}, \sqrt[3]{gr^2}, \sqrt[3]{gb^2}, \sqrt[3]{br^2}, \sqrt[3]{bg^2}, \sqrt[3]{rgb},\\ &\sqrt[4]{r^3g}, \sqrt[4]{r^3b}, \sqrt[4]{g^3r}, \sqrt[4]{g^3b}, \sqrt[4]{b^3r}, \sqrt[4]{b^3g},\\ &\sqrt[4]{r^2gb}, \sqrt[4]{g^2rb}, \sqrt[4]{b^{2rg}}]\end{aligned} \tag{2.17}$$

We denote a root-polynomial extension of an RGB column vector ρ as $\bar{\rho}$. As with PCC, the RGB-extension naturally increases the size of the transform matrix **M**. Thus, RPCC is performed by a 7×3, 13×3, and 22×3 matrices (Eq. 2.3), if we strictly follow Equations (2.15) to (2.17)).

In their study, their root-polynomial solution outperforms PCC on average with $0.2\Delta E^*_{ab}$.

2.6.3 Artificial Neural Networks (Color Correction)

An alternative approach for color correction is the use of artificial neural network (ANN). Although several network architectures are capable of performing color correction, the most used techniques are Support Vector Machines (SVMs) and fully-connected feed-forward neural networks, both explained in section 2.7.

ANN have shown to be robust when optimized correctly, and achieve equally good results as a well fitted polynomial approach (Cheung et al. (2004)). However, both SVMs and fully-connected neural networks require extensive hyperparameter-tuning, a tedious process performed through trial and error, or using the computational expensive grid-search with cross-validation (explained in section 2.7.3.2).

2.6.4 Gaussian Process Regression (Color Correction)

A Gaussian process (GP) is a machine learning approach, that has, as of today, not been applied to the field of color correction. With no referable background story for color correction, we present the general Gaussian process in section 2.7.1.

2.7 Machine Learning

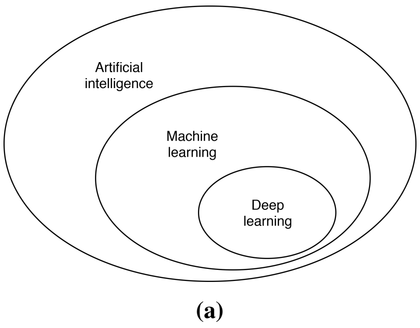
Artificial intelligence (AI) is an area of computer science where intelligence is demonstrated by machines, as opposed to humans or animals. The field of AI is divided into hierarchical subsets, where AI is the super-set of all its subcategories (Fig. 2.8a). Machine learning is one of the fields encompassed by AI and revolves around data-driven learning. That is, given data and corresponding answers, machines can find relationships between data and answers, and *learn* the rules that define the relationships (Fig. 2.8b). These rules can later be applied to new data to produce original answers. This process, of giving machines data with corresponding answers, is referred to as **supervised learning**. In this section we first present two statistic-based supervised learning approaches; SVMs and GPs. Then, we introduce the concept of deep learning and the fundamental theory behind ANN.

2.7.1 Gaussian Processes

GPs are widely recognized as a powerful, yet practical tool for solving both classification and non-linear regression (Williams (1997)).

A GPs can be thought of as a generalization of the Gaussian probability distribution over a finite vector space to a function space of infinite dimensions (MacKay (1998)). The processes are probabilistic models of functions and are used for solving both classification and non-linear regression problems.

To be more precise, a GP is used to describe a distribution over functions $f(x)$ such that any finite set of function values $f(x_1), f(x_2), ..., f(x_n)$ have a joint Gaussian distribution (Rasmussen and Williams, 2006, Chapter 2).



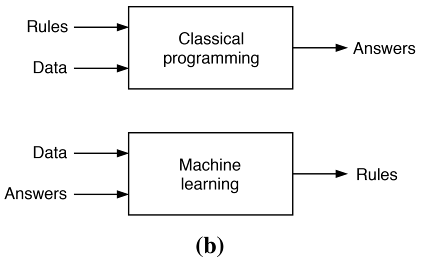


Figure 2.8a: A Venn diagram showing the relationship between artificial intelligence, machine learning, and deep learning. **Figure 2.8b:** The machine learning paradigm: The figure illustrates the difference between classical (non-data driven) programming, and machine learning.

A GP is fully specified by its mean and covariance function (i.e. no explicit hyper-parameters). We use Rasmussen and Williams (2006) definition, who defines the mean function $m(x)$ of a real process $f(x)$ as,

$$m(x) = \mathbb{E}[f(x)] \tag{2.18}$$

The covariance function of $f(x)$, commonly called 'the kernel' $k(x, x')$ is given by,

$$cov(f(x), f(x')) = k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))], \tag{2.19}$$

The GP, can therefore be denoted as,

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{2.20}$$

We often let the mean function be the zero function, $m(X) = 0$, as it can be accounted for by extending the kernel with an extra term. In such a case, all possible structures captured by a GP model is completely determined by the kernel and implies that the kernel alone determines the GP's ability to generalize and predict new data.

While no explicit hyperparameters are required, the kernel parameters need to be optimized. The GP uses log marginal likelihood to adjusts its kernel parameters on the assumption that the data follows a Gaussian (Normal) distribution. The log marginal likelihood formula is given by:

$$\begin{aligned}\log p(y|X) = &-\frac{1}{2}(y - m(X))^T(k + \sigma_n^2 I)^{-1}(y - m(X)) \\ &- \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log 2\pi\end{aligned} \tag{2.21}$$

where σ_n^2 is a noise variance. The first term, $-\frac{1}{2}(y - m(X))^T(k + \sigma_n^2 I)^{-1}(y - m(X))$ measures the data-fit and is negative quadratic. It is the only term dependent on the GP output y (derived from the GP's response to its observations) and is the core of the marginal likelihood function. The second term, $\frac{1}{2}\log|K + \sigma_n^2 I|$ measures and penalizes the complexity of the model, hence, it is a complexity term. The last term, $\frac{n}{2}\log 2\pi$ is independent of observations, and is a log normalization term. Given $m(X) = 0$ (i.e. using the zero function as mean), the log marginal likelihood is reduced to $-\frac{1}{2}y^T(k + \sigma_n^2 I)^{-1}y$

The prior of the GP (Fig. 2.9a) is only specified by the mean function and the covariance function (i.e., the kernel). Given a dataset consisting of two observations $\mathcal{D} = (x_1, y_1), (x_2, y_2)$, we can condition the joint Gaussian prior distribution on these observations and graph a new joint posterior distribution (Fig. 2.9c), simply by evaluating the mean and covariance matrix and generating new samples.

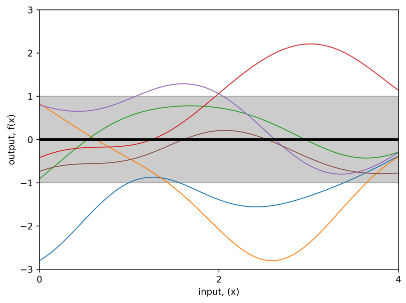
Although its drawback of computational expensiveness, $O(n^3)$. The GP is an extremely versatile supervised learning technique and can approximate almost any function regardless of complexity. This flexibility makes it robust to changing and unknown environments, and its built-in Occam's razor makes it highly suitable for a large problem space.

2.7.1.1 Kernels

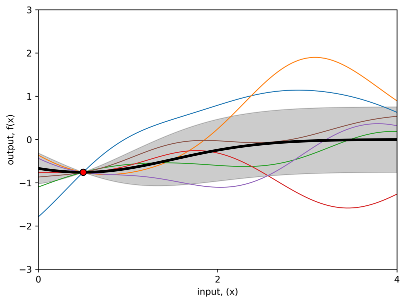
Kernels are, as discusses above, a crucial part of GPs. We have two main categories of kernels: Stationary and non-stationary kernels. **Stationary kernels** consider only the relativity between two observations (data points), and disregard the data points themselves. This property results in the kernel being invariant to translations in the input space. **Non-stationary kernels** are dependent on the specific values of the observations and are, thus, not invariant to translations. In the following paragraphs we discuss the most important kernels; radial-basis function (RBF) kernel, matérn kernel, rational quadratic (RQ) kernel, constant kernel, and the white kernel.

Constant kernel: Is the simplest kernel and is often used in combination with other kernels. When multiplied with another kernel, it scales the magnitude of that kernel. If however, combined with a kernel using the sum operator, it shifts the mean of the GP.

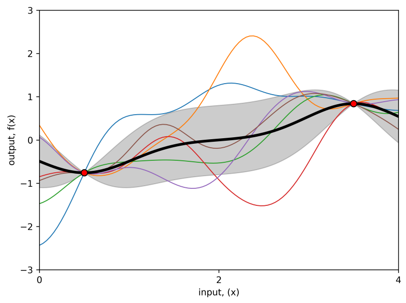
$$k(x_i, x_j) = C \; \forall \; x_i, x_j \quad \text{(where C is a constant value)} \tag{2.22}$$



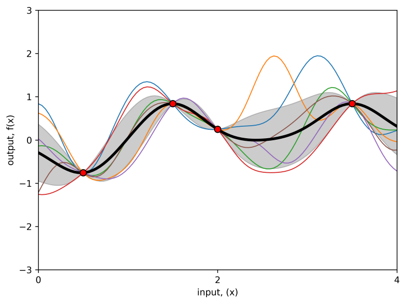
(a) Prior



(b) Posterior, with one datapoint



(c) Posterior, with two datapoints



(d) Posterior, with four datapoints

Figure 2.9: A visual representation of a GP model of a one-dimensional function. Subfig. **a** shows three functions drawn at random from a GP prior. Subfig. **b** to **d** show the functions conditioned on one to four observations. In all plots the shaded area represents the probabilistic nature in the form of a pointwise 95% confidence interval. The bold line represents the mean of the functions.

Radial-basis function (RBF): Also known as the squared exponential kernel, is probably the most used kernel and is stationary. It can be shown that the RBF corresponds to a Bayesian linear regression model with an infinite number of basis functions (Rasmussen and Williams, 2006, Chapter 2), implying that a GP with a RBF kernel has a mean square derivative of all orders, resulting in a highly smooth model. The RBF kernel is given by:

$$k(x_i, x_j) = \exp\left(-\tfrac{1}{2} d(\tfrac{x_i}{l}, \tfrac{x_j}{l})^2\right) \tag{2.23}$$

Matérn kernel: Is a generalization of the RBF kernel. It uses an additional parameter ν to control the smoothness of the approximated function. When $\nu \to \infty$ the model becomes very smooth, and we obtain the RBF kernel. The matérn kernel function is given by:

$$k(x_i, x_j) = \sigma^2 \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(\gamma \sqrt{2\nu} d(\tfrac{x_i}{l}, \tfrac{x_j}{l})\right)^\nu K_\nu \left(\gamma \sqrt{2\nu} d(\tfrac{x_i}{l}, \tfrac{x_j}{l})\right) \tag{2.24}$$

where K_ν is a modified Bessel function (Abramowitz and Stegun, 1964, chapter 9), and ν and l is a positive parameter.

Rational quadratic (RQ) kernel: Is the equivalent of summing together many RBF kernels with different lengthscales. The parameter α determines the relative weighting of the scale mixture. When $\alpha \to \infty$, RQ is identical to the RBF.

$$k(x_i, x_j) = \left(\tfrac{1 + d(x_i, x_j)^2}{(2 * \alpha * l^2)}\right)^{-\alpha} \tag{2.25}$$

White kernel: Also known as white noise, is combined with other kernels, to explain the noise component of a signal. It has the form:

$$k(x_i, x_j) = N \quad if\ x_i == x_j\ else\ 0 \quad \text{(where N is the level of noise)} \tag{2.26}$$

The white noise kernel is often added to prevent models from overfitting.

2.7.2 Support Vector Machine

The Support Vector Machine (SVM) is one of the most commonly used approaches for low configuration supervised learning. The low configuration property makes SVMs the go-to 'first try' approach in problems where domain knowledge is low.

The theory behind SVMs originates from statistical learning theory, and was first proposed and formally described by Cortes and Vapnik (1995). The SVM is a linear classifier, diving an n-dimensional feature space into two parts, one for each class. The SVM accepts n-dimensional data points (each belonging to one of two classes)

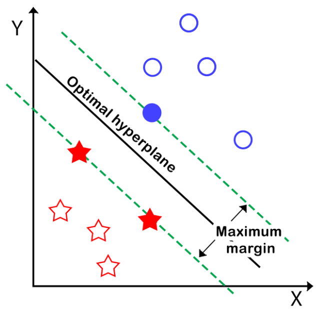


Figure 2.10: Support Vector Machine classification showing an optimal classification. The support vectors, marked with dark red and blue squares, define the margin of largest separation between the two classes. Reprinted from OpenCV[8].

as input and attempts to find an (n-1) dimensional hyperplane that separates the input data according to their corresponding classes. The hyperplane defines the classification model and can classify new unlabeled data by placing an n-dimensional data point in the feature space, and gauging its position relative to the hyperplane (e.g., above the hyperplane $\rightarrow$ class A, below the hyperplane $\rightarrow$ class B).

The SVM tries to separate two classes using a linear hyperplane. There are, however, an infinite number of hyperplanes that can classify the data. The SVMs goal is to find the hyperplane that divides the input data (based on classes) with the largest possible margin (Fig. 2.10). It does so by adjusts the slope of the hyperplane, through a process referred to as 'training the machine'. More formally, an optimizer function is applied to maximize the distance between the hyperplane and the data points closest to the hyperplane. These data points are referred to as **support vectors** and is where the name Support Vector Machine originates from.

For non-linear classification, SVMs use what is known as 'the kernel trick', where a non-linear kernel maps data points to a higher dimensional feature space. By transforming the input data into a different feature space, the data (previously not separable by a linear function), may, in fact, be linearly separable (Fig. 2.11. The kernel trick reduces the problem to a linear classification problem.

The most popular non-linear kernel functions are the RBF, Polynomial kernel, and the Sigmoid kernel. Choosing the right kernel function is hard, and it often boils down to trial and error.

[8]`https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html`

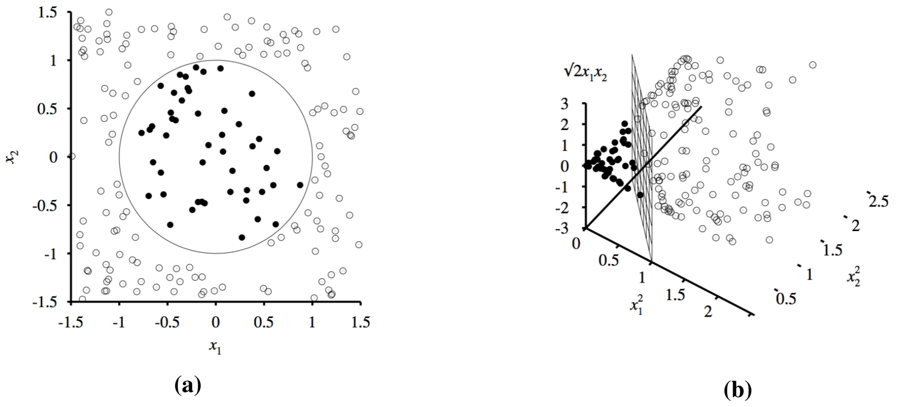


Figure 2.11: Subfig. a shows a two-dimensional training set. Positive examples is denoted as black circles and negative as white. Subfig. b shows the same data after mapped into a three-dimensional space, with $(x_i, x_j)^2$ as kernel function. Reprinted from (Russell and Norvig, 2016, chapter 18).

Using a complex kernel that transforms data to a too high dimensionality may lead to overfitting. It is desirable to find the dimensional space that facilitates data classification without aggressively increasing the number of dimensions. To ensure a general prediction model, we may have to allow for some misclassifications. Soft margin (i.e., a margin that allows for misclassification) was introduced in Cortes and Vapnik (1995), and implements a tuning parameter C (cost). The cost defines the degree of violation the soft margin accepts, where a low C permits few violations and vice versa.

Support Vector Regression

support vector regression (SVR) was first proposed by Drucker et al. (1996) and follows the core principle of the SVM classification. Instead of defining a linear hyperplane that best separates the input data into two classes, SVR outputs real-valued numbers. However, performing regression (not classification) requires a more flexible model to allow for some error. Thus, the SVR introduces a margin of tolerance, represented with ϵ. The ϵ value defines the margin of tolerance where no penalty is given to the model if an observation is within its bounds and vice versa. SVR tries to find a function, f(x), that deviates from the input data by a value no greater than ϵ (Fig. 2.12). The loss function is modified to include a distance measure (i.e., a distance between data points and f(x)), penalizing the model if any data point is further away than the maximum

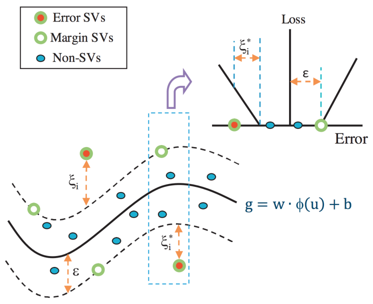


Figure 2.12: Nonlinear SVR with Vapnik's ε-insensitive loss function. Reprinted from N and Deka (2014).

margin of tolerance. In such a case, the model adjusts itself to include the outlying data points within its tolerance margin. This concept is implemented as an epsilon-insensitive loss function and was first proposed by Drucker et al. (1996).

The way the SVR's loss function handles errors makes it robust against outliers and noise, and it is a characteristic that leads to the SVR having a good generalization performance.

2.7.3 Deep Learning & Artificial Neural Networks

When solving complex regression problems, it is often hard to fit classical models (i.e., non-data driven models) to the problem. With the ability to learn intricate features, deep learning approaches can find hidden relationships in data and swiftly extract prominent features. This property has, in recent times, lead to increased use of deep learning where deep learning frequently outperforms classical computer algorithms.

Early artificial neural network (ANN) were directly based on our understanding of the human brain, more specific nerve cells, or neurons. Inspired by the structure of neurons, McCulloch and Pitts (1943) developed a simple mathematical model (Fig. 2.13), that roughly speaking outputs zero until a certain threshold, decided by an activation function, section 2.7.3, is reached. Their model is represented as a connected graph where neurons constitute nodes, and synapses are modeled as weighted edges. More specific a node (neuron) takes in weighted inputs(w_i) from other node outputs(x_i), sums these inputs $\Sigma(w_i * x_i)$ and finally adds a constant b from a bias unit, resulting in a value in. In the last stage, an activation function is evaluated based on in and decides whether the neuron should fire (i.e., output a non-zero value).

Since 1943 our understanding of the biological neuron has advanced, and today,

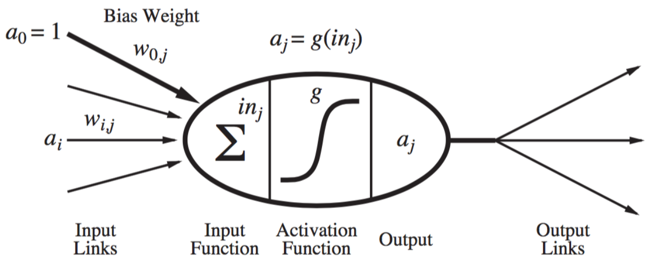


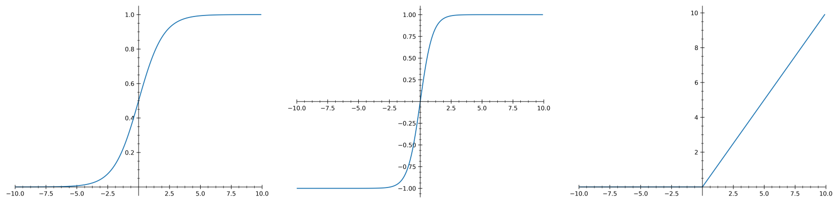
Figure 2.13: A simple mathematical model for a neuron. Where the output activation is $a_j = g(\sum_{i=0}^{n} w_{i,j}a_i)$, a_i is the output activation of neuron i and $w_{i,j}$ is the weight between neuron i and this one. Reprinted from Russell and Norvig (2016).

we have more complex models of the human brain. The way we use artificial neurons in computer science, however, has stayed almost unchanged, because AI researchers early turned their focus away from the physical properties of neurons and dived into their abstract properties instead. These abstract properties include their ability to learn, resiliency to noisy input, and their properties for distributed computations. Today, neuroscience is regarded as an important source of inspiration for ANN and deep learning research. However, it is no longer the predominant guide for the field.

ANN are often structured as a layered architecture, where inputs are fed through the first layer, called the input layer. The input data is propagated through the network, and altered by edge weights, biases, and activation function, before finally being evaluated to an output in the final layer, the output layer. A regularly used ANN is the multilayered feed-forward neural network. Here, layers consist of one or more nodes, where every node connects to both the preceding and successive layer. We refer to all layers between the input and output layers as hidden layers. The connections between layers are weighted directed edges in the model graph, pointing downstream (i.e., from input to output). By adjusting the weights in the model as a response to inputs and their corresponding ground truth, we say that the model is learning. Building these models with many hidden layers (i.e., a deep network) and having it readjust weights in response to input data, is the concept of deep learning.

Activation Functions

The output of every node, in a neural network, is decided by an activation function that takes in, as parameters, a set of inputs. Although there is a great variety of activation functions in use today, we only describe the most common implementations: Sigmoid, ReLu, and tanh (Fig. 2.14).



(a) Sigmoid **(b)** Tanh **(c)** ReLU

Figure 2.14: The most commonly used activation functions in an ANN.

Sigmoid is defined as $f(x) = \frac{1}{1+e^{-x}}$ and was one of the first activation functions to be implemented. It outputs a value within a continuous range from zero to one but saturates for inputs of large positive and negative magnitudes. This saturation, and the fact that its derivative range from 0 to 0.25, implies that layers deep in the neural network get a near-zero gradient, thus halting the learning process. Due to this, sigmoid is often a poor choice for hidden nodes and is mainly used in the last layer (output layer) of neural networks and networks without backpropagation (explained in the next section).

Rectified Linear Unit (ReLu), given by: $h(a) = max(0, a)$, has become highly popular, and is the go-to activation function for hidden layers. It is non-saturating (i.e., its output continues to grow and does not converge) and has a tendency to speed up the convergence of the learning compared to other activation functions. One of the main drawbacks, however, is the possibility of certain nodes never being activated (dead nodes), thus running the risk of getting stuck in a local minima.

The hyperbolic tangent (tanh) is similar to sigmoid in shape, but is shifted down on the y-axis, letting it output negative values. Its output ranges from -1 to 1 (Fig.2.14), and is zero centered, $tanh(0) = 0$, avoiding some of sigmoid's learning problems. Another feature is that its derivative is well defined over the entire value range, and, therefore, preferred over ReLu in cases where ReLu halts due to non-defined derivatives.

Optimization

Neural networks try to optimize an objective function. In the field of deep learning, optimizing refers to the task of minimizing some function $f(x)$ often referred to as the **cost function**, **loss function**, or **error function**. When optimizing neural networks, Stochastic Gradient Decent is one of the most frequently used algorithms.

Stochastic Gradient Decent (SGD) (LeCun et al. (1989a)) is a simple algorithm

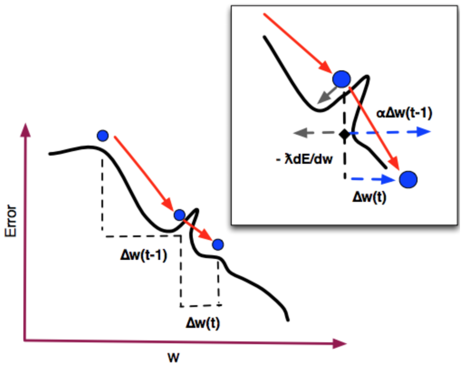


Figure 2.15: Stochastic Gradient Decent with momentum. The red line shows how momentum gives the SGD algorithm a push over the top, thereby avoiding getting stuck in the local minima. The grey arrow indicates the direction without momentum. Reprinted from Downing (2017).

that computes all gradients of a small batch of randomly selected cases and determines their combined effect on the loss function. This effect is then used in backpropagation, an algorithm that uses Jacobian matrices to propagates an error through the network (from output to input). The backpropagation readjusts the network weights according to their error contribution, updating all weights in the network.

A crucial parameter for the SGD is the **learning rate**, a hyperparameter that determines how large the weight updates are. It must, therefore, be chosen with care. If the learning rate is too high, the algorithm may not find the global loss function optima. On the other hand, a too small learning rate can cause the algorithm to get stuck in a local minima.

Introducing **Momentum** leads the SGD algorithm to converge faster and avoid local minima. Momentum can be seen as an evolution of the SGD algorithm and is a strategy where the gradient descent is given momentum in the direction of its previous move. The momentum allows the gradient descent to avoid a directional change towards a local minima, allowing it to diverge from sub-optimal solutions (Fig. 2.15).

Adaptive learning rate was introduced to further minimize the chance of getting stuck in a local minima and to converge faster. It computes individual learning rates for each parameter, as opposed to one unified, global learning rate. Today, one of the most used methods for adaptive learning is Adam (Kingma and Ba (2014)), which includes a way to maintain an exponentially decaying average, similar to that of momentum.

Generalization

A key attribute of a trained ANN is its ability to generalize. Generalization is the ability to handle unseen data, based on knowledge gained from training data, recognizing essential features, without memorizing the exact training data.

The opposite of generalization is overfitting (i.e., model is aggressively fitted to the training data). Luckily, there are several strategies that can help the network avoid overfitting, where **regularization** has, in recent years, been one of the most used methods to prevent this. This thesis will focus on two regularization strategies; dropout and parameter norm penalties.

Dropout: Dropout is a regularization technique where a subset of randomly selected neurons is deactivated during training. Dropout leads a more generalized model, forcing multiple nodes to learn the same concepts.

Parameter Norm Penalties: Is a regularization techniques that only penalizes the weight parameters of the model. A simple and common approach for parameter norm penalty is L^2 parameter norm penalty (L^2**-regularization**), commonly known as weight decay. L^2-regularization is a technique where high absolute valued parameters (weights) are penalized through the use of a regularization term in the cost function. The idea is that lowering the weights, causes the distribution of importance to spread out across the layer, as opposed to having some weights becoming very large, thus, overruling the consensus of the layer.

2.7.3.1 Convolutional Neural Networks

The theory presented in this section is gathered from Goodfellow et al. (2016, chapter 9) and Stanford CS class CS231n[9], if not explicitly stated otherwise.

Convolutional neural networks (CNNs) (LeCun et al. (1989b)) are similar to multi-layered feed-forward networks, with alterations specialized for image detection. With the success of Krizhevsky et al. (2012) on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al. (2015)), where they had considerably better accuracy than previous state-of-the-art solutions, CNN's got much attention from researchers all over the world, and is today one of the hottest fields of study in the field of AI. As the name indicates, CNNs are heavily reliant on the mathematical operation convolution.

A typical CNN takes an image (represented as 3d tensor) as input and propagates it through locally connected convolution and max-pooling layers. The final convolution/pooling layer outputs a feature map, which is propagated through one or more fully connected layers, resulting in a final an output (Fig. 2.16).

[9] `http://cs231n.github.io`

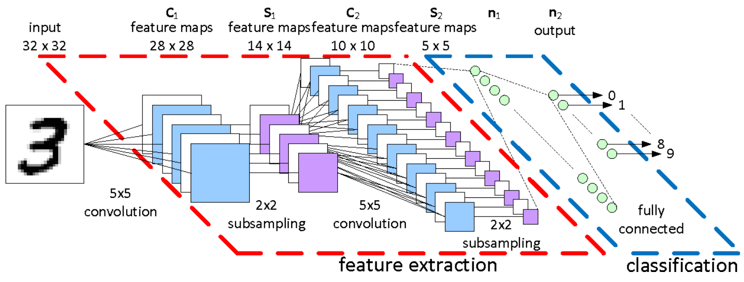


Figure 2.16: A CNN architecture for a handwritten digit recognition task. Reprinted from Bart and Eindhoven (2011).

Convolution

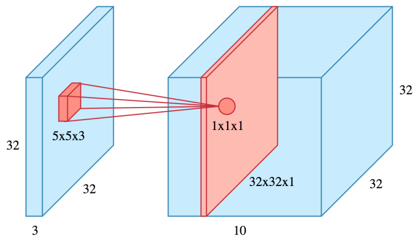
The convolutional layer (Fig. 2.17a) is the core building block in a CNN, and performs the majority of the calculations in the neural network.

Its layer parameters consist of a set of learnable filters/kernels, where each output node in a kernel is connected to a local region of the previous layer. With local regions, every node is only receiving information for a subset of the nodes in the previous layer. The size of the kernel decides the total amount of input values per output node (ie.e, how many nodes from the prior layer a node is connected to). Every kernel output node connects to one separate local region, a process called **Sparse Connectivity**. Sparse connectivity is one of the main properties of the convolutional layer, where the outcome is fewer mathematical operations, and a drastic performance boost compared to fully-connected layers.

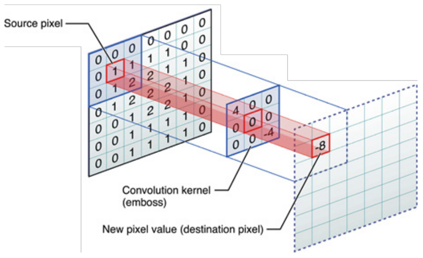
Another characteristic of the convolutional layer is **Parameter Sharing**, where each set of weights (i.e., the kernel) in the layer is used multiple times. The combined results of the parameter sharing are denoted as one feature map, of many feature maps, in the layer. Parameter sharing results in fewer parameters and saves computations, while also making features invariant to translation.

The convolution outputs a linear pattern and is propagated through a nonlinear activation function, such as ReLu. This stage is often referred to as the **detector stage**, and a **pooling-layer** often follows.

[10]`https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html`



(a)


Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)
-8

(b)

Figure 2.17a shows a convolutional layer. The red box in the input layer maps to one node in one of the feature maps in the convolutional layer. Reprinted from Nameer Hirschkind and Khim (2017). **Figure 2.17b** a numerical example of a 2D convolution. Reprinted from Apple[10].

Pooling

The pooling operation is an additional step of translation invariance. This time the invariance is, however, local and reduces the effect of slight object off-sets. The pooling operation runs a fixed size kernel across every feature map, sampling the desired attributes within the kernel window. Hence, pooling reduces the complexity of the data representation and also helps make the model less invariant to small translations of the input.

A form of pooling is max-pooling, where the desired attribute sampled is the maximum value within the kernel window. As an example, when trying to determine if an image contains a door, we do not need to know the doorknob's position with pixel-perfect accuracy, we only need to know that it is approximately in the vertical center.

2.7.3.2 Training, Validation, and Test Sets

A supervised learning model is trained, often referred to as fitting the model, by updating its internal parameter in response to training data and the corresponding labels.

To perform an unbiased evaluation of a model; all available data cannot be used during training. It is, therefore, common practice to partition the dataset into a training set and a testing set (Fig. 2.18). The model is trained on the training data, and when the model is adequately tuned, the test set is used to evaluate the performance of the final model. It is essential that observations from the testing set are not used, in any way, to choose model structure and hyperparameters. Hence, we call this a **held-out test set**, not to be used until final evaluation.

However, if the test set cannot be used to evaluate the model iteratively, there is no way of knowing how well an intermediate model is performing. For intermediate evaluations, it is common practice to split the training set into two subsets; **a smaller training set**, and **a validation set** (Fig. 2.18, Method 2). The validation set can now be used to estimate the model's generalization error (i.e., performance) during and after training, allowing the user to update the model architecture and hyperparameters accordingly.

If a validation set is not created from the initial training set, and instead, the test set is used to alter the model hyperparameters, every time the model is tuned, some information from the test set is leaked to the model. This leakage is often referred to as *information leakage* (Chollet, 2017, chapter 4), a fundamental principle to avoid in supervised learning. Tuning the model in response to the test set results leads to a model that performs artificially well as it is, to some degree, optimized for the final evaluation.

Cross-Validation

Dividing the dataset into fixed training, validation, and testing sets can be problematic if the dataset is small. It often leads to the validation and test set containing too few samples to be statistically representative of the data representation.

This problem can be solved using cross-validation. Following the held-out test set convention, one part of the available data is held out as a test set.

With cross-validation, instead of having a training and validation set, the idea is to repeatedly choose new, random training and validation subsets (or splits) of the dataset and benchmark these splits.

The most common method is the k-fold cross-validation, in which we split the data into k different subsets (also called folds), and use k-1 of the subsets for training, leaving the last fold for validation (Fig. 2.18, Method 1). This procedure is repeated until all fold have individually been validated. An often used split is the *leave-one-ou*t cross-validation, where for each run, only one training example is held out for validation ($k = N - 1$). The leave-one-out cross-validation maximizes the amount of training data.

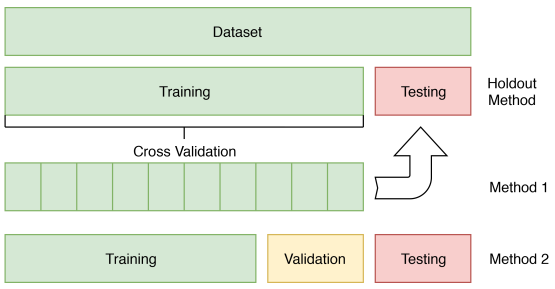


Figure 2.18: A visual representation of how to split a dataset. Method 1, shows the K-fold cross-validation, using the holdout method. Method 2, shows the regular training, validation, and testing split, also using the holdout method.

Chapter 3

Methods

The fundamental objective of the thesis is to assess the relationship between bilirubin and the camera sensor responses induced by skin color. All images in the provided dataset require post-processing color correction. This chapter is two-fold and presents the methodology of color correction and bilirubin regression in two separate sections. To give perspective, we first present the jaundice computer vision pipeline, from raw image to prediction, and the dataset used in the thesis.

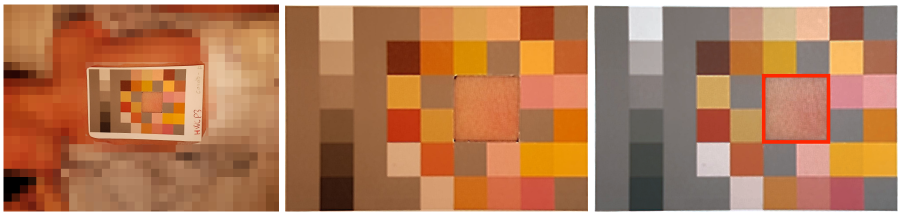
Jaundice Computer Vision Pipeline

The dataset, described in section 3.1, is a collection of 564 images. As discussed in section 2.2, jaundice is caused by high concentrations of bilirubin, a compound that gradually turns skin yellow.

We are interested in the relationship between RGB values captured by a camera and bilirubin concentration. If any consistent relationship exists, a regression model can predict a patient's bilirubin concentration via skin color.

For each image (Fig. 3.1a) illumination varies, and inconsistently shifts RGB values. This shift, also called error, pollutes the perceived skin color and may distort any possible RGB-to-bilirubin relationship. Having an illumination and hardware independent color standard is thus imperative when building a prediction model.

A SkinChecker is placed on the chest of all neonates and allows for color correction of human skin. Manual segmentation on the SkinChecker (Fig. 3.1b) is performed to simplify the following steps. With the SkinChecker as the reference point for the color



(a) Raw input image (Image is censored)   **(b)** Raw segmented SkinChecker   **(c)** Color corrected SkinChecker

Figure 3.1: Illustrates the jaundice computer vision pipeline. From raw images, (Fig. 3.1a), the SkinChecker is manually segmented. Gaussian Process Regression color correction is applied to the image. Finally, the visible skin patch within the SkinChecker (highlighted by the red square in Fig. 3.1c, is manually segmented and the average RGB color is sampled.

correction solutions, we apply pixelwise color correction (Fig. 3.1c), trying to restore the original colors.

After color correction, the visible skin patch within the SkinChecker is manually segmented (red square in Fig. 3.1c), and the mean RGB color value used by the regression model is calculated. To reiterate; two separate segmentations are performed. First, the SkinChecker is segmented from the raw image, to perform color correction. Then the small skin patch (within the SkinChecker) is segmented for the use in prediction models.

3.1 Jaundice Dataset

For the prediction of bilirubin, a labeled dataset is acquired from St. Olav's hospital. The dataset consists of 564 images depicting newborn babies. In each image, a SkinChecker, as discussed in section 2.5.2, is placed on the chest of the neonate, exposing the skin color within the SkinChecker. Furthermore, each image has corresponding metadata containing external information about the neonate, including the total serum bilirubin (TSB) blood value used as ground truth for bilirubin prediction.

The dataset is collected from 141 unique neonates, where four images were taken at different ranges:

- close range (flash)
- medium range (flash)
- long range (flash)

- long range (without flash)

The SkinChecker is manually segmented using the annotation program RectLabel[1]. All SkinCheckers are then cropped out and saved. The dataset is cleansed, removing all neonates with missing TSB measurements (i.e., blood sample). In total eight neonates were removed, leaving a new dataset with 133 newborns.

The images were taken by a Samsung Galaxy S7 with a 12 Mega Pixel camera, yielding 4032x3024 resolution images. A tungsten light bulb (CIE A reproduced light) was placed next to the neonates illuminating the SkinChecker in all images. Each image is provided with the following corresponding metadata:

- Age (hours)
- Birth weight (grams)
- Ethnicity
- total serum bilirubin - Target value
- transcutaneous bilirubin (TcB)

Ethnicity: Since the collection of data was conducted at St. Olav's Hospital, without focusing on gathering a wide variety of ethical backgrounds, the dataset consists of 92% Caucasian neonates. Additionally, the remaining 8% are annotated as non-Caucasian, and can therefore not be assigned to any specific race. This study, therefore, concentrates on the prediction of bilirubin on Caucasian neonates, disregarding the 8% non-Caucasians.

Age & Weight: As discussed in background, due to the accumulation of bilirubin over time, some of the metadata is vital for the prediction of bilirubin concentration. In addition to the raw image, we use two a priori known features; Age (hours) and body weight at birth (grams). The treatment chart given by Fig. 2.3b indicates that both age and weight influence bilirubin concentration, thus implying the importance of the features.

Total Serum Bilirubin: TSB is the measured concentration of bilirubin obtained by a blood sample. It is seen as the *gold standard* and is used to determine if a patient requires treatment. TSB is used as the target value in the prediction of bilirubin.

Transcutaneous Bilirubin: transcutaneous bilirubin (TcB) is the measured skin reflectance given by a bilirubinometer. In this study, the Dräger JM-103 (section 2.3.1) was used to measure neonates' TcB, and the data is provided in the metadata file. The TcB is used to compare the proposed solutions to the state-of-the-art bilirubin prediction machines.

[1]`https://rectlabel.com/`

Due to the redefinition of the problem, focusing only on Caucasian neonates, non-Caucasian data had to be discarded. In total 12 children were removed, giving a new dataset of **121 children** (484 images in total). We refer to this new dataset as the Caucasian jaundice dataset.

3.2 Color Correction

The starting point of this thesis was to find an off-the-shelf color correction algorithm to create *standardized* skin images for the jaundice detection algorithm. However, due to related literature's use of high-quality cameras (e.g., NIKON D70), we hesitate to apply state-of-the-art color correction algorithms blindly and believe additional testing on smartphone camera images is required.

Taking into consideration the end-goal - color correction used in fieldwork - we require the color correction algorithm to be device independent and show high robustness towards illumination. We hypothesize that the relationship between camera RGBs and CIE L*a*b* triplets are dependent on the device used to capture the image and the light source illuminating the target object. As part of the thesis, we propose two separate color correction frameworks that combine previously well-established color correction solutions, to create robust, device independent color corrected images. The first framework is based on polynomial color correction (PCC), assembling widely used polynomial extensions in a combination of internal methods. The second, is reliant on the discoveries made by Finlayson et al. (2015)), mixing their root-polynomial (RPCC) methods.

Finally, we test the novel Gaussian process regression (GPR) for color correction. Due to its extreme modeling flexibility, and added noise kernels, we believe the GPR may compete and even outperform state-of-the-art approaches, creating highly complex models without overfitting.

3.2.1 Color Correction Framework Solutions

Literature describes a wide range of polynomial and root-polynomial fits to perform color correction. We are, however, unaware of previous work that proposes a solution as to which extension is to be applied in a given scenario. To overcome the practical issues related to color correction, we create two color correction frameworks. The frameworks, polynomial color correction framework (PCCF) and root-polynomial color correction framework (RPCCF), are implementations of widely used polynomial and root-polynomial extensions. The goal of the frameworks is to find the most appropriate RGB-extension fit for any given camera and illumination setting.

The frameworks are defined by their set of extension operators that transform sensor RGB values to m-element column vectors of an arbitrary combination or power. We will refer to these extension operators as **internal methods** θ_k (e.g. $(r, g, b)^T \xrightarrow{\theta_k} \hat{\rho}_2(r, g, b, r^2, g^2, b^2)^T$).

A framework builds a color correction model for each of its internal methods (i.e. polynomial expansion) using least-squares regression to solve Eq. 2.7, through Eq. 2.9, where P is replaced with the $N \times m$ extended color response of N surfaces ($\hat{P}$ or $\bar{P}$). Doing so, results in an $m \times 3$ matrix **M**, mapping $\hat{\rho} \xrightarrow{\mathbf{M}} (L*, a*, b*)$.

Each internal method is evaluated using leave-one-out cross-validation. That is, for each internal method a color correction model is built using all but one of the color patches from the color checker. The model is then tested on the remaining patch (i.e., predicting the CIE L*a*b* value and calculating error). This process is repeated for all color patches on the color checker, and the ΔE^*_{ab} is calculated for each pass.

We choose to separate PCC and root-polynomial color correction (RPCC) in case one of them is, by a large margin, superior to the other. Additionally, having fewer internal methods reduces the computational complexity of the frameworks. It is highly expensive to calculate the average cross-validation error for all color patches and all internal methods. The repeated cross-validation is one of the bigger drawbacks of the framework approach. It is important to stress that these frameworks are only implementations of previously known methods and that the semi-novelty lies in the leave-one-out comparison to select the scenario dependent best fitting RBG-extension.

Polynomial Color Correction Framework Solution (PCCF)

We hypothesize that the optimal polynomial RGB-extension is dependent on camera sensors and illumination sources. However, there is no a priori knowledge that indicates the relationship between device and light, and by extension, which polynomial fit yields the best mapping.

To create the set of polynomial extension operators (i.e., the set of internal methods), we apply an iterative scaling approach, increasing the complexity of internal methods for each iteration. We start off by adding a constant term to the linear color correction (LCC):

$$\rho = (r, g, b)^T \xrightarrow{\theta_1} \hat{\rho}_1 = (r, g, b, 1)^T \tag{3.1}$$

While not a significant alteration, the added term gives the least-squares regression some leeway when finding the best mapping. Some may argue the added constant is in fact not a polynomial extension of the LCC. However, we believe the approach is worth looking into, and define it as a polynomial extension of degree 1.

Continuing, we follow the polynomial orders, and combinations of R, G, and B, and create the following collection of RGB-extension operators. The first subscript denotes the polynomial order, and the second subscript denotes the method id within the polynomial order.

$$\hat{\rho}_{1,1} = [r, g, b, 1] \tag{3.2}$$

$$\hat{\rho}_{2,1} = [r, g, b, rg, rb, gb, 1] \tag{3.3}$$

$$\hat{\rho}_{2,2} = [r, g, b, r^2, g^2, b^2, 1] \tag{3.4}$$

$$\hat{\rho}_{3,1} = [r, g, b, rg, rb, gb, r^2, g^2, b^2, rgb, 1] \tag{3.5}$$

$$\hat{\rho}_{3,2} = [r, g, b, r^2g, r^2b, g^2r, g^2b, b^2r, b^2g, rgb, 1] \tag{3.6}$$

$$\begin{aligned}\hat{\rho}_{3,3} = [&r, g, b, rg, rb, gb, r^2, g^2, b^2,\\ &r^3, g^3, b^3, rgb, 1]\end{aligned} \tag{3.7}$$

$$\begin{aligned}\hat{\rho}_{3,4} = [&r, g, b, rg, rb, gb, r^2, g^2, b^2,\\ &r^2g, r^2b, g^2r, g^2b, b^2r, b^2g, rgb, 1]\end{aligned} \tag{3.8}$$

$$\begin{aligned}\hat{\rho}_{3,5} = [&r, g, b, rg, rb, gb, r^2, g^2, b^2,\\ &r^2g, r^2b, g^2r, g^2b, b^2r, b^2g, r^3, g^3, b^3, rgb, 1]\end{aligned} \tag{3.9}$$

The best internal method, for an arbitrary image, is chosen using leave-one-out cross-validation. The image is then color corrected using Eq. 2.7 with $\hat{P}$ replacing P, the $N \times m$ internal method polynomial color response of the N color surfaces.

Root-Polynomial Color Correction Framework Solution (RPCCF)

For a fixed exposure, PCC has shown significantly better results than a 3×3 LCC. However, as pointed out by Finlayson et al. (2015), exposure changes the vector of polynomial components in a nonlinear way, resulting in hue and saturation shifts. Their solution, RPCC, claims to fix these issues by expanding the RGB terms with root-polynomial extensions instead.

We adopt their idea and create the RPCCF. The framework includes the root-polynomial extensions suggested in their paper, where the subscript of $\bar{\rho}$ denotes the root-polynomial order.

$$\bar{\rho}_2 = [r, g, b, \sqrt{rg}, \sqrt{rb}, \sqrt{gb}, 1] \tag{3.10}$$

$$\begin{aligned}\bar{\rho}_3 = [&r, g, b, \sqrt{rg}, \sqrt{rb}, \sqrt{gb}, \sqrt[3]{rg^2}, \sqrt[3]{rb^2}, \sqrt[3]{gr^2}, \sqrt[3]{gb^2}, \\ &\sqrt[3]{br^2}, \sqrt[3]{bg^2}, \sqrt[3]{rgb}]\end{aligned} \tag{3.11}$$

$$\begin{aligned}\bar{\rho}_4 = [&r, g, b, \sqrt{rg}, \sqrt{rb}, \sqrt{gb}, \\ &\sqrt[3]{rg^2}, \sqrt[3]{rb^2}, \sqrt[3]{gr^2}, \sqrt[3]{gb^2}, \sqrt[3]{br^2}, \sqrt[3]{bg^2}, \sqrt[3]{rgb}, \\ &\sqrt[4]{r^3g}, \sqrt[4]{r^3b}, \sqrt[4]{g^3r}, \sqrt[4]{g^3b}, \sqrt[4]{b^3r}, \sqrt[4]{b^3g}, \\ &\sqrt[4]{r^2gb}, \sqrt[4]{g^2rb}, \sqrt[4]{b^2rg}]\end{aligned} \tag{3.12}$$

The best internal method, for an arbitrary image, is chosen using leave-one-out cross-validation. The image is then color corrected using Eq. 2.7 with $\bar{P}$ replacing P, the $N \times m$ internal method root-polynomial color response of the N color surfaces.

3.2.2 Gaussian Process Regression Solution

GPR is a very flexible machine learning technique that requires no explicit tuning of hyperparameters (other than the choice of kernel). Due to its flexibility and generalization capabilities (through the use of noise kernels), we hypothesize that GPR can be used as a robust and accurate method for color correction.

For color correction, we train three separate Gaussian processes (GPs), one for each color channel (R, G, B). To predict a new color, the three GPR models take the RGB input color and predicts their respective CIE L*a*b* intensity value (e.g., L* or a* or b*). The three individual results are combined to create the new CIE L*a*b* color coordinate.

The GPR implementation is initiated with a prior's covariance, specified by a kernel object. The hyperparameters of the kernel are optimized using gradient-descent on the marginal likelihood function during fitting, equivalent to maximizing the log of the marginal likelihood (LML). This property makes GPR superior to other supervised learning techniques, as it avoids heavy computational validation approaches, like cross-validation, to tune its hyperparameters. The LML may have multiple local optima, and trail and error testing is employed to verify that the optimal or close to the optimal solution is found by starting the optimizer repeatedly.

Choosing Kernel

GPR is a general method that can be extended to a wide range of problems. As discussed in background, to implement GPs for regression purposes, the prior of the GP needs to be specified by passing a kernel.

Widely used kernels are empirically tested; *RBF*, *Matérn*, and *rational quadratic (RQ)*. Additionally, combinations of *Constant*- and *White kernel* are added to the kernel function, to find the best performing kernel composition.

3.2.3 Color Correction Evaluation Methods

In every image a color checker is placed near-center of the camera frame. For each individual color (on the in-scene color checker), we exclude 10% of the sample area on all sides (Fig. 3.2), to eliminate noise from potentially faulty segmentation, and sample the average RGB value. The resulting average RGBs and corresponding CIE L*a*b* values (from the color checker) are used to create color correction models. To evaluate the performance of a model, we use three different evaluation methods; EM 1, EM 2, and EM 3. A common evaluation approach is to asses each model using leave-one-out cross-validation. However, highlighting the performance of a model when invoking different constraints may be valuable.

The following evaluation methods (EM) are:

- EM 1: All colors included
- EM 2: Semi-4-fold cross-validation
- EM 3: Leave-one-out cross-validation

EM 1: All Colors Included

The first evaluation method (EM 1) allows models to train on all colors. It gauges each method's ability to create color correction models that map all input colors to corresponding target colors. The evaluation method rewards overfitting but is included to expose methods that indeed overfit and to reveal methods' performance when all data is available for model fitting.

EM 2: Semi-4-Fold Cross-Validation

The semi-4-fold cross-validation (EM 2) is added to expose models that do not generalize well. Four randomly selected color patches are removed, thereby eliminating a considerable amount of 'known' target colors in the CIE L*a*b* color space. The total

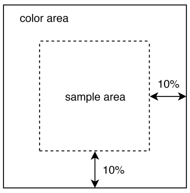


Figure 3.2: Sample area of a color patch: Each color is sampled, excluding 10% of the interest area to eliminate noise from potentially faulty segmentation.

error is calculating by aggregating the prediction error on the four excluded colors, and a generalized model should be able to predict unseen colors with minimal error.

We call it semi cross-validation as EM 2 does not exhaustively run through all selection possibilities, seeing that $\binom{N}{4}$ possibilities are too many. The evaluation method is, thus, only meant to give supplementary results, and for the semi-4-fold cross-validation, we perform the four random color selections 50 times, averaging out the scores.

EM 3: Leave-One-Out Cross-Validation

The main evaluation method (EM3) performs leave-one-out cross-validation and is an iterative approach where one color patch is withheld every run. For each iteration, the remaining colors are used to build a color correction model, which predicts the CIE L*a*b* value of the withheld color patch. By the end of the iteration, all color patches have been withheld and the mean ΔE^*_{ab} error is calculated.

It is worth noting that leave-one-out cross-validation (EM3) is, from a research perspective, the most reliable method of evaluation. It is both deterministic, and unbiased and is, therefore, the primary evaluation metric. EM1 and EM2 are merely supplementary evaluation methods that highlight different properties of the color correction solutions.

3.2.4 Experiments

To measure the performance of the proposed solutions; three separate experiments are set forth. For each experiment, all color correction models are evaluated according to evaluation methods from section 3.2.3.

3.2.4.1 Experiment I: Color Correction Using SpyderCHECKR, Evaluated on SpyderCHECKR

The purpose of experiment I, is to evaluate the color correction methods in terms of general color correction. That is, to assess each method's performance when operating on a diverse color checker displaying a wide range of colors. The experiment makes use of a SpyderCHECKR 24, containing 24 of the most common spectrally engineered colors, including a grayscale.

For a controlled environment, the SpyderCHECKR 24 is placed in a viewing box (Raw images used, Fig. B.1). The viewing box is imaged with an iPhone 6s, in three different light setting reproduced by CIE illuminants; CIE A, CIE D65 and CIE F11 (Table 3.1). Manual segmentation of the 24 color patches is performed, and the in-scene color values are compared to CIE L*a*b* target values (appendix A).

For each image, LCC, PCCF, RPCCF, and GPR are evaluated in terms of EM 1, EM 2, and EM 3. It is worth noting that both EM 1 and EM 3 are deterministic, while EM 2 is dependent on the four randomly selected color patches that are removed.

ID	CIE Illuminant	Color glow	Temperatur (K)	Description
Image 1	A	🟧	~ 2856	Incandescent / Tungsten
Image 2	D65	⬜	~ 6504	Noon Daylight
Image 3	F11	🟧	~ 4000	Philips TL84, Fluorescent

Table 3.1: Different illumination settings used in experiment I and experiment II. The light sources reproduce CIE illuminants, and with the use of a viewing box provide controlled environments.

3.2.4.2 Experiment II: Color Correction Using SkinChecker, Evaluated on SpyderCHECKR

The SkinChecker has a low color diversity, containing colors in proximity to human skin (mostly red-yellow) (Fig. 2.7). To see the color correction method's ability to predict unseen colors; the models are trained on a SkinChecker and evaluated on a SpyderCHECKR 24. If a solution can build a color correction model from a small set of colors, and still predict a diverse color sub-space, the model must be generalized. The experiment is designed to expose overfitting, and award the methods that create general color correction models when available color patches are scarce.

A SpyderCHECKR 24 and SkinChecker are placed, side by side, in a viewing box (Raw images used, Fig. B.2). Again, an iPhone 6s is used to capture the images in CIE A, CIE D65, and CIE F11 reproduced illumination (Fig. 3.1). Each method is only evaluated in terms of EM 1, where the errors are calculated on the SpyderCHECKR 24. Since models are trained on one color checker and evaluated on another, EM 2 and EM 3 cannot be used.

Regarding the SkinChecker, it is important to note that only one gray color patch (from the 20 equal grays) is used. Only using one of the gray patches, is to not skew the entire color correction towards that particular gray color.

3.2.4.3 Experiment III: Color Correction Using SkinChecker, Evaluated on SkinChecker

The experiment uses the jaundice dataset (section 3.1), containing images captured in uncontrolled environments during fieldwork, and tends to be inconsistent. When solving the jaundice detection problem, a solution cannot condition on controlled test environments. To simulate real life, we evaluate the color correction methods on the jaundice dataset, collected by fieldworkers for the intended use in jaundice prediction.

Based on the sampled data and the corresponding CIE L*a*b* triplets, each color correction solution builds a model, evaluated according to EM 1, EM 2, and EM 3. Additionally, a modification of EM 2 is introduced; **Evaluation Metric 4 (EM 4)**. EM 2 randomly excludes four color patches from the color checker where all color patches are candidates in the selection. However, we are interested in the method's prediction ability on skin color. Therefore, when using EM 4, the selection candidates are reduced to the set of **skin related** color patches (i.e., all non-gray colors). The EM 4 results should indicate which color correction solution is most appropriate for skin color correction during fieldwork.

The set of the SkinChecker color patches is listed in appendix A.

3.3 Bilirubin Prediction

The overall goal of this thesis is to assess machine- and deep learning approaches in the prediction of bilirubin concentration using human skin color, and available metadata (age and weight). Using the Caucasians jaundice dataset; three regression models are proposed to predict TSB. Since the solution only provides a *prediction* of TSB, based on camera skin color measurements, we refer to the solution output as **MobileCam-estimated bilirubin concentration (MCB)**.

This section presents the data preprocessing applied to the Caucasians jaundice dataset, the three proposed regression models, and the experiments conducted to evaluate them.

3.3.1 Data Preprocessing

When available data is limited, machine learning algorithms often struggle to find appropriate data distributions. With 484 images in the Caucasian jaundice dataset, preprocessing becomes a necessity to fully utilize all available features. To ensure device and illumination independent images, a SkinChecker, in combination with GPR, is used to perform color correction on all images. The normalization and standardization techniques are applied to aid the learning algorithms converge faster.

Weight: Having analyzed the dataset and talked to domain experts, the weight distribution is transformed to a normal distribution around its mean, shifted to zero. Following the *3-sigma rule*, 68% of the weight values are, after transformation, within one standard deviation σ and 95% are within 2σ, from the mean. Transforming to a normal distribution is reasoned on the fact that weight (at birth) is a natural phenomenon and usually follows a normal distribution. Figure 3.3 shows the performed distribution transformation of both age and weight.

Age: Using domain knowledge about bilirubin concentration and its accumulation and decay over time, age is logarithmically (using the *generalized logistic function*, also known as Richards' curve (Richards (1959))) scaled between zero and one.

The logarithmic scaling is applied to approximate the naturally occurring relationship between bilirubin concentration and time, following the treatment chart (Fig. 2.3b. Values from 0 to 100 hours are spread out in the distribution, and any value above 100 is squeezed close to 1. The generalized logistic function is given by:

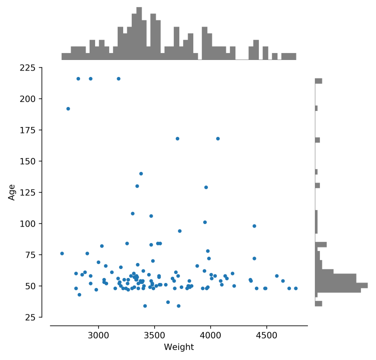
$$Y(t) = A + \frac{K - A}{(C + Q * e^{-B(t-M)})^{\frac{1}{v}}} \tag{3.13}$$

where A and K are its lower and upper asymptote, and B its growth rate. v decides where the function's maximum growth occurs, and Q, C, and M are constants. Figure 3.3 shows the performed distribution transformation of both age and weight.

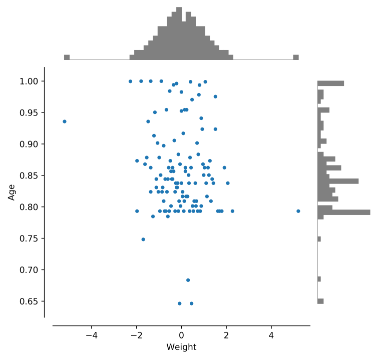
Images: All skin patches from within the SkinChecker are manually segmented after color correction (Fig. 3.1c). Pixel values are then min-max scaled from 0 - 255 to 0 - 1.

3.3.2 Experimental Setup

The Caucasians jaundice dataset is used to build prediction models through supervised learning. All images are labeled with measured TSB (blood value), used as ground truth during model fitting. Before training the regression models, a randomly chosen test set, τ, is extracted from the Caucasian jaundice dataset, following the hold-out test set principle discussed in section 2.7.3.2. After extraction, τ contains images and



(a) Unscaled distribution



(b) Scaled distribution

Figure 3.3: Before and after age and weight parameter distribution. **Figure 3.3a:** Unscaled distribution of the dataset, with respect to age (hours from birth) and weight (grams). **Figure 3.3b:** The distribution of the dataset after scaling is applied. Weight (grams) is normal distributed, and age (hours from birth) is logarithmically scaled.

corresponding metadata for 37 children (148 images) and accounts for 30% of the available data. The test set is used to evaluate the final models using the following evaluation metrics.

Evaluation Metrics

To evaluate the bilirubin regression models, two evaluation metrics are used; Pearson correlation coefficient, and TSB - MCB[2] difference.

The **Pearson correlation coefficient** (Rodgers and Nicewander (1988)) between ground truth (TSB) and prediction (MCB), denoted r, measures the strength and direction of the linear relationship between two variables and is given by:

$$r_{x,y} = \frac{\sum((x-\bar{x}) * (y-\bar{y}))}{\sqrt{\sum(x-\bar{x})^2 * \sum(y-\bar{y})^2}} \tag{3.14}$$

where $X = x_1, ..., x_n$ is a set of n target values, $Y = y_1, ..., y_n$ the set of predictions. $\bar{x}$ and $\bar{y}$ are the sample means of X and Y respectively.

[2]MobileCam-estimated bilirubin (MCB) concentration is measured bilirubin concentration, predicted using mobile camera

In a scatter plot, where TSB runs along the y-axis, and MCB the x-axis, any deviation from the identity line (i.e., $y = x$) indicates a prediction error. We create a regression line based on the prediction vs. ground truth data points to show the linear relationship between the model prediction (MCB) and TSB.

The **TSB - MCB difference** quantitatively measures the magnitude of the prediction error. To visualize the difference; a Bland-Altman plot is implemented. The plot reveals trends in the predictions, often highlighting under- or overestimation of the target value.

Dräger JM-103 (section 2.3.1) is a state-of-the-art bilirubinometer, which as discussed in background, has achieved close to 95% correlation on Caucasian neonates. In the jaundice dataset, each observation is accompanied with a JM-103 TcB measurement, which all MCB results are compared to. While not an evaluation metric in self, it gives an indication as to how well the machine learning solution performs.

Input Values

The theory of color correction indicates that the relationship between R, G, and B (e.g., R*G, R*B, G*B) is valuable information. We denote the extension operator that transforms an RGB-vector to an m-element column vector with added polynomial terms as $\theta_{k,i}$ where k represents the polynomial degree, and i represents the method id (Eq. 3.2)

Hypothesizing that extended RGB information might be valuable for bilirubin prediction, different polynomial RGB-extensions are tried as input. Input varies from r, g, b (i.e. $k = 1, i = 1$) to $r, g, b, rg, rb, gb, r^2, g^2, b^2, 1$ (i.e. $k = 3$, $i = 1$). We are uncertain as to how a neonate's age and weight contribute to the concentration of bilirubin, and thus try training each model both with and without these parameters.

3.3.3 Solutions

For the measurement of MobileCam-estimated bilirubin concentration (MCB), three different solutions are proposed, each evaluated on the randomly selected test set τ. The first solution is based on the expansion of Support Vector Machines (SVMs); support vector regression (SVR). A fully-connected feed-forward neural network is then explored, using average RGB color values and their polynomial extensions. Lastly, looking for spatial features in color corrected skin patches, a convolutional neural network (CNN) is tested.

While each solution implements different regression techniques, all are evaluated using the same test set and evaluation methods. In the following sections, we go into detail on the individual solutions.

3.3.3.1 Solution I: Support Vector Regression

When data is scarce, SVR is known to provide high-quality results and is generally robust against outliers. Since SVR is computationally fast, it is an efficient way to gauge the possibility of success, when no prior domain knowledge is known. The purpose of this experiment is to set a baseline for the jaundice prediction results and assess the possibility of finding any relationship between skin color and bilirubin concentration.

As discussed in section 2.7.2, preprocessing/scaling input data is essential to ensure that features with high absolute values do not overpower numerically small features. Also, the use of both stationary and non-stationary kernels requires the data to be normalized, due to the translation issue a non-stationary kernel face. From the color corrected skin patches, the average RGB values are extracted and normalized between 0 and 1. The neonates' age are log distributed between 0 and 1, and the weights are standardized to a normal distribution.

Implementation

We use the scikit-learn implementation of SVR[3], based on Chang and Lin (2011). The framework, written in Python, implements most state-of-the-art methods and kernels, and is, thus, robust and flexible. The framework builds an optimal prediction model using different input values and hyperparameters in a k-fold grid search. The final model is fitted and used to generate predictions on the test set τ.

Hyperparameter-Tuning

Four common kernels are tested; Linear, polynomial, radial-basis function (RBF), and sigmoid (Table 3.3). Additionally, the hyperparameters: C, γ, ϵ, $degree$, $\theta_{k,i}$, and A & W (Table 3.2), can be explicitly chosen. To find the best combination of hyperparameters and kernels, a grid-search with 5-fold cross-validation is employed, trying all possible combinations. The goal is to identify the optimal hyperparameter set tuning the SVR to accurately predict new/unknown data.

[3]`http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html`

Parameter	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
C	1	10	100	1000	2000	5000	-	-
γ	0.05	0.1	0.2	0.5	0.7	0.8	0.9	-
ϵ	0.005	0.01	0.05	0.1	0.2	0.3	-	-
degree	2	3	4	-	-	-	-	-
$\theta_{k,i}$	12	21	22	31	32	33	34	35

Table 3.2: Hyperparameters used in grid-search with SVR. See table 3.3 for kernel applicable hyperparameter. O_i represents $Option_i$, as all parameters have several options.

$\theta_{k,i}$ is the hyperparameter for RGB-extensions, C is a penalizing parameter for predictions outside the SVR margin of tolerance. ϵ specifies the epsilon-tube within which no penalty is given to a false prediction. Lastly, the γ is the kernel polynomial degree (only applicable for the polynomial kernel).

Kernel	C	γ	ϵ	degree	$\theta_{k,i}$
Linear	True	-	-	-	True
Polynomial	True	-	-	True	True
RBF	True	True	True	-	True
Sigmoid	True	True	True	-	True

Table 3.3: Kernel's applicable hyperparameter in the SVR grid-search. See table 3.2 for hyper-parameter notations.

3.3.3.2 Solution II: Fully Connected Neural Networks

There may be relationships between the features in the dataset that an SVR, with its kernels, is unable to detect.

Artificial neural networks (ANNs) are robust machine learning techniques. With their ability to learn complex features they can, as opposed to classical approaches, swiftly extract prominent relationships between input data. Neural networks are implemented to better understand the problem domain and to see if there are any higher complexity relationships in the data. To ensure that each input parameter is equally accounted for, all data is normalized as in *Experiment I*.

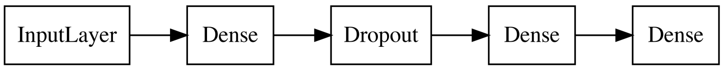


Figure 3.4: An abstraction of the neural network model.

Implementation

The neural network solution relies on Tensorflow and Keras. Tensorflow[4] is one of the most used ANN frameworks today and is supported by one of the largest online communities for deep learning. On top of Tensorflow, the high-level library Keras[5] creates an easy-to-use interface. Keras is implemented in Python and is designed for rapid experimentation with ANNs. Its focus is on user-friendliness, modularity, and extensibility. Since Keras is an official part of Tensorflow, it additionally offers the full low-level functionality of Tensorflow.

A grid-search with 5-fold cross-validation is used to find the optimal combination of model architecture and hyperparameters. The resulting hyperparameters from the grid-search are used to build and train the final neural network model on all available training data (both training and validation set). It is important to note that the test set τ is not used at this stage, and will not be used until the final model evaluation. To verify that the final model is not under- or overfitted; a training process, using 30% of the training data as the validation set, is initiated. The training process is recorded and plotted in a loss/epoch graph.

To generate the actual results; the final model is applied to the held-out test set τ, where the model is evaluated in terms of the evaluation metric explained in section 3.3.2, Test Setup.

Architecture and Hyperparameter-Tuning

The grid-search is used to determine the following hyperparameters: *number of layers*, *nodes per layer*, *activation function*, *batch normalization*, *dropout*, *L2 regularizer*, *optimizer*, *batch size*, and $\theta_{k,i}$.

Two different grid-searches are performed: Coarse- and fine grid-search. The idea behind the coarse grid-search (table 3.4) is to find the general area of interest for the network architecture. When a rough network architecture is found, a finer grid-search (i.e. smaller incremental steps between hyperparameter options) (table 3.5) is performed to optimize the final model (Fig. 3.4).

[4]`https://www.tensorflow.org/`
[5]`https://keras.io/`

Parameter	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
Number of layers	1	2	3	-	-	-	-	-
Nodes per layer	5	10	20	40	80	100	150	-
Activation func	ReLU	Tanh	Sigmoid	-	-	-	-	-
Batch norm	0	1	-	-	-	-	-	-
Dropout	0	0.1	0.3	0.5	-	-	-	-
L2 regularizer	0	0.01	0.001	-	-	-	-	-
Optimizer	Adam	RMSProp	-	-	-	-	-	-
Batch size	1	2	5	-	-	-	-	-
$\theta_{k,i}$	12	21	22	31	32	33	34	35

Table 3.4: Hyperparameters used in the coarse grid-search with fully-connected neural networks. *Batch norm* is batch normalization and *activation func* is activation function. $\theta_{k,i}$ is the hyperparameter for input values. O_i represents $Option_i$, as all parameters have several options.

Parameter	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
Nodes first layer	65	70	75	80	85	90	-	-
Nodes final layer	18	19	20	21	22	-	-	-
Activation func	Tanh	-	-	-	-	-	-	-
Dropout	0	0.3	0.4	-	-	-	-	-
L2 regularizer	0.005	0.01	0.015	0.02	0.025	0.03	-	-
Optimizer	Adam	-	-	-	-	-	-	-
Batch size	1	-	-	-	-	-	-	-
$\theta_{k,i}$	35	-	-	-	-	-	-	-

Table 3.5: Hyperparameters used in the fine tuning grid-search with fully-connected neural networks. The choice of parameters is based on the coarse grid search. O_i represents $Option_i$, as all parameters have several options.

Batch normalization is used between each hidden layer or not at all (only applicable if batch size > 1). When used, batch normalization relieves the effects of the internal covariate shift (Ioffe and Szegedy (2015)). The batches are normalized after each hidden layer and applies a transformation maintaining the mean activation close

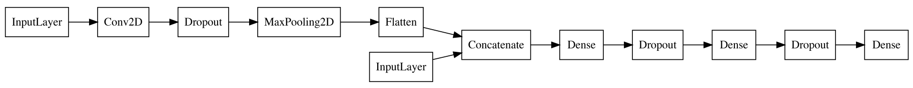


Figure 3.5: An abstraction of the convolutional neural network model.

to 0 and the standard deviation close to 1. Batch normalization allows the use of higher learning rates and weight initialization are less prominent.

Dropout: Dropout is a regularization technique where a subset of randomly selected neurons are deactivated during training. Dropout leads a more generalized model because it forces multiple nodes to learn the same concept.

L2-regularization is used to penalize large weights. 0.001, 0.01, and 0.02 regularization factors are tried empirically.

Training is performed with the RMSProp optimizer or the Adam optimizer (section 2.7.3). RMSProp employs a learning rate at $1e - 4$ and ρ at 0.9, while the Adam optimizer uses the same learning rate, but $beta_1$ at 0.9 and $beta_2$ at 0.999.

The neural network weights are initialized by sampling uniform distribution within a positive and negative limit:

$$Var(W) = \sqrt{\frac{6}{n_{in} + n_{out}}} \tag{3.15}$$

where n_{in} is the number of neurons feeding into the neuron in question, and n_{out} is the number of neurons the output is fed to. W is the initialization distribution for the neuron. This method is called the Glorot uniform initialize (Glorot and Bengio (2010)).

3.3.3.3 Solution III: Convolutional Neural Networks

The CNN solution is designed to assess if any spatial features in the skin patch may be valuable for bilirubin prediction accuracy. An example of a potentially relevant spatial feature is the color along blood veins.

To search for spatial features; a CNN is built. As with the prior solutions, the age and weight parameters are still relevant and added to the final fully-connected regression head (together with features from the convolutional layers).

The network consists of two separate input layers. The first input accepts images and is connected to a set of convolutional layers. The second input directly sends the age and weight parameters to a concatenation layer, where they are combined with the features from the convolutional layers. These aggregate features are then sent through a fully-connected feed-forward network regression head (Fig. 3.5).

Implementation

Using Tensorflow and Keras, 5-fold cross-validation is implemented, and different model architectures and hyperparameters are empirical determined. To verify that the final model is not under- or overfitted; a training process, using 30% of the training data as the validation set, is initiated. The training process is recorded and plotted in a loss/epoch graph. To generate the actual results; the final model is applied to the held-out test set τ, where the model is evaluated in terms of the evaluation metric explained in section 3.3.2, Test Setup.

Input-Values

The color corrected skin patch, resized to a 100 x 100 image array, is passed to the first input layer. Regarding the age and weight parameters, trail and error showed that a neonate's weight did not provide any useful information, and is thus not used (not sent to the network).

Architecture and Hyperparameter-Tuning

The grid-search implementation does not support multiple input layers, and thus all proposed architecture and hyperparameters are empirically tested to find the best performing model (verified through cross-validation).

The following paragraphs explain the hyperparameters and architectures of the two sub-networks:

- A convolutional network, with color corrected skin-patch as input.
- A feed-forward network regression head

The convolutional network: Other than small color differences, skin does not display many features. The convolutional part of the network does, therefore, not need to be complex, and the number of layers are limited to one or two convolutional layers. Additionally, the kernel size and number of kernels are adjustable parameters found empirically. All convolutional layers use the ReLU activation function (section 2.7.3), where max pooling- and dropout is optional.

The feed-forward network regression head concatenates the convolutional network and the age input. It consists of one or two hidden layers before a final output node provides the model prediction. The hidden layers follow the same principles as in *Experiment II*.

Hyperparameters To summarize the paragraphs above, the following hyperparameters are tuned through trail and error (table 3.6): *Kernels per layer*, *Kernel size*, *number of final fully-connected layers*, *nodes per fully-connected layer*, *activation function* *fully-connected layers*, *batch normalization*, *dropout*, *L2 regularizer*, *optimizer*, *batch size*.

Parameter	O_1	O_2	O_3	O_4	O_5	O_6	O_7
Kernels per layer	2	4	8	-	-	-	-
Kernel size	(3,3)	(5,5)	-	-	-	-	-
Number of fcl	1	2	3	-	-	-	-
Nodes per fcl	5	10	20	30	-	-	-
Activation func fcl	ReLU	Tanh	-	-	-	-	-
Batch norm	0	1	-	-	-	-	-
Dropout	0	0.3	0.5	-	-	-	-
L2 regularizer	0	0.01	0.001	-	-	-	-
Optimizer	Adam	RMSProp	-	-	-	-	-
Batch size	1	2	5	-	-	-	-

Table 3.6: Possible hyperparameters for the convolutional neural network. *fcl* represents fully-connected layer, *func* is short for function, and *norm* denotes normalization. O_i represents $Option_i$, as all parameters have several options.

Chapter 4

Results

This chapter presents the results from the conducted experiments explained in Chapter 3, and is divided into two sections; color correction and bilirubin prediction.

4.1 Color Correction

To answer the research questions; three separate experiments were performed, and the results are presented individually. Note that all error values presented are given as ΔE^*_{ab} in the CIE L*a*b* color space.

Table 4.1 is included to summarize the color correction evaluation methods (section 3.2.3) used in the experiments.

Evaluation method	Description
EM 1	All colors included
EM 2	Semi-4-fold cross-validation
EM 3	Leave-one-out cross-validation

Table 4.1: Displays evaluation methods used in the color correction experiments, which are fully described in section 3.2.3.

4.1.1 Experiment I: Color Correction Using SpyderCHECKR, Evaluated on SpyderCHECKR

In experiment I, color correction models are trained on a SpyderCHECKR 24 and evaluated on the same image. Since the SpyderChecker 24 contains diverse colors, the experiment tests the color correction methods in general color correction.

Table 4.2 shows the results from Experiment I, evaluating the linear color correction (LCC), polynomial color correction framework (PCCF), root-polynomial color correction framework (RPCCF) and Gaussian process regression (GPR) according to EM 1, EM 2, and EM 3. Three separate images were captured and color corrected in different CIE illuminant reproduced light (Fig. 2.2); CIE D65, CIE F11, and CIE A, reflected in the table by grouping results with corresponding illuminants. The results are derived by comparing the color corrected RGB responses with the SpyderCHECKR's ground truth CIE L*a*b* triplets.

The table indicates that the RPCCF is the best solution for general color correction in all light settings. Further, it shows that all solutions are struggeling to color correct in CIE A reproduced light.

	CIE D65			CIE F11			CIE A			
solution	EM 1	EM 2	EM 3	EM 1	EM 2	EM 3	EM 1	EM 2	EM 3	Avg. EM 3
LCC	6.93	8.22	8.04	6.99	8.58	8.09	9.35	12.0	10.83	8.99
PCCF	4.32	7.15	**6.49**	5.15	7.82	7.61	6.45	10.49	9.71	7.94
RPCCF	**3.94**	**5.73**	**6.49**	**4.76**	**6.96**	**7.09**	**5.40**	**8.40**	**8.05**	**7.21**
GPR	4.00	6.91	6.92	5.11	7.79	8.02	6.05	11.15	10.10	8.35

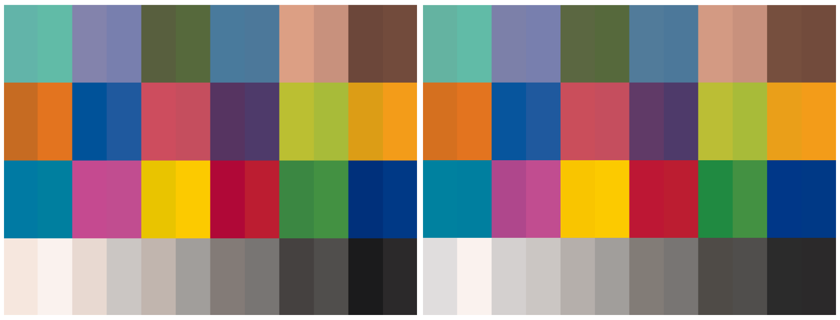
Table 4.2: Four color correction methods were trained on an imaged SpyderCHECKR 24 and evaluated on the same image. For each evaluation method, the best performing (lowest ΔE^*_{ab}) solution is highlighted in bold. The last column represents the average EM 3 ΔE^*_{ab} across all illuminants.

Table 4.3 presents the best performing internal methods from the proposed color correction frameworks (PCCF and RPCCF). The results indicate that regardless of illumination, both frameworks solve the SyderCHECKR 24 color correction using one preferred RGB-extension.

Framework	D65	size	F11	size	A	size
PCCF	$\hat{\rho}_{2,1}$	4x3	$\hat{\rho}_{2,1}$	4x3	$\hat{\rho}_{2,1}$	4x3
RPCCF	$\bar{\rho}_2$	7x3	$\bar{\rho}_2$	7x3	$\bar{\rho}_2$	7x3

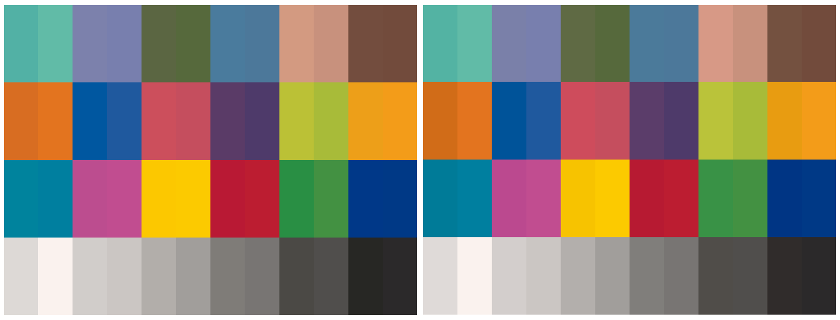
Table 4.3: Best performing internal methods of the PCCF and RPCCF for each imaged SpyderCHECKR in CIE D65, CIE F11 and CIE A reproduced illumination. The methods are ranked by their leave-one-out cross-validation results measured in ΔE^*_{ab}.

Figure 4.1 is a visualization of applied color correction in noon daylight (CIE D65). The images are reconstructions from both color correction and ground truth color values. Each color patch is divided into two parts, where the left-hand side is the color corrected color, and the right-hand side is the ground truth color. A color difference between the left and right color indicates that the color correction is not perfect.



(a) Color correction: LCC

(b) Color correction: PCCF

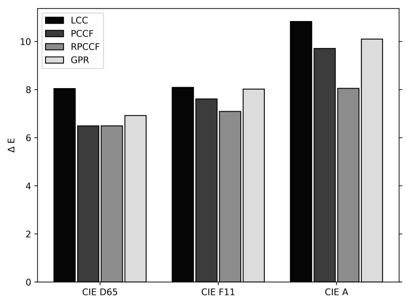


(c) Color correction: RPCCF

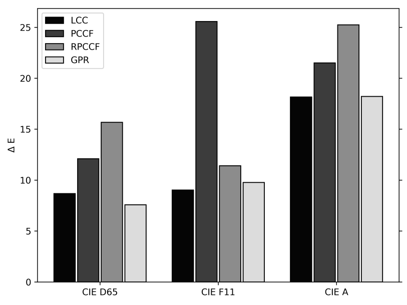
(d) Color correction: GPR

Figure 4.1: Color correction models are trained on an imaged SpyderCHECKR 24 and evaluated on the same image. The color checker is placed in a viewing box and captured in CIE D65 reproduced illumination. Each color patch, in the reconstructed image, is divided into *two parts*: **left** shows the color corrected color, **right** shows the ground truth target color. All images are recreated from CIE L*a*b* coordinates to RGB with CIE D65 illuminant.

Figure 4.2a illustrates the cross-validation (EM 3) ΔE^*_{ab} results from experiment I. The bar plot graphs are reiterations of the results given in table 4.2 and are only supplementary figures to highlight the performance differences.



(a)



(b)

Figure 4.2a: Bar plot visualization of leave-one-out cross-validation results (EM3) from Experiment I. **Figure 4.2b:** Bar plot visualization of leave-one-out cross-validation results (EM3) from Experiment II. Both bar plots are divided into three subsection, representing the results on each image in the CIE illuminant reproduced light; D65, F11, and A, respectively.

4.1.2 Experiment II: Color Correction Using SkinChecker, Evaluated on SpyderCHECKR

The LCC, PCCF, RPCCF, and GPR were tested by training color correction models on a SkinChecker and evaluated on a SpyderCHECKR 24. Both color checkers were imaged, side by side, in CIE D65, CIE F11, and CIE A reproduced illumination. Table 4.4 shows the results from Experiment II (visualized in Fig. 4.2b) for each CIE illuminant. The applied color corrections, for the noon daylight (D65) illuminated image, are shown in figure 4.4.

From the table it is worth noting that the general color correction solutions LCC, and GPR are performing well. Again we take note of the overall poor performance in the CIE A reproduced light.

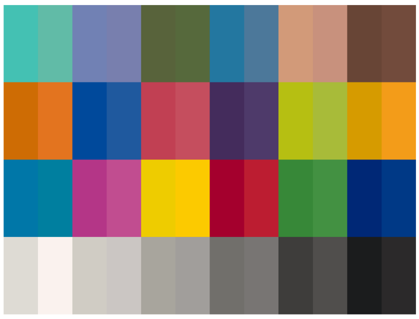
solution	CIE D65	CIE F11	CIE A	average
LCC	8.68	**9.02**	**18.16**	11.95
PCCF	12.09	25.57	21.50	19.72
RPCCF	15.67	11.41	25.24	17.44
GPR	**7.58**	9.77	18.21	**11.85**

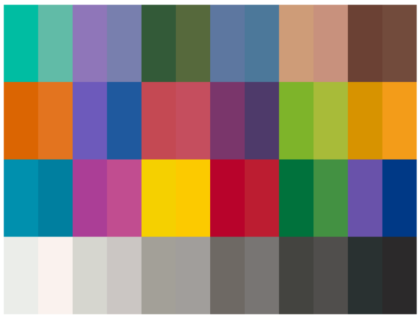
Table 4.4: The LCC, PCCF, RPCCF, and GPR are trained on a SkinChecker and evaluated on a SpyderCHECKR 24. The best performing (i.e. lowest ΔE^*_{ab}) solution in terms of EM 1 is highlighted in bold. The last column represents the average EM 1 ΔE^*_{ab} across all illuminants. All numerical values presented are results from EM 1.

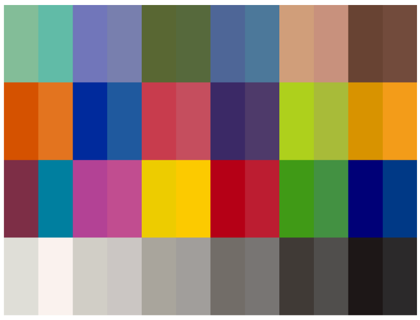
Table 4.5 presents the best performing internal methods from the proposed color correction frameworks. The key takeaway from these results is that the PCCF is using a different internal method than in experiment I. Further, both frameworks are using the same internal method regardless of illumination.

Framework	D65	size	F11	size	A	size
PCCF	$\hat{\rho}_{2,2}$	7x3	$\hat{\rho}_{2,2}$	7x3	$\hat{\rho}_{2,2}$	4x3
RPCCF	$\bar{\rho}_2$	7x3	$\bar{\rho}_2$	7x3	$\bar{\rho}_2$	7x3

Table 4.5: Best performing internal methods of the PCCF and RPCCF for each imaged SpyderCHECKR in CIE D65, CIE F11, and CIE A reproduced illumination. The methods are ranked by their leave-one-out cross-validation results measured in ΔE^*_{ab}.


(a) Color correction: LCC


(b) Color correction: PCCF


(c) Color correction: RPCCF

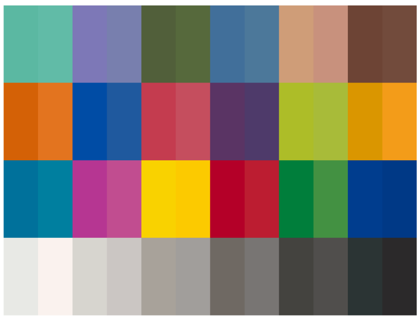

(d) Color correction: GPR

Figure 4.3: Color correction models are trained on an imaged SkinChecker and evaluated on a SpyderCHECKR 24. The color checkers are placed in a viewing box and captured in CIE D65 reproduced illumination. Each color patch, in the reconstructed image, is divided into *two parts*: **left** shows the color corrected color, **right** shows the ground truth target color. All images are recreated from CIE L*a*b* coordinates to RGB with CIE D65 illuminant.

4.1.3 Experiment III: Color Correction Using SkinChecker, Evaluated on SkinChecker

In experiment III color correction models are trained on a SkinChecker and evaluated on the same image. The models are evaluated on all images in the jaundice dataset, thus testing each solution in terms of skin color correction.

Table 4.6 shows the results from Experiment III, evaluating the LCC, PCCF, RPCCF, and GPR. The jaundice dataset is collected through fieldwork, and the light sources are not reproducible in terms of a single CIE illuminant. The results of EM 3 and EM 4 are visualized in figure 4.4.

The table indicates that both the PCCF and GPR are viable solutions when color correcting skin. Their results are approximately equal across all evaluation metrics.

solution	EM 1	EM 2	EM 3	EM 4
LCC	7.86	8.30	8.42	6.54
PCCF	2.46	**3.62**	**3.45**	**4.37**
RPCCF	3.79	4.71	4.74	5.07
GPR	**2.45**	3.73	3.53	4.40

Table 4.6: The LCC, PCCF, RPCCF, and GPR are trained on a SkinChecker and evaluated on a SkinChecker. The best performing (i.e. lowest ΔE^*_{ab}) solution is highlighted in bold.

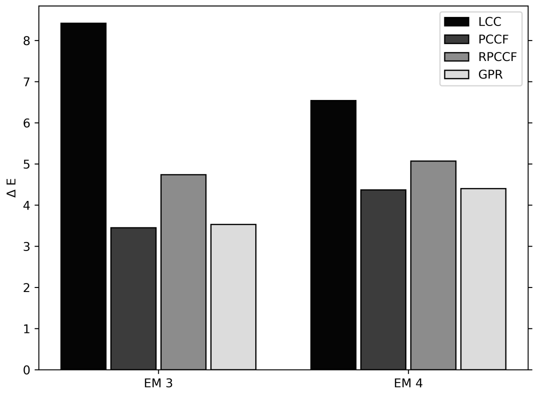


Figure 4.4: Bar plot visualization of leave-one-out cross-validation results (EM3) and semi-4-fold cross-validation on skin related candidates (EM 4) from Experiment III.

To give insight in the proposed color correction frameworks, we present the results for each internal method. Table 4.7 shows the internal results for the PCCF, where $\hat{\rho}_{3,1}$ is the best performing internal methods in terms of EM 3 (cross-validation). The RPCCF results are more polarized, where $\bar{\rho}_2$ outperforms the rivaling internal methods by a large margin. The distributions of selected internal methods are shown in figure 4.6 and highlight the success of $\hat{\rho}_{3,1}$ and $\bar{\rho}_2$. It is interesting to see that the center-most methods of the PCCF are achieving the best results.

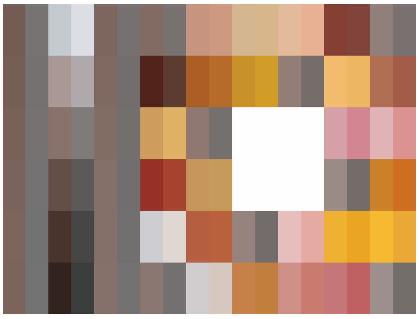
method	size	EM 1	EM 2	EM 3	EM 4
$\hat{\rho}_{11}$	4x3	4.28	4.61	4.84	**4.20**
$\hat{\rho}_{21}$	7x3	2.85	**3.66**	3.63	4.54
$\hat{\rho}_{22}$	7x3	2.94	3.68	3.69	4.63
$\hat{\rho}_{31}$	11x3	2.37	3.88	**3.53**	4.56
$\hat{\rho}_{32}$	11x3	2.40	3.75	3.77	4.74
$\hat{\rho}_{33}$	14x3	2.12	4.26	4.05	4.74
$\hat{\rho}_{34}$	17x3	1.98	5.51	5.04	8.03
$\hat{\rho}_{35}$	20x3	**1.73**	7.98	6.64	11.97

Table 4.7: All color correction evaluation method results for all internal PCCF methods. The best performing (i.e. lowest ΔE^*_{ab}) internal method is highlighted in bold.

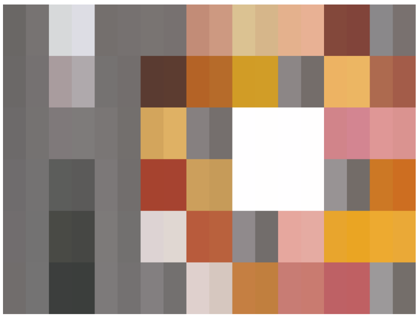
method	size	EM 1	EM 2	EM 3	EM 4
$\bar{\rho}_2$	4x3	3.79	**4.71**	**4.74**	**5.06**
$\bar{\rho}_3$	7x3	3.14	6.29	6.04	13.36
$\bar{\rho}_4$	7x3	**1.54**	23.85	13.73	63.57

Table 4.8: All color correction evaluation method results for all internal RPCCF methods. The best performing (i.e. lowest ΔE^*_{ab}) internal method is highlighted in bold.

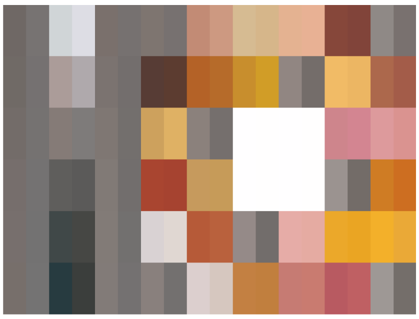
Four visualizations of color corrected SkinCheckers using LCC, PCCF, RPCCF, and GPR respectively, are illustrated in Fig. 4.5. The color corrected image is randomly selected from the jaundice dataset.



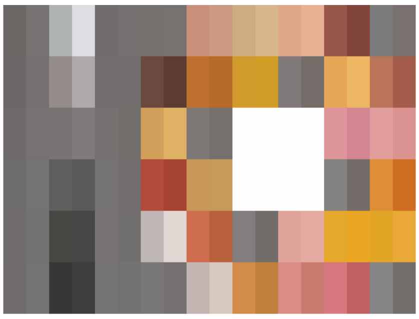
(a) Color correction: LCC



(b) Color correction: PCCF



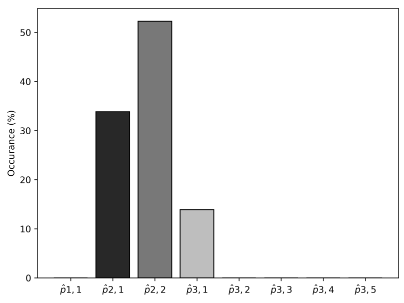
(c) Color correction: RPCCF



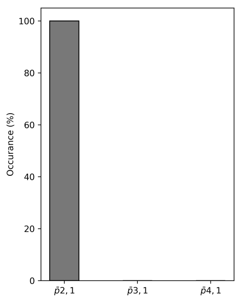
(d) Color correction: GPR

Figure 4.5: Color correction models are trained on an imaged SkinChecker and evaluated on the same image. Each color patch, in the reconstructed image, is divided into *two parts*: **left** shows the color corrected color, **right** shows the ground truth target color. All images are recreated from CIE L*a*b* coordinates to RGB with CIE D65 illuminant.

Figure 4.6 shows the two internal method distributions of the color correction frameworks (PCCF, RPCCF) in bar plots. It is important to note that the PCCF is alternating between three internal methods, while theRPCCF is only opting for one.



(a) PCCF internal method distribution



(b) RPCCF internal method distribution

Figure 4.6: The bar plot displays the distribution of selected internal methods. The plots show that PCCF alternates between three polynomial extensions, while the RPCCF only uses one.

4.2 Bilirubin Prediction

To solve the bilirubin prediction problem, we propose three machine learning based non-linear regression techniques; support vector regression (SVR), fully-connected feed-forward neural networks, and convolutional neural networks (CNNs). The presented results are produced by evaluating each model on the held out test set, τ, comparing the predicted MobileCam Bilirubin (MCB) measurment to the ground truth total serum bilirubin (TSB) value. Before conducting the experiments, all images are color corrected.

4.2.1 Support Vector Regression

After the grid search using k-fold cross-validation ($k = 5$), the following parameters were found to be optimal: **C** = 2000, $\epsilon = 0.3$, $\gamma = 0.9$, **kernel** = radial-basis function (RBF), RGB expansion $= \theta_{3,1}$. Only the log distributed age normalization adds value to the bilirubin prediction, and the neonate weights were not used.

Table 4.9 shows the final result of the SVR model, alongside the Dräger JM-103 bilirubinometer. In the experiment, JM-103 outperforms the SVR implementation with a 0.04 higher Pearson correlation coefficient.

Solution	Mean difference (TSB - MCB/TcB)	Std	**Corr**
SVR	26.09	22.33	**0.91**
JM-103	21.41	25.68	**0.95**

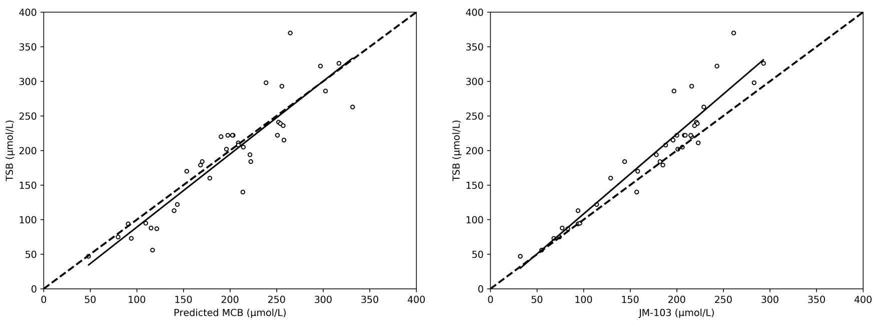
Table 4.9: Result of the Support Vector Regression model and the Dräger JM-103 bilirubinometer. Std represents the standard deviation from the mean difference (TSB - MCB/TcB), and the correlation is calculated using the Pearson correlation coefficient.

Figure 4.7 visualizes the linear relationship between prediction (MCB/ transcutaneous bilirubin (TcB)) and ground truth (TSB) for both the SVR model and the JM-103 bilirubinometer. The Dräger JM-103, underestimates TSB in the higher concentration ranges, indicated by the bold regression line and its divergence away from the identity line (Fig. 4.7b). The SVR model, while on average closer to the identity line, wrongly predicts target values by a larger margin on both sides (Fig. 4.7a).

The Bland-Altman plot (Fig. 4.8) further highlights these trends, where the Dräger JM-103 has a mean difference of +20 (bold line in figure 4.7b). The plot indicates that the JM-103 predictions, on average, are 20 µmol/L lower than the actual bilirubin concentration. It is very important to note that the difference is defined as ground truth minus prediction (TSB - prediction(MCB/TcB)). A difference larger than zero indicates a prediction less than the ground truth and vice versa ($diff > 0 \implies prediction < TSB$). Positive difference scores, therefore, lead to false negatives (i.e., stating that a sick patient is healthy). Such predictions are not tolerated in the medical community.

Ideally, the axes should be flipped to give an intuitive understanding of the visual representation (i.e., higher values indicate overestimations and vice versa). However, since other medical publications place TSB along the y-axis, and predictions along the x-axis, for comparison, we follow the same practice.

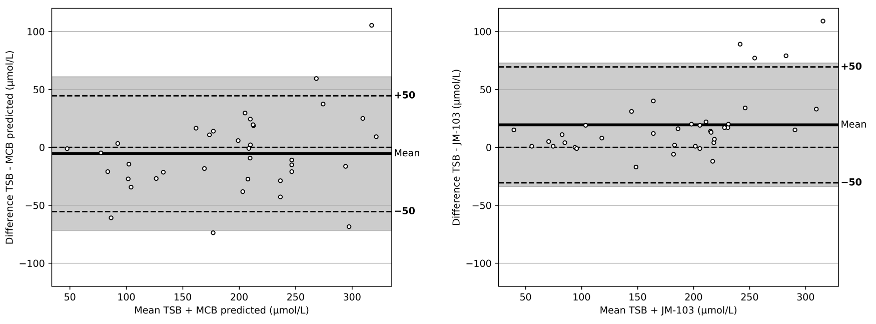
The SVR average mean difference is slightly below zero, illustrating overestimations of TSB. The predictions are, however, volatile with prediction errors both below and above ground truth.



(a) Linear regression of SVR MCB measurment

(b) Linear regression plot of Dräger JM-103 TcB measurment

Figure 4.7: Linear regression plot of MCB/TcB measured versus TSB measurements. The bold line shows the regression line derived from the predictions, and the dotted line is the identity line ($y = x$). The plot consist of data from evaluation of 37 children.



(a) Bland-Altman plot of SVR MCB measurment

(b) Bland-Altman plot of Dräger JM-103 TcB measurment

Figure 4.8: Bland-Altman (difference-plot) for the total population between measured MCB/TcB and TSB measurements. The gray area shows the 95% confidence interval from the mean prediction (bold line). Dotted outer lines shows margin of error[1]. The plot consist of data from the evaluation of 37 children.

[1]The margin of error is a recommended safety margin set by the hospital to make sure all children in the danger zone is correctly tested

4.2.2 Fully Connected Neural Networks

Finding the optimal neural network architecture requires rigorous trail and error experimentation. After a coarse grid search with 5-fold cross-validation, a more narrow search (based on the results from the coarse search) was conducted. The second grid-search, shown in table 3.5), deemed the following parameters to be prominent, and defines the final neural network model (Table 4.10):

Parameter	value
Epochs	500
Batch size	1
Optimizer	Adam
Input parameters	Median RGB & age
Input RBG extension	$\hat{\rho}_{3,5}$
First layer	
Nodes	70
Dropout	30%
Activation function	tanh
L2 Regularization (weights)	0.02
Second layer	
Nodes	19
Dropout	0%
Activation function	tanh
L2 Regularization (weights)	0.02

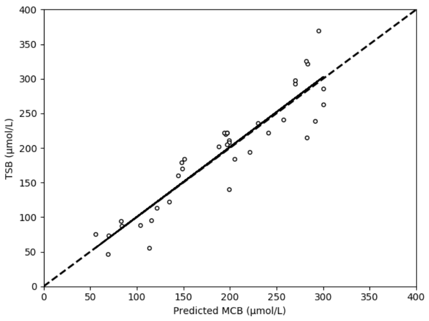
Table 4.10: The hyperparameters of the fine tuned neural network used to create the final results evaluated on τ.

Table 4.11 presents the results from the fine-tuned fully-connected neural network, alongside the Dräger JM-103. The bilirubinometer still achieves higher correlation than the neural network, having a 0.03 higher correlation coefficient. Additionally, the JM-103 predicts to the lowest 'mean difference' of the two (TSB - MCB/TcB), but is outperformed by the neural network in terms of standard deviation.

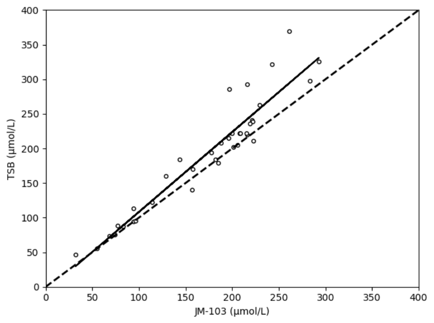
Solution	Mean difference (TSB - MCB/TcB)	Std	**Corr**
Neural network	25.65	17.45	**0.92**
JM-103	21.41	25.68	**0.95**

Table 4.11: The results of the fitted neural network model and the Dräger JM-103 bilirubinometer. Std represents the standard deviation from the mean difference (TSB - MCB/TcB), and the correlation is calculated using the Pearson correlation coefficient.

The linear relationship between prediction and ground truth blood values is plotted in figure 4.9. The fully-connected neural network is on average (regression line) close to the identity line. As with the SVR, it wrongly predicts the target values on both sides. These trends are further highlighted in the Bland-Altman plot (Fig. 4.10). Looking at the Bland-Altman plot, Dräger JM-103 has **four** predictions above the upper dotted line (+50), implying it evaluates a possibly sick patient to be healthy (false negative). The neural network on the other hand, only predicts **one** false negative. Again. it is important to remember the counter-intuitive axes, and that positive difference values imply underestimations.

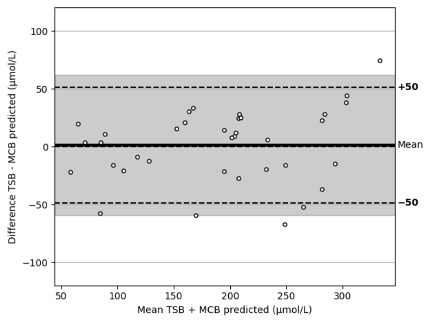


(a) Linear regression of neural network MCB measurment

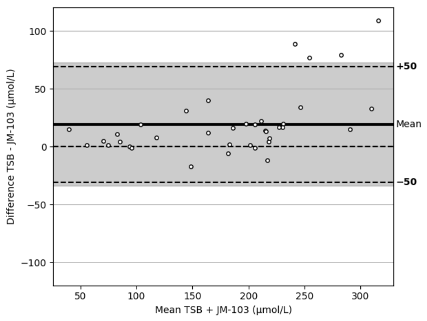


(b) Linear regression plot of JM-103

Figure 4.9: Linear regression plot of MCB/TcB measured versus TSB measurements. The bold line shows the regression line derived from the predictions, and the dotted line is the line of identity ($y = x$). The plot consist of data from evaluation of 37 children.



(a) Bland-Altman plot of neural network MCB measurement



(b) Bland-Altman plot of JM-103 TcB measurement

Figure 4.10: Bland-Altman (difference-plot), for the total population between measured MCB/TcB and TSB measurements. The gray area shows the 95% confidence interval from mean prediction (bold line). Dotted outer lines shows margin of error. The plot consist of data from the evaluation 37 children.

Figure 4.11 shows the mean-squared error loss history on a training set (70%) and a validation set (30%), where the tests set is still held out. The graph is generated from a dummy-training process to verify that the model found through the 5-fold cross-validation grid-search is adequately trained. The plot shows no indications of under- or overfitting, as the training- and validation loss output close to equal values.

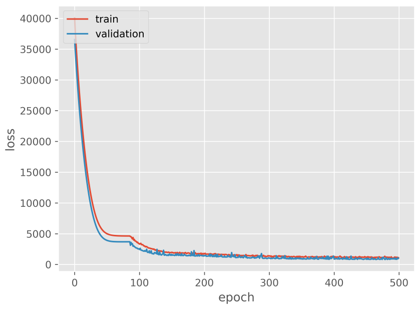


Figure 4.11: Loss history-plot, showing training vs. validation MSE loss. The validation set consist of 30% of the training data.

4.2.3 Convolutional Neural Networks

To look for spatially neighboring color features; a CNN was implemented. Empirical testing of different models and architectures was performed to find the optimal configuration. The final CNN is presented in table 4.12:

Parameter	value
Epochs	500
Batch size	1
Optimizer	Adam
Input parameters	RGB (100x100) & age
First convolution layer	
Kernel size	(3, 3)
Number of kernels	4
Dropout	10%
Activation function	ReLU
L2 Regularization (weights)	0.02
Max-pooling size	2
First fully-connected layer	
Nodes	25
Dropout	20%
Activation function	tanh
L2 Regularization (weights)	0.02
Second fully-connected layer	
Nodes	19
Dropout	0%
Activation function	tanh
L2 Regularization (weights)	0.02

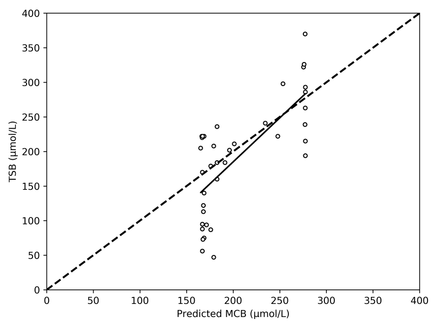
Table 4.12: The hyperparameters and architecture of the final convolutional neural network used to create final results evaluated on the test set, τ.

The final result of the CNN, alongside the Dräger JM-103 is presented in table 4.13. In the experiment, the Dräger outperforms the network with 0.22 higher correlation. Additionally, it outperforms the network on both mean difference and standard deviation.

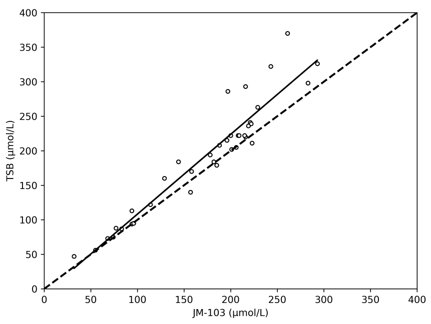
Solution	Mean difference (TSB - MCB/TcB)	Std	**Corr**
Convolutional Neural network	47.55	33.55	**0.73**
JM-103	21.41	25.68	**0.95**

Table 4.13: The result of the fitted convolutional Neural Network model and the Dräger JM-103 bilirubinometer. Std represents the standard deviation from the mean difference (TSB - MCB/TcB), and the correlation is calculated using the Pearson correlation coefficient.

The linear regression plot in figure 4.12a indicates that the CNN is unable to predict the entire range of bilirubin values. With so little information in images, it is hard to train the convolutional layers and shows why the correlation is so low.



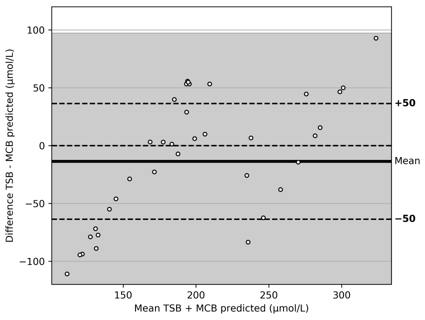
(a) Linear regression of CNN MCB measurment



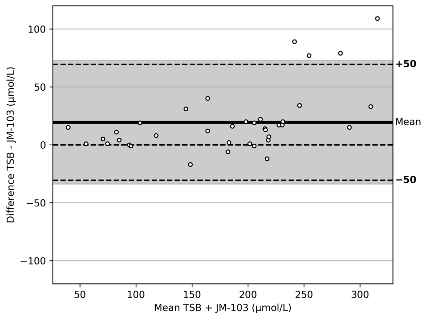
(b) Linear regression plot of JM-103

Figure 4.12: Linear regression plot of MCB/TcB measured versus TSB measurements. The bold line shows the regression line derived from the predictions, and the dotted line is the line of identity $(y = x)$. The plot consist of data from evaluation of 37 children.

The Bland-Altman plot further highlights the poor performance of the CNN. Both plots illustrate that the network is primarily predicting **two** values, with some predictions in between.



(a) Bland-Altman plot of CNN MCB measurement



(b) Bland-Altman plot of JM-103

Figure 4.13: Bland-Altman (difference-plot), for the total population between measured MCB/TcB and TSB measurements. The gray area shows the 95% confidence interval from mean prediction (bold line). Dotted outer lines shows margin of error. The plot consist of data from the evaluation of 37 children.

Regardless of the poor performance, the mean-squared error loss history plot, from figure 4.14, shows no indication of under- or overfitting. Again, the history is generated from a dummy-training process to verify that the model found through the 5-fold cross-validation grid-search is adequately trained. The training process was conducted using a 70% training set and a 30% validation set.

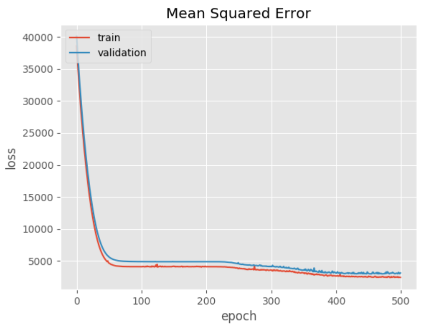


Figure 4.14: Loss history-plot, showing training vs validation MSE loss. The validation set consist of 30% of the training data.

Chapter 5

Discussion

This chapter will discuss and evaluate the results presented in Chapter 5 and reflect upon the strengths and weaknesses of the proposed solutions. To relate the findings to the two main research questions, we divide the chapter into two sections; color correction and bilirubin prediction.

5.1 Color Correction

In this thesis, we explore the use of computer vision and machine learning for the prediction of jaundice using embedded smartphone cameras.

The use of smartphones promotes widespread availability but comes at a cost, instrumental inconsistency. Camera sensors vary from model to model, and the color values captured are camera **dependent**. Additionally, controlling light, and the aggregate effect of light sources has proven to be a difficult task.

We perform color correction to overcome these issues, creating *standardized* skin images for the jaundice detection algorithm. Literature does not agree upon approaches for human skin color correction, and set forth the primary color correction research questions: *What is the most accurate and robust technique to perform human skin color correction for scientific use?*

The following sections present a discussion on the linear color correction (LCC), the proposed framework solutions, and the novel Gaussian process regression (GPR). Note that the color correction errors used in the discussion refer to the leave-one-out cross-validation results (EM3) unless otherwise specified.

5.1.1 Linear Color Correction (LCC)

To simplify the comparison of the proposed solutions, we use the simple $3x3$ LCC as an evaluation baseline. The LCC does not perform aggressive color correction and should, from a theoretical standpoint, build general color correction models that perform adequately in any scenario.

In experiment I, a SpyderCHECKR 24 is used to evaluate color correction methods on a diverse range of colors. The LCC achieves an average $\Delta E^*_{ab} = 8.99$. As mentioned in section 2.4.4, a $\Delta E^*_{ab} = 2.3$ color difference is regarded as 'just noticeable' and anything less than $1\Delta E^*_{ab}$ is imperceptible to the human eye. This implies that the LCC results in a highly visible color difference $(8.99 >> 2.3)$.

A key observation is that the lowest color correction error is achieved in CIE D65 reproduced light $(\Delta E^*_{ab} = 8.04)$, with close to identical results in fluorescent light (CIE F11: $\Delta E^*_{ab} = 8.09$). Interestingly, when the light source is reproduced from a CIE A tungsten light bulb, the model color corrects to a $\Delta E^*_{ab} = 10.83$ color error, approximately $3\Delta E^*_{ab}$ higher. A significant portion of this error is attributed to failed color correction on the grayscale of the color checker.

The idea behind experiment II is to gauge color correction solutions in general color correction (evaluated on a diverse range of colors) when models use a specialized color checker (i.e., smaller color subspace) as a reference point. Building a color correction model from the SkinChecker, evaluated on the SpyderCHECKR 24, the LCC achieves an average $\Delta E^*_{ab} = 11.95$. As expected, we see an increase in average error compared to Experiment I $(8.99 < 11.95)$. The SkinChecker does not contain dominant blue or green color patches, and thus, the model does not have any reference points for these during prediction. Nevertheless, looking at figure 4.3a, we see the LCC can recreate **all** colors in close proximity to the ground truth. From table 4.4 we, again, observe the influence of illumination and the significant error increase caused by the tungsten light bulb.

While being the least scientific in terms of execution, experiment III is arguably the most important experiment. The results average out 564 color corrections (one per image), targeted towards skin with the use of a SkinChecker. All SkinCheckers were illuminated with CIE A reproduced light from a tungsten light bulb. However, no viewing box was used at the time of photography, and image quality and light sources are, thus, inconsistent.

From Table 4.6, the LCC achieves an average $\Delta E^*_{ab} = 8.42$. Considering that color patches on the SkinChecker are clustered in the same color subspace (closely related to human skin, Fig. 2.7), $\Delta E^*_{ab} = 8.43$ is very high. The LCC's non-aggressive color correction yields unsatisfactory results, and more sophisticated color correction solutions are required to perform human skin color correction for scientific use.

5.1.2 PCC and RPCC Frameworks

Literature describes a wide range of color correction methods. However, there is no consensus regarding a preferred unified solution. The relationship, between camera RGBs and device independent CIE L*a*b triplets, is a function of light sources and image capturing devices. However, finding it is not a trivial task. With least squares polynomial and root-polynomial color correction (RPCC), a user needs to identify the best dimensional fit (i.e., RGB-extension) and is only achieved through rigorous experimentation.

To accommodate and simplify the search for this fit, we combine a collection of low-to-high complexity models in a color correction framework. The proposed solutions are attempts to give color correction flexibility without actually doing so. Having created a baseline with the LCC, it is easier to assess the performance of each individual framework. Table 4.2 from Experiment I shows that, compared to each other, both proposed solutions achieve next to equal results, outperforming the baseline with more than $1\Delta E^*_{ab}$. We observe the same trend as when investigating the LCC; i.e., that error varies when different CIE illuminants reproduce light. Again, the CIE A tungsten light bulb creates conditions where both frameworks struggle to achieve good results.

One of the main reasons Finlayson et al. (2015) proposed the RPCC is that exposure changes the vector of polynomial components in a nonlinear way, resulting in hue and saturation shifts. Based on experiment I, the root-polynomial framework performs more stable color correction, in different light settings, compared to the polynomial framework.

In experiment II (training on one color checker, and evaluation on another), the solutions do not know which color checker they are to be evaluated on and naturally optimize for the reference checker (in this case, the SkinChecker). The frameworks, therefore, choose internal methods (RGB-extensions) with dimensional fit best suited for skin color correction; $\hat{\rho}_{2.2}$, $\bar{\rho}_{2.2}$. With no reference color for either green or blue, both frameworks fail to build color correction models that can recreate green/blue related colors. Illustrated by figure 4.3c, the root-polynomial color correction framework (RPCCF) solution creates a purple-brown color as its prediction for turquoise. The polynomial color correction framework (PCCF) solution, on the other hand, does not create any extreme predictions like the RPCCF but shows a notably visible color difference in a large subset of the colors. The results from the experiment indicate that both solutions fail to generalize, and that the less complex LCC significantly outperforms them (LCC: ΔE^*_{ab} = 11.95 << RPCCF: ΔE^*_{ab} = 17.33 < PCCF: ΔE^*_{ab} = 19.72).

Experiment III highlights the importance of more aggressive color correction for human skin. Compared to the LCC baseline, both solutions reduce the color correction error by approximately 50%. The internal method distribution graph from experiment

III (Fig. 4.6) shows how often each internal method is picked across all 564 images. Focusing on the PCCF solution; the figure shows that the framework alternates between three internal methods, more specifically the center-most complexities (i.e., second and third-degree polynomial extensions). The distribution may indicate that human skin color correction is best performed using medium complexity RGB-extensions. The bar plot (Fig. 4.6a) strengthens the hypothesis that no single color correction model is best suited in all scenarios and that combining models of increased complexity lowers the chance of under- and overfit.

When examining the internal method distribution for the RPCCF solution, only the most simple approach (k=2) was applied. One way of looking at this is to assume that the RPCCF favors low complexity models and that $\bar{\rho}_{2,1}$ is the best extension, regardless of the color correction case. However, and this is where we believe Finlayson et al. (2015) come to short, when the root-polynomials increment in degrees, the extensions include *all possible* terms within that degree. The implemented RPCCF is a pure implementation of the root-polynomial extensions presented in the paper, with no experimental adaptations. Thus, the RPCCF only contains three internal methods, where the extensions for $k = 3$ and $k = 4$ are too complex for skin color correction.

From Experiment III the results indicate that both frameworks perform good color correction (PCCF: $\Delta E^*_{ab} = 3.45$, RPCCF: $\Delta E^*_{ab} = 4.74$), and by a large margin, outperform the baseline ($\Delta E^*_{ab} = 8.42$). However, the solutions are not perfect, and the following paragraphs highlight the weaknesses of the framework solutions.

The first major drawback of a framework is that all possible relationship-functions are not tested. To fix this; additional internal methods can be added, but the possibilities would still be limited to the size of an incremental step. A natural incremental step is to take one term and increase its dimensionality once (e.g., $r \rightarrow r^2$), and let that be a new internal method. The idea can be taken even further, by scaling each term in the extension vector with a coefficient (e.g., $r \rightarrow k * r^2$). However, to find the absolute best complexity fit, the step size between each internal method would have to be infinitely small, resulting in an infinite amount of cross-validation runs.

The second drawback is that, as of now, the proposed frameworks are limited to polynomial and root-polynomial extensions. Creating new frameworks of different complexities (e.g., exponential) can, of course, solve this issue. However, both the discussed solutions, are by no means scalable, and their pseudo-flexibility comes from empirical trial and error using leave-one-out cross-validation, a computationally expensive evaluation method. The frameworks are, thus, in regards to time-complexity not feasible.

5.1.3 Gaussian Process Regression

Assuming the best color correction mapping (i.e., the complexity-relationship between camera RGBs and target values) is dependent on the specific color correction scenario (image capturing device, and illumination), a flexible color correction algorithm should, in theory, be beneficial.

To create a flexible and adapting color correction algorithm we look to machine learning. The challenge with machine learning is that the algorithms often require large amounts of data to learn relationships. In color correction, the available data is limited to the number of color patches in the color checker, often resulting in a small datasets. As mentioned in background, we are not pioneers when it comes to this idea, and machine learning and particularly artificial neural networks have been implemented to perform color correction, achieving approximately equal results to an optimized polynomial color correction (PCC) (with optimized PCC we mean a PCC model using the best polynomial fit for that image). As previously discussed, one of the main issues with PCC is that the user needs to find the best polynomial fit through trial and error. However, with neural networks, the user faces a similar problem where the network's hyperparameters (e.g., hidden layers, activation functions, learning rate) need to be empirically determined.

To incorporate adaptiveness and flexibility without the related tuning issues, we propose GPR color correction. While GPR already is a widely used tool for modeling in industry, it is a novel approach to the problem of color correction. Using kernels and the log marginal likelihood, GPR can perform color correction without any explicit hyperparameter tuning. Additionally, the kernels can take on close to any complexity (e.g., polynomial and exponential), making GPR models highly flexible. Being an instance of machine learning, it can find the best mapping complexity for each individual color correction case.

As with any other machine learning technique, a major concern for the GPR is overfitting. This problem is circumvented by adding an aggressive noise kernel to the kernel function, forcing the GPR to generalize. To make the GPR universal for all color checkers; the radial-basis function (RBF) kernel (de-facto default GPR kernel) is used, giving the regression model a high probability for generalization.

Looking at the results in experiment I, the color correction errors show that the GPR, on average, has a $1\Delta E^*_{ab}$ higher error than the best performing RPCCF. The high average error is mostly attributed to its poor color correction capabilities in light produced by the tungsten light bulb (CIE A). However, in CIE D65 reproduced illumination it achieves approximately the same results as both frameworks, indicating that the GPR is capable of general color correction, but that it is sensitive to warm colored light. Compared to the baseline, the GPR is superior in terms of performance and improves on the LCC with $\Delta E^*_{ab} = 0.5$.

When training on the SkinChecker, and evaluated on the SpyderCHECKR 24, the GPR is unmatched by the frameworks and outperforms the general LCC. Both the results and the reconstructed color corrected image (Fig. 4.3d) indicate that the GPR most likely employs less aggressive color correction than the frameworks and creates general color correction models. Experiment II clearly shows that, even when diverse color data is limited (no green or blue colors), the GPR is able to build color correction models that can predict a wide range of colors.

Unlike the frameworks, GPR uses machine learning to adapt its model to each image. When color correcting human skin, the results from experiment III show that the GPR and PCCF perform equally ($\Delta E^*_{ab} = 0.08$ difference is negligible). Further, Table 4.6 shows that the performances of GPR and PCCF, in terms of EM 4 (evaluation method highlighting skin color correction), are equal. Overall, the results indicate that both the GPR and PCCF solutions are viable options when performing human skin color correction.

5.1.4 Summary

The results from experiment I and II merit a small discussion related to light sources. As with the LCC, both frameworks, and the GPR produce a substantially higher error performing color correction in CIE A reproduced light than noon daylight (CIE D65). The warmth and yellow glow from the tungsten light bulb (CIE A) alter the grayscale to the extent that no color correction solution is able to fix them correctly. It is also interesting to see that both frameworks stick to the same internal method regardless of light. With only two examples a confident conclusion cannot be derived from this observation but is nevertheless worth taking note of.

Based on the results we can with certainty state that the tungsten light bulb facilitates poor illuminations for research related experiments. We, therefore, recommend that the continued data gathering for jaundice research is performed in cool white light (high kelvin temperatures) [RQ 1.2](*Which CIE illuminant is most suitable for general color correction?*). When gauging each method's error difference across different illuminations, the RPCCF is, by a small margin, the most stable [RQ 1.1](*Which color correction technique is most robust to varying illumination?*).

Experiment II was designed to answer RQ 1.3, and the results suggest that the GPR, on average, is the color correction technique that best generalizes its prediction capability to a more diverse color subspace. Due to its none aggressiveness, the LCC also achieves respectable results. It is worth noting that answering RQ 1.3 does not necessarily give any valuable information for skin color correction. When working with specialized color correction it is not crucial that a model can generalize well to a diverse range of colors, and the property is regarded as a bonus rather than a requirement. Therefore, the high errors of PCCF and RPCCF in experiment II cannot really be regarded as negative.

When reviewing all experiments, the results show that no single color correction solution can, regardless of scenario, produce the lowest ΔE^*_{ab} error [RQ 1.4](*Is there a single best color correction model regardless of CIE RGB color subspaces CIE reproduced illumination?*). Further, a comparison between the frameworks and GPR does not justly describe the relationship between performance and individual solutions. Frameworks are collections of internal methods, and the individual results show that the correction error produced by the GPR and the optimal internal methods are close to equal. We, therefore, answer RQ 1.4 by stating that the GPR solution is better than stand-alone internal methods, but is not necessarily better than a unified framework solution.

Confirming literature's prior findings on machine learning, the GPR achieves competitive results compared to state-of-the-art solutions, especially when considering human skin color correction. If least squares regression models are tuned to the optimal dimensional complexity fit, they perform equally and sometimes better than our machine learning approach, GPR. In light of RQ 1.5, we cannot, based on the observations, claim that GPR produces more robust color corrections. However, we show that the GPR is equally robust as state-of-the-art color correction solutions while relieving users of tedious parameter tuning.

5.2 Bilirubin Prediction

Previous studies have shown there is a linear correlation between absorbed light at specified wavelengths and total serum bilirubin (TSB). Since light absorption effects visual features, the correlation should, at least to some degree, extend to the perceived color of skin. Perceived skin color is, nevertheless, subjective to the observer, and the human eye is not skilled at comparing colors. Cameras are, however, designed to capture and represent visual features digitally, which allows machines to interpret them. Additionally, digital cameras have in recent time become a commodity and are often embedded in smartphones, where even the cheaper ones come with a reasonably good camera. To create an affordable and readily available platform for jaundice detection, we explore the possibility of bilirubin prediction, through the use of camera sensor RGB values.

The following section presents a discussion on the jaundice dataset, and the proposed bilirubin prediction solutions; support vector regression (SVR), fully-connected neural networks, and convolutional neural networks (CNNs). Lastly, all findings are summarized and related to the research questions.

5.2.1 Dataset

The jaundice dataset truly defines the boundaries and potential of this research. Consisting of only 133 children (where all relevant metadata is intact), the dataset is, from a machine learning perspective, incredibly small. Additionally, the distribution of the target feature (TSB) is not uniform but normally distributed. The dataset, thus, contains few instances of extreme TSB values, values that arguably are the most important. Few extreme values complicate the use of machine learning as it impairs a model's ability to learn features related to severe hyperbilirubinemia thoroughly.

Another important dataset limitation to discuss is the distribution of race and ethnicity. 121 out of 133 children in the provided dataset are Caucasian (91%), and the remaining non-Caucasian data is not enough to create general models for all races. It is therefore natural to redefine the problem and specify that the prediction algorithm is only to be evaluated on the Caucasian race. To accommodate the new problem definition; the 12 non-Caucasian children were removed. Unfortunately, the Caucasians-only restriction inhibits solutions from producing results for the intended target group (sub-Saharan Africa), a group that sorely needs jaundice detection technologies. However, seeing that the proposed solutions are all based on machine learning, new models for other races can quickly be generated if additional data is provided.

5.2.1.1 Metadata and Its Contribution

Preparing the metadata for machine learning algorithms; the neonates' age and weights are normalized. The reasoning behind the selected normalization is covered in section 3.1. Not explicitly covered by the results, the experiments revealed that no matter normalization technique, the weight parameter did not give any increased prediction accuracy. On the contrary, it functioned more like noise and reduced the prediction score by a small margin.

We argue that the newborn's weight would add value if the dataset contained preterm babies. The treatment chart (Fig. 2.3b) indicates that bilirubin concentration, and by extension, the threshold for treatment is dependent on body weight. The chart divides neonates into treatment groups based on their body weight where the intervals between groups are coarse (0.5 - 1 kg). In the jaundice dataset, all children are within the same group (body weight > 2500) (Fig. 3.3a), and due to its noisy behavior, the weight parameter was left out when building the prediction models.

Age, however, did significantly improve the bilirubin prediction, where the logarithmic scaling (between 0-1) was the most appropriate data distribution. The logarithmic scaling is consistent with the treatment chart and draws an approximated logarithmic function similar to the treatment determination line from the chart. The treatment determination function is linear from $t = 0$ h to $t = 72h$ and indicates that bilirubin concentration is heavily dependent on time the first three days after birth, further supported by the graph in figure 2.3a. The treatment determination function is, however, flat for any $t > 72h$, suggesting that age becomes insignificant over time. Since nearly all data resides within the linear range of the treatment determination function, it is not surprising that the age parameter is essential to the bilirubin prediction models.

5.2.2 Evaluation of Dräger Performance

Before building any regression model with machine learning techniques, a test set τ is held out, not to be used until final evaluation. Since the entire jaundice dataset is so limited in size, holding out 30% of all available data, amounts to a small and sensitive test set. The test set consist of 37 children, and we worry all results may be dependent on the seed used to randomly select it. It is, therefore, crucial to verify that the test data contains both low and high TSB values, so that in broad strokes, all cases are represented. Inspecting the Bland-Altman plots from the conducted experiments verifies that the test set contains TSB values from 45 to 370 (treatment starts at $TSB = 350$).

To support the MobileCam-estimated bilirubin concentration (MCB) measurement and its reliability, the Dräger JM-103 is used to create a baseline for the test set. Analyzing the MCB measurements against the JM-103 enables a comparison demonstrating the prediction behavior relative to a state-of-the-art, expensive bilirubinometer commonly used as a screening method by modern hospitals. Earlier studies have shown that the Dräger JM-103 achieves approximately 0.95 correlation on Caucasian neonates.

Evaluating the Dräger JM-103 on the held out test set τ, the results substantiate previous studies as it achieves a 0.95 correlation. The linear regression plot (Fig. 4.7b), shows that for small values (below 200 micromol/L) the Dräger JM-103 generates accurate predictions, but that they become less precise in the higher TSB ranges. Not only are they less precise, but the results reveal a trend where the large values are underestimated. Underestimation of high TSB values is critical as it leaves patients vulnerable to undetected hyperbilirubinemia. It is important to note that an under-estimated value is elevated upwards in the Bland-Altman plot. We stress this detail because it is counter-intuitive ($TSB - prediction > 0 \implies prediction < TSB$).

These observations are further strengthened by the Bland-Altman plot (Fig. 4.8b), showing four underestimated values, outside the $50\mu mol/L$ margin of error[1](know as

[1]The margin of error is a recommended safety margin set by the hospital to make sure all children in the

an outlier)[2]. Underestimating bilirubin concentrations is regarded as a false negative, a result that is not acceptable within the medical community. The hyperbilirubinemia can potentially go undetected if no other analyses are made, leaving the neonates untreated and in the danger zone.

The Bland-Altman plot also shows that the JM-103 bilirubinometer, on average, underestimates the TSB values with approximately $20\mu mol/L$ (bold line). However, to be fair to the JM-103, the high mean difference is most likely attributed to the four large outliers, shifting the mean to higher values. Looking at the confidence interval (the light gray area); the plot shows that the Dräger JM-103 is overall very precise.

5.2.3 Support Vector Regression

Support Vector Machines (SVMs) are widely known for their prediction capabilities on limited datasets. To get a grasp of the problem complexity the SVR is implemented, a low-configuration regression tool.

Evaluated on the test set τ, the SVR achieves 0.91 correlation. Considering the low complexity algorithm behind the prediction; these results are promising, and close to the Dräger JM-103 (0.95 correlation). Of course, a 0.04 correlation difference cannot be regarded as negligible, and admittedly the results are not equally good. The linear regression plot (Fig. 4.7a) shows that, on average, the SVR slightly overestimates low TSB values (values below $150\mu mol/L$), but that its regression line is close to the identity line[3].

At first glance, having a regression line with the same slope as the identity line would seem to indicate great predictions. It is, however, important to realize that the results merely implies that the SVR both over- and under-estimates equally much across the entire range of bilirubin concentrations. The 95% confidence interval (light-gray area)(Fig. 4.8a) in the Bland-Altman plot highlights these trends, as the confidence interval is large (i.e., sparse predictions). Comparing the SVR to the JM-103, the Bland-Altman plot shows that the bilirubinometer has a 20% smaller confidence interval.

The Bland-Altman plot further illustrates that the SVR has two fewer underestimated outliers (predictions outside the margin of error). The underestimated outliers are false negatives and may lead to sick children going untreated. On the other side of the spectrum, the SVR has three overestimated outliers (false positives), where the Dräger JM-103 has none. False positives are, however, much more desirable than false negatives as it only leads to unnecessary follow-ups for wrongly predicted children (i.e., more blood samples to analyze).

danger zone is correctly tested

[2]We define an outlier as a prediction that is $50\mu mol/L$ different from the true TSB value

[3]The identity line is the line defined as $y = x$, sometimes called the 1:1 line

The grid-search, using 5-fold cross-validation, found the RBF kernel to be the best performing kernel. Being the most complex kernel, one can argue that the required function to perform bilirubin regression is of an even higher dimensionality than the RBF kernel can represent.

5.2.4 Fully Connected Neural Networks

Neural networks can approximate close to any function by adding dimensional complexity through deeper networks (i.e., more hidden layers). Seeing that the SVR might not capture all essential features with the RBF kernel, the next natural step is to use fully-connected neural networks.

With a 0.92 correlation, the fully-connected neural network achieves close to competitive results compared to the expensive Dräger JM-103 (0.95 correlation). The neural network only improves by 0.01 correlation to the SVR, which statistically is not much. However, looking at the standard deviation, and mean difference, the neural network is arguably significantly more stable. Added to this, with the SVR being based on a hyperplane, it is uncertain as to how much it will improve if more data becomes available. We are, however, confident that a neural network will be more flexible and adapt better to additional data.

Since the inputs are derived from a single RGB vector (one color), the network has to be shallow. The amount of information between the three color channels is limited, and a large network will most likely overfit. However, seeing that the input dimensionality is scaled by extending the RGB vector, a somewhat broad network is reasonable.

After the grid search, the resulting optimal configuration had two hidden layers; 70 nodes in layer one, and 19 nodes in layer two. The broad architecture allows the network to find the best relationship between the polynomially extended color values. Ideally, a good model would only require the three color values (R, G, B) as input and discover the appropriate RGB-extension (i.e., complexity between the color values). However, the restrictive dataset does not provide enough data, and this workload is relieved from the network by explicitly giving it the polynomially extended RBG vector.

The regression line from the linear regression plot in figure 4.9a shows that the neural network's mean prediction is nearly equal to the identity line. As discussed, this does not necessarily imply accurate predictions but indicates that there are no over- or underestimation trends. The Bland-Altman plot illustrates that the network only has **one** underestimated outlier, just $\sim 10\mu mol/L$ outside the margin of error. Compared to the Dräger JM-103, the neural network is superior in the most critical areas of bilirubin concentration. In fact, the neural network predicts three less false negatives than the Drager JM-103 bilirubinometer, a 75% improvement. While 75% improvement is statistically misleading due to the low prediction count, the neural network arguably outperforms the bilirubinometer concerning false-negative predictions.

The neural network solution does, however, overestimates TSB in several cases (i.e., evaluates a patient to be sick, when they, in fact, are not), which corrupts the performance metrics (e.g., correlation, standard deviation). We argue that the overestimation of bilirubin concentration (false positive) is not harmful, merely an inconvenience. The Bland-Altman plot further shows a decrease in the confidence interval compared to the SVR (9% less). It is still 12% larger than Dräger JM-103's confidence interval, but given more training data, the neural network is likely to achieve better predictions.

5.2.5 Convolutional Neural networks

CNNs are powerful tools to find spatial features and are often used to classify and detect objects in images. While the skin patches extracted after color correction exhibit little to no features visible to the human eye, some properties like color relativity within a neighborhood, or colors running along blood vessels may hold valuable information to the prediction of bilirubin. As of today, domain experts do not use any spatial features to visually assert if a patient has jaundice or not. However, this does not mean that the features do not exist.

The reasoning behind the CNN is to put this idea, to the test. By running small kernels (e.g., kernel size = $3x3$) over the skin patch, the convolution layer looks for any features that may be hidden in the skin patch. One of the main concerns when using CNNs without transfer learning is that they require large amounts of data to properly train. The jaundice dataset, as discussed, does not contain many images, and the convolutional layers are struggling to train correctly. A dataset consisting of $100x100$ images, where all pixels are approximately equal, is not enough information to train the layers in a CNN.

If the convolution network is able to learn that it can use the kernels to simply extract the mean RGB values, it should be able to achieve equal results as the feed-forward fully-connected network. Unfortunately, we have not been successful in configuring the CNN to do so. The results from both the linear regression and Bland-Altman plot indicate that the network is guessing the two most frequently occurring TSB values.

5.2.6 Summary

Having analyzed all solutions; the fully-connected network shows promising results. Although reaching a lower correlation than the costly, state-of-the-art Dräger JM-103, the solution predicts fewer values below the margin of error, thereby producing 75% fewer false negative predictions. This number is, of course, misleading as the number of predictions are so small, but it does give a general idea of Dräger JM-103's instability and the network's performance in the high TSB ranges.

Looking at the plots, and the correlation of 0.92, the neural network is close to the Dräger JM-103 and achieves competitive results [RQ 2.1](*Can computer vision and deep learning be used to achieve competitive results to state-of-the-art transcutaneous bilirubin (TcB) measurement approaches (bilirubinometers)?*). However, as with TcB measurements, the results also suggest that none of the solutions put forth are robust or accurate enough to replace blood samples (TSB). The machine learning solutions can, therefore, not serve as a standalone approach to assess neonatal jaundice [RQ 2.3](*Can image-based bilirubin prediction serve as a standalone diagnostic tool for severe hyperbilirubinemia?*). Nevertheless, being so close to an apparatus that, as of today, is widely distributed in modern hospitals, the neural network can function as an affordable screening method and be a valuable alternative to the transcutaneous bilirubinometers.

Seeing that the neural network achieves competitive results, even on images with low quality, it is safe to say that the proposed solution is flexible enough to be used in outpatient and home environments [RQ 2.2](*Given color correction, are image-based bilirubin prediction models flexible enough to be used in outpatient and home environments?*). Bilirubinometers are known to perform poorly in the higher ranges of bilirubin concentration, and it is worrying to see it predict four false negatives on a test set τ consisting of only 37 children (11%). Being more stable in the critical regions; the neural network may, in fact, be a better option for detecting severe hyperbilirubinemia.

At this point, no evidence suggests there exist spatial adjacent color features in human skin that provides any additional information about the underlying bilirubin concentration [RQ 2.4](*Does spatially adjacent color features in human skin provide additional information about the underlying bilirubin concentration?*). However, failing to train the CNN properly, further testing is required to fully answer this question.

5.3 Project Reflection

In the initial phase of the project, we were not sure as to which machine learning approach would be best suited for jaundice detection. A common approach in deep learning is to always build bigger and deeper networks for increased accuracy. However, with such limited data, we were convinced that simply throwing a state-of-the-art CNN at the problem would not work. This hypothesis was further strengthened by the failed training of the small convolutional network used to look for spatially neighboring features.

With 484 images, we had to think outside the box and relieve the learning-based models of a lot of the learning. An end-to-end approach was quickly disregarded, as there was not enough data for the model to learn color correction, or even recognize that the neonate's skin is the region of interest. We feel like this project has given us valuable insight when it comes to machine learning and the tools applied to different problems. Deeper is not always better, and we are happy to see that the small feed-forward fully-connected network is performing so well.

Looking back at the entire project; color correction became a considerably more important part the thesis than initially intended. At first, we wanted to find an off-the-shelf color correction algorithm to correct the color errors in the skin images. However, seeing that color is the core information for jaundice detection, we had to be certain that the applied algorithm is robust to varying illuminations and that it can perform accurate skin color correction. It was very reassuring to conduct the color correction experiments ourselves and compare the different solutions.

Chapter 6

Conclusion and Future Work

The yellow skin discoloration associated with jaundice is attributed to high concentrations of bilirubin, a waste product formed during the break down of oxygen-carrying components in blood. It is the light absorption properties of bilirubin, absorbing green and blue light, that causes the skin to be perceived as yellow. Studies show there is a linear correlation between skin absorbed light and total serum bilirubin (TSB), the bilirubin concentration measured in blood samples. Skin reflectance measurements can, therefore, be used to estimate bilirubin concentration, often referred to as transcutaneous bilirubin (TcB). TcB measurements are, today, regarded as reliable substitutes for TSB and are used as non-invasive screening methods by the medical community. The required apparatus, called bilirubinometer, is, unfortunately, expensive (50 000 NOK) and limits widespread availability.

Using smartphone cameras to capture skin induced color values, we create a low-cost alternative to bilirubinometers through computer vision and deep learning. The drastic impact light sources and camera sensors have on images, accentuate the need for color correction. To create standardized skin images for the prediction of bilirubin, we present Gaussian process regression (GPR) as a novel approach for color correction. The results show that the proposed GPR achieves competitive results compared to state-of-the-art color correction techniques.

The bilirubin predictions from camera captured skin color, MobileCam-estimated bilirubin concentration (MCB) measurements, show promising results. Building a neural network-based regression model, we get 0.92 correlation on the held-out test set, achieving competitive results to the expensive state-of-the-art Dräger JM-103. While lower correlation, the neural network predictions are more accurate in the critical TSB ranges, resulting in 75% fewer false negatives than the JM-103. Since our solution requires no equipment other than a smartphone and an inexpensive SkinChecker, it has large potential for monitoring neonates in outpatient and private settings. A camera-based approach allows for widespread use, even in low- and middle-income countries, as smartphones are starting to become a commodity. We conclude that our solution can be implemented as a mobile application and is an affordable screening alternative to expensive bilirubinometers.

6.1 Future Work

This thesis provides a foundation for a mobile camera bilirubin prediction framework, but there is still room for improvement. We divide the discussion of future work into the following subsections; color correction, bilirubin prediction, and fully automatic mobile app.

6.1.1 Color Correction

When preparing images for machine learning, the actual resulting color does not matter. Unless transfer learning is applied, machine learning models, unlike humans, do not compare colors to their prior belief of how colors should look. Machine learning algorithms are only concerned about the relative relationship between the colors presented during training. Therefore, the most crucial part of color correction is to convert all images to a single standard. It does not matter if there is an error, as long as the error is equal for all cases. Future color correction will focus on error balance, and accentuate the white-balance in images. Hopefully, emphasis on white-balance will result in more standardized images and provide robustness to the overall jaundice detection pipeline.

As mentioned in the root-polynomial color correction (RPCC) discussion section, the dimensional increments to the RGB-extension vectors are too large. To create a better root-polynomial framework we will experiment with added internal methods of lower complexity and see if they improve the root-polynomial color correction framework (RPCCF).

6.1.2 Bilirubin Prediction

For machine learning techniques to create robust, and generalized regression models, large amounts of data are required. As discussed, the jaundice dataset is limited in size, and the models are most likely far from optimal.

Nevertheless, seeing that the proposed solutions are based on machine learning, the models only require post-training on additional data to become better. For future work, the task of collecting more jaundice related data becomes imperative.

Extensive jaundice data collection is already on-going at Akershus University Hospital. As we discovered, cool-white illumination settings (high kelvin) result in lower color correction errors. Therefore, all continued image capturing related to the gathering of jaundice data will be performed in white light, and the tungsten light bulb (CIE A) will be avoided.

Unfortunately, the provided jaundice dataset only contains 12 non-Caucasian neonates, and we cannot infer the MCB correlation for non-Caucasian patients. More data is, therefore, required to build racially independent prediction models. Luckily, additional funding has already been granted, and jaundice data collection for non-Caucasians will commence in Tanzania, Mexico, and Nepal. If a unified prediction model can be created, or if ethnicity must be provided as input is yet to be seen.

6.1.3 Convolution Neural Networks

As discussed, we were not able to properly train the convolutional neural network (CNN). For future work, we would first like to find a better CNN architecture and improve the results. Secondly, the skin patches used from the raw images do not contain a lot of information, and we want to look into the possibility of using the entire neonate torso to train the CNN.

6.1.4 Fully Automatic Mobile App

The research presented in the thesis is mostly theoretical, and all experiments are conducted on computers. For the bilirubin prediction solution to be readily available, all correction algorithms and regression models must be implemented into a smartphone application. The application will need to be easy to use, and limit the possibility of user error. Automatic detection and image capturing of the SkinChecker when the light is adequate will become an essential feature. A mobile application allows for widespread availability, where even the average person has the technology to perform robust jaundice screenings.

Bibliography

Abramowitz, M., Stegun, I. A., 1964. Handbook of mathematical functions: with formulas, graphs, and mathematical tables. Vol. 55. Courier Corporation.

Anderson, M., Motta, R., Chandrasekar, S., Stokes, M., 1996. Proposal for a standard default color space for the internet—srgb. Color and Imaging Conference 1996 (1), 238–245.

Anderson, R. R., Parrish, J. A., 1981. The optics of human skin. Journal of investigative dermatology 77 (1), 13–19.

Aster, R. C., Borchers, B., Thurber, C. H., 2011. Parameter estimation and inverse problems. Vol. 90. Academic Press.

Avery GB, Fletcher MA, M. M., 1994. In: Avery's neonatology: Pathophysiology and management of the newborn, 4th Edition. Lippincott Company, pp. 630–725.

Bart, C., Eindhoven, H., 2011. Speed sign detection and recognition by convolutional neural networks. Proceedings of the 8th International Automotive Congress.

Brito, C. C., 2016. Spectral printing for monitoring jaundice in newborns, internal article.

Chang, C.-C., Lin, C.-J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cheung, V., Westland, S., Connah, D., Ripamonti, C., 1 2004. A comparative study of the characterisation of colour cameras by means of neural networks and polynomial transforms. Coloration Technology 120 (1), 19–25.
URL http:https://doi.org/10.1111/j.1478-4408.2004.tb00201.x

Chollet, F., 2017. Deep Learning with Python, 1st Edition. Manning Publications Co., Greenwich, CT, USA.

Cortes, C., Vapnik, V., Sep 1995. Support-vector networks. Machine Learning 20 (3), 273–297.
URL https://doi.org/10.1007/BF00994018

Downing, K. L., 2017. Regularization and Optimization of Backpropagation. http://www.idi.ntnu.no/emner/it3105/tdt76/lectures/deep-lecture-3.pdf, [Online; accessed 03-Dec-2017].

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., Vapnik, V., 1996. Support vector regression machines. In: Proceedings of the 9th International Conference on Neural Information Processing Systems. NIPS'96. MIT Press, Cambridge, MA, USA, pp. 155–161.
URL http://dl.acm.org/citation.cfm?id=2998981.2999003

El-Beshbishi, S. N., Shattuck, K. E., Mohammad, A. A., Petersen, J. R., 2009. Hyperbilirubinemia and transcutaneous bilirubinometry. Clinical Chemistry 55 (7), 1280–1287.
URL http://clinchem.aaccjnls.org/content/55/7/1280

Finlayson, G. D., Mackiewicz, M., Hurlbert, A., May 2015. Color correction using root-polynomial regression. IEEE Transactions on Image Processing 24 (5), 1460–1470.

Ford, A., Roberts, A., 1998. Colour space conversions. Westminster University, London 1998, 1–31.

Francesco Raimondi, Lama Silvia, F. L. M. S. A. C. B. R. M. P. M., Capasso, L., 2012. Measuring transcutaneous bilirubin: a comparative analysis of three devices on a multiracial population. BMC Pediatrics.
URL https://doi.org/10.1186/1471-2431-12-70

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, http://www.deeplearningbook.org.

Grassmann, H., 1853. Zur theorie der farbenmischung. Annalen der Physik 165 (5), 69–84.

Guowei, H., Ronnier, L. M., A., R. P., 2001. A study of digital camera colorimetric characterization based on polynomial modeling. Color Research & Application 26 (1), 76–84.

HunterLab, 2005. Equivalent white light sources and cie illuminants, applications note.
URL https://web.archive.org/web/20050523033826/http://www.hunterlab.com:80/appnotes/an05_05.pdf

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167.
URL http://arxiv.org/abs/1502.03167

K Bhutani, V., Zipursky, A., Blencowe, H., Khanna, R., Sgro, M., Ebbesen, F., Bell, J., Mori, R., Slusher, T., Fahmy, N., K Paul, V., Du, L., Okolo, A., de Almeida, M. F., Olusanya, B., Kumar, P., Cousens, S., E Lawn, J., 12 2013. Neonatal hyperbilirubinemia and rhesus disease of the newborn: incidence and impairment estimates for 2010 at regional and global levels. Pediatric research 74, 86.

Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. CoRR abs/1412.6980.
URL http://arxiv.org/abs/1412.6980

Kollias, N., Baqer, A. H., 1987. Absorption mechanisms of human melanin in the visible, 400–720 nm. Journal of investigative dermatology 89 (4), 384–388.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS'12. Curran Associates Inc., USA, pp. 1097–1105.
URL http://dl.acm.org/citation.cfm?id=2999134.2999257

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., 1989a. Backpropagation applied to handwritten zip code recognition. Neural Computation 1 (4), 541–551.
URL https://doi.org/10.1162/neco.1989.1.4.541

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., 1989b. Backpropagation applied to handwritten zip code recognition. Neural Computation 1 (4), 541–551.
URL https://doi.org/10.1162/neco.1989.1.4.541

MacKay, D. J., 1998. Introduction to gaussian processes. NATO ASI Series F Computer and Systems Sciences 168, 133–166.

Maisels, M. J., 2015. Transcutaneous bilirubin measurement: Does it work in the real world? Pediatrics 135 (2), 364–366.
URL http://pediatrics.aappublications.org/content/135/2/364

Maisels, M. J., Kring, E., 2006. Transcutaneous bilirubin levels in the first 96 hours in a normal newborn population of >=35 weeks' gestation. Pediatrics 117 (4), 1169–1173.
URL http://pediatrics.aappublications.org/content/117/4/1169

Maisels, M. J., Ostrea, E. M., Touch, S., Clune, S. E., Cepeda, E., Kring, E., Gracey, K., Jackson, C., Talbot, D., Huang, R., 2004. Evaluation of a new transcutaneous bilirubinometer. Pediatrics 113 (6), 1628–1635.
URL http://pediatrics.aappublications.org/content/113/6/1628

Masuda, Y., Yamashita, T., Hirao, T., Takahashi, M., 2009. An innovative method to measure skin pigmentation. Skin research and technology 15 (2), 224–229.

McCamy, C. S., Marcus, H., Davidson, J., et al., 1976. A color-rendition chart. J. App. Photog. Eng 2 (3), 95–99.

McCulloch, W. S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics 5 (4), 115–133.

McDonagh, A. F., Lightner, D. A., 1985. 'like a shrivelled blood orange'—bilirubin, jaundice, and phototherapy. Pediatrics 75 (3), 443–455.
URL http://pediatrics.aappublications.org/content/75/3/443

N, S. R., Deka, P. C., 2014. Support vector machine applications in the field of hydrology: A review. Applied Soft Computing 19, 372 – 386.
URL http://www.sciencedirect.com/science/article/pii/S1568494614000611

Nameer Hirschkind, J. P., Khim, J., 2017. Convolutional Neural Network. https://brilliant.org/wiki/convolutional-neural-network/, [Online; accessed 03-Dec-2017].

Nishad, P., 2013. Various colour spaces and colour space conversion. International Journal of Global Research in Computer Science (UGC Approved Journal) 4 (1), 44–48.

Park, J., Park, K., 1995. Professional colour communicator-the definitive colour selector. Coloration Technology 111 (3), 56–57.

Pascale, D., 2003. A review of rgb color spaces... from xyy to r'g'b'. Babel Color 18, 136–152.

Randeberg, L. L., 2005. Diagnostic applications of diffuse reflectance spectroscopy. Ph.D. thesis, Norwegian University of Science and Technology.

Rasmussen, C. E., Williams, C. K., 2006. Gaussian process for machine learning. MIT press.

Richards, F. J., 1959. A flexible growth function for empirical use. Journal of Experimental Botany 10 (29), 290–300.
URL http://www.jstor.org/stable/23686557

Rodgers, J. L., Nicewander, W. A., 1988. Thirteen ways to look at the correlation coefficient. The American Statistician 42 (1), 59–66.
URL https://doi.org/10.1080/00031305.1988.10475524

Rubaltelli, F. F., Gourley, G. R., Loskamp, N., Modi, N., Roth-Kleiner, M., Sender, A., Vert, P., 2001. Transcutaneous bilirubin measurement: A multicenter evaluation of a new device. Pediatrics 107 (6), 1264–1271.
URL http://pediatrics.aappublications.org/content/107/6/1264

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115 (3), 211–252.

Russell, S. J., Norvig, P., 2016. Artificial intelligence: a modern approach. Pearson Education Limited,.

Sharma, G., 2002. Digital Color Imaging Handbook. CRC Press, Inc., Boca Raton, FL, USA.

Slusher, T. M., Zipursky, A., Bhutani, V. K., 2011. A global need for affordable neonatal jaundice technologies. Seminars in Perinatology 35 (3), 185 – 191. URL http://www.sciencedirect.com/science/article/pii/S0146000511000437

Taylor, J. A., Stout, J. W., de Greef, L., Goel, M., Patel, S., Chung, E. K., Koduri, A., McMahon, S., Dickerson, J., Simpson, E. A., Larson, E. C., 2017. Use of a smartphone app to assess neonatal jaundice. Pediatrics 140 (3). URL http://pediatrics.aappublications.org/content/140/3/e20170312

Tenhunen, R., Marver, H. S., Schmid, R., 1969. Microsomal heme oxygenase characterization of the enzyme. Journal of Biological Chemistry 244 (23), 6388–6394.

Wandell, B. A., 1995. Foundations of vision. Sinauer Associates.

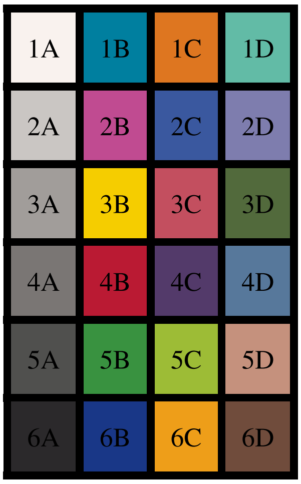
Williams, C. K. I., 1997. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In: Learning and Inference in Graphical Models. Kluwer, pp. 599–621.

Wyszecki, G., Stiles, W. S., 1982. Color science. Vol. 8. Wiley New York.

Zijlstra, W. G., Buursma, A., van Assendelft, O. W., 2000. Visible and near infrared absorption spectra of human and animal haemoglobin: determination and application. VSP.

Appendix A

Color Checkers



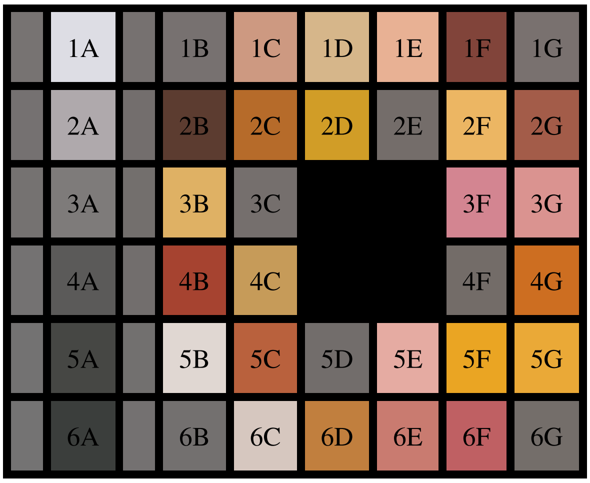
1A	1B	1C	1D
2A	2B	2C	2D
3A	3B	3C	3D
4A	4B	4C	4D
5A	5B	5C	5D
6A	6B	6C	6D

Table A.1: The SpyderCHECKR 24 with 24 unique colors characterized by different spectral responses

		L*a*b*			sRGB		
Patch	Name	L*	a*	b*	R	G	B
1A	Card White	96.04	2.16	2.60	249	242	238
2A	20% Gray	80.44	1.17	2.05	202	198	195
3A	40% Gray	65.52	0.69	1.86	161	157	154
4A	60% Gray	49.62	0.58	1.56	122	118	116
5A	80% Gray	33.55	0.35	1.40	80	80	78
6A	Card Black	16.91	1.43	-0.81	43	41	43
1B	Primary Cyan	47.12	-32.52	-28.75	0	127	159
2B	Primary Magenta	50.49	53.45	-13.55	192	75	145
3B	Primary Yellow	83.61	3.36	87.02	245	205	0
4B	Primary Red	41.05	60.75	31.17	186	26	51
5B	Primary Green	54.14	-40.76	34.75	57	146	64
6B	Primary Blue	24.75	13.78	-49.48	25	55	135
1C	Primary Orange	60.94	38.21	61.31	222	118	32
2C	Blueprint	37.80	7.30	-43.04	58	88	159
3C	Pink	49.81	48.50	15.76	195	79	95
4C	Violet	28.88	19.36	-24.48	83	58	106
5C	Apple Green	72.45	-23.57	60.47	157	188	54
6C	Sunflower	71.65	23.74	72.28	238	158	25
1C	Aqua	70.19	-31.85	1.98	98	187	166
2D	Lavender	54.38	8.84	-25.71	126	125	174
3D	Evergreen	42.03	-15.78	22.93	82	106	60
4D	Steel Blue	48.82	-5.11	-23.08	87	120	155
5D	Classic Light Skin	65.10	18.14	18.68	197	145	125
6D	Classic Dark Skin	36.13	14.15	15.78	112	76	60

Table A.2: Values from the SpyderCHECKR 24[1]

[1] `https://www.datacolor.com/wp-content/uploads/2018/01/SpyderCheckr_Color_Data_V2.pdf`



	1A		1B	1C	1D	1E	1F	1G
	2A		2B	2C	2D	2E	2F	2G
	3A		3B	3C			3F	3G
	4A		4B	4C			4F	4G
	5A		5B	5C	5D	5E	5F	5G
	6A		6B	6C	6D	6E	6F	6G

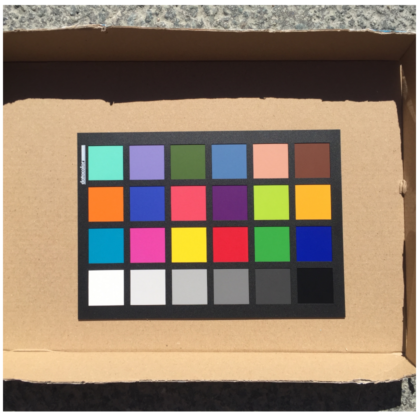
Table A.3: The SkinChecker by Picterus AS

	L*a*b*			sRGB		
Patch	L*	a*	b*	R	G	B
1A	88.38	1.51	-3.32	221	221	228
2A	70.14	2.72	-0.74	175	169	172
3A	51.97	1.43	0.66	126	123	122
4A	38.53	-0.15	0.93	91	90	89
5A	30.25	-1.12	2.06	70	71	68
6A	26.02	-1.50	0.71	59	62	60
1B	48.32	1.82	1.49	119	113	112
2B	28.82	12.48	13.19	92	60	48
3B	74.93	8.10	45.14	223	177	100
4B	42.03	39.70	31.89	166	67	48
5B	86.81	2.21	3.52	224	215	210
6B	47.62	0.74	1.01	115	112	111
1C	67.77	15.92	20.13	205	153	129
2C	52.60	24.83	47.00	182	107	42
3C	47.52	2.17	1.81	117	111	109
4C	66.73	8.33	40.28	198	155	89
5C	51.02	32.66	35.69	185	97	61
6C	81.35	4.01	5.79	214	199	191
1D	75.98	5.20	26.58	214	182	138
2D	68.04	8.81	63.77	209	157	39
5D	46.63	1.55	1.91	114	109	107
6D	58.96	19.64	44.55	193	127	62
1E	76.70	16.15	22.70	232	177	148
2E	46.84	2.26	2.76	116	109	106
5E	75.25	20.00	13.09	229	171	162
6E	59.69	29.13	18.90	201	123	112
1F	36.42	24.92	17.63	129	68	58
2F	77.67	10.08	49.47	236	182	99
3F	64.09	31.40	6.45	211	133	145
4F	46.33	2.08	3.07	115	108	104
5F	72.85	15.48	70.20	234	165	35
6F	52.44	38.26	16.14	191	96	99
1G	48.53	2.43	2.41	121	113	111
2G	46.97	26.89	23.61	163	92	73
3G	68.04	26.16	12.62	218	147	144
4G	56.56	32.21	55.49	205	110	33
5G	73.81	14.00	64.45	234	169	55
6G	47.21	1.44	3.02	116	110	106

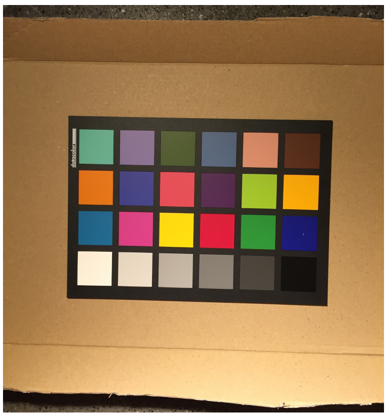
Table A.4: Values from SkinChecker by Picterus AS

Appendix B

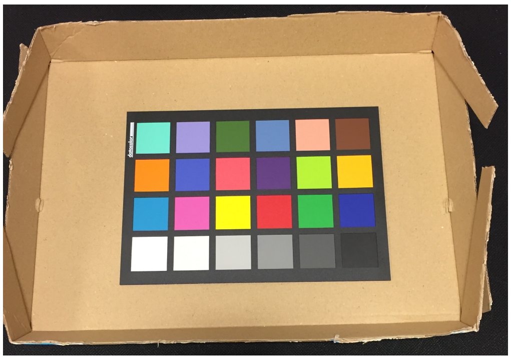
Viewing Box Dataset Images



(a) Noon Daylight
(CIE D65 reproduced illumination)



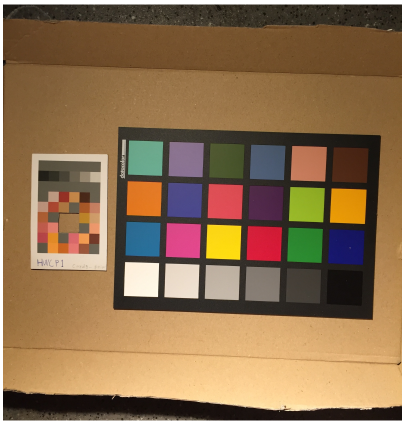
(b) Tungsten light
(CIE A reproduced illumination)



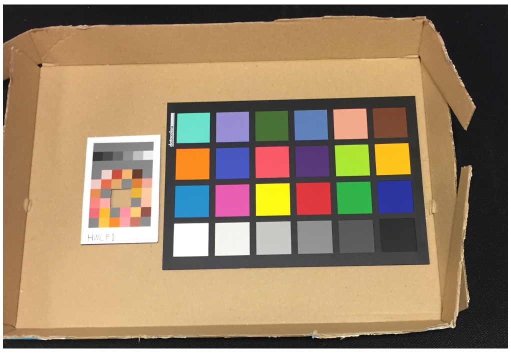
(c) Fluorescent light
(CIE F11 reproduced illumination)

Figure B.1: Color correction experiment I: SpyderCHECKR 24 dataset.

(a) Noon Daylight
(CIE D65 reproduced illumination)



(b) Tungsten light
(CIE A reproduced illumination)



(c) Fluorescent light
(CIE F11 reproduced illumination)

Figure B.2: Color correction experiment II: SpyderCHECKR 24 and SkinChecker dataset.