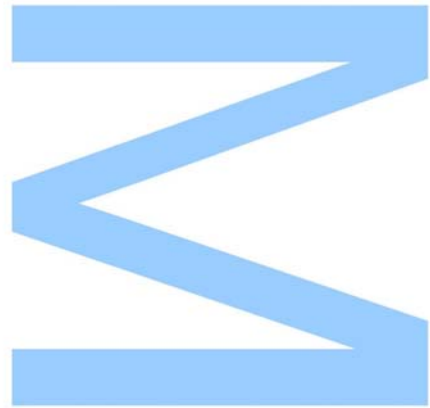




Computer Vision algorithm for Determination of sulfonamides in water



Inês Baptista da Rocha

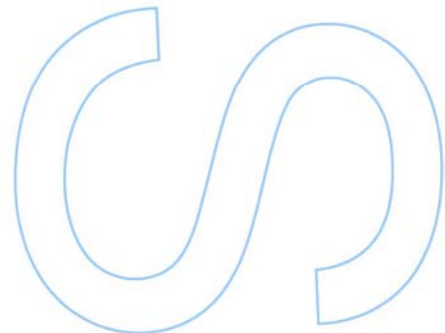
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Faculdade de Ciências
2020

Orientador

Hélder Filipe Pinto de Oliveira, FCUP, INESC TEC

Supervisor

Pedro Henrique Moreira Queirós Carvalho, INESC TEC

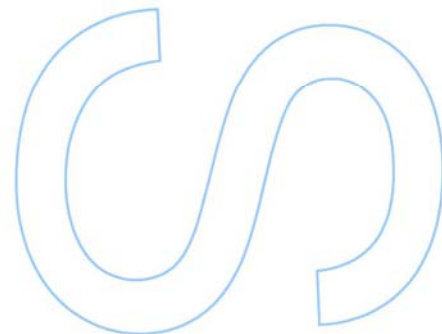
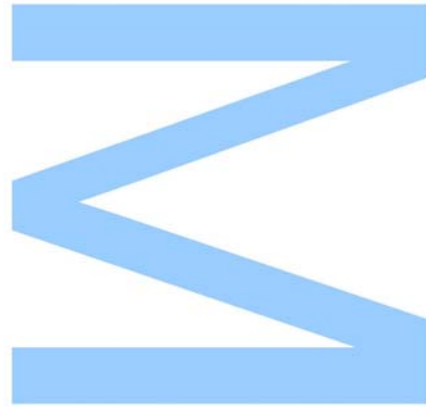




Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ____/____/____



Abstract

The consumption of antibiotics by humans and animals has increased in recent decades, and with it their presence in aquatic environments. This presence has caused bacteria to become resistant to these antibiotics, making treatment more difficult when these resistant bacteria come into contact with animals or humans. One of these antibiotics is sulfonamides.

These antibiotics are usually detected by taking a water sample to a laboratory and quantifying it using expensive methods. Recently, digital colorimetry has emerged as a new method for detecting sulfonamides in water. When a reagent comes into contact with a water sample, a color is produced from which we can infer the concentration of sulfonamides. To ensure that the color is not affected by the illumination when taking a photograph, a color reference target is positioned next to the sample to correct the colors in the image and obtain a true color of the sample.

This method has already been implemented in smartphones to provide a faster and more practical tool that can be used immediately when collecting water samples. Despite this improvement, the algorithms used can still be outperformed by the use of machine learning, as they will provide more accurately results.

The best feature of machine learning models is their ability to learn from data. This allows them to perform tasks such as object recognition as well as or better than humans, and to detect the concentration of sulfonamides based on the color of a sample.

We developed 4 machine learning models that can be converted to be use in a smartphone app. A modified SSD MobileNet V2 model to detect the color patches of the color reference target, a modified SSD MobileNet FPNLite to detect the color of the sample, a machine learning model to color correct the colors of an image, and a random forest tree regression model to calculate the sulfonamides concentration based on the corrected color of the sample. These models improve the accuracy of the older algorithms and allow for more precise analysis of this hazardous substance in water samples.

Keywords: Computer Vision, Image Processing, Object Detection, Color Correction, Machine Learning, Deep Learning.

Resumo

O consumo de antibióticos por humanos e animais tem aumentado nas últimas décadas, e com isso a sua presença em ambientes aquáticos. Essa presença tem vindo a causar bactérias a tornassem-se resistentes a esses antibióticos, dificultando o tratamento quando essas bactérias resistentes entram em contato com animais ou humanos. Um desses antibióticos são as sulfonamidas. Estes antibióticos são geralmente detetados levando uma amostra de água a um laboratório e quantificando-a usando métodos caros. Recentemente, a colorimetria digital surgiu como um novo método para detetar sulfonamidas na água. Quando um reagente entra em contato com uma amostra de água, é produzida uma cor da qual podemos inferir a concentração de sulfonamidas. Para garantir que a cor não seja afetada pela iluminação ao tirar uma fotografia, um alvo de referência de cor é posicionado próximo da amostra para corrigir as cores na imagem e obter uma cor verdadeira da amostra. Este método já foi implementado em smartphones para fornecer uma ferramenta mais rápida e prática que pode ser usada imediatamente na recolha de amostras de água. Apesar dessa melhoria, os algoritmos utilizados ainda podem ser superados pelo uso de machine learning, pois estes fornecem resultados com maior precisão. A melhor característica de machine learning é a capacidade de aprender com os dados. Isto permite que realizem tarefas como reconhecimento de objetos tão bem ou melhores que os humanos e também conseguem detetar a concentração de sulfonamidas com base na cor de uma amostra. Desenvolvemos 4 modelos de machine learning que podem ser convertidos para uso num aplicativo de smartphone. Um modelo SSD MobileNet V2 modificado para detetar as cores do alvo de referência de cor, um SSD MobileNet FPNLite modificado para detetar a cor da amostra, um modelo de machine learning para corrigir as cores da imagem e um modelo de random forest tree para calcular a concentração de sulfonamidas com base na cor corrigida da amostra. Estes modelos melhoram a precisão dos algoritmos mais antigos e permitem uma análise mais precisa desta substância perigosa em amostras de água.

Palavras Chave: Visão computacional, Processamento de imagens, Detecção de objectos, Correção de cor, Machine Learning, Deep Learning.

Acknowledgements

This work was financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e Tecnologia within project POCI-01-0145-FEDER-031756.

I would like to thank Hélder Oliveira and Henrique Carvalho for the assistance they provided with this dissertation.

I would also like to thank the most important people in my life: my father, my mother, my brother and my boyfriend for all the support, encouragement and patience they have given me. And especially for all the housework they took on to help me focus and do my best to produce a good work. Without them, this dissertation would not have been finished.

And last but not least, I would like to thank my cats for all the wonderful naps they took beside me to keep me company and for all the cuddles I received during the many hours I have spent working on this dissertation.

Thank you all for this amazing experience.

Inês Rocha

Contents

Abstract	i
Resumo	iii
Acknowledgements	v
Contents	ix
List of Tables	xi
List of Figures	xv
Acronyms	xvii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Contribution	3
1.4 Organization	3
2 Literature Revision	5
2.1 Detecting and quantifying sulfonamides	5
2.2 Object Detection	8
2.3 Color correction	9

2.3.1	Traditional methods	10
2.3.2	Learning Methods	12
2.4	Summary	14
3	Baseline Method for Color Correction on Uncontrolled Lighting Conditions	17
3.1	Methodology	18
3.1.1	Data	19
3.1.2	Color Correction	20
3.2	Results	22
3.3	Conclusions	24
4	Machine Learning Models	25
4.1	TensorFlow and TensorFlow Lite	27
4.2	Patches Detection	28
4.2.1	Building training dataset	28
4.2.2	Model	29
4.2.3	Results	30
4.3	Disk and Color Detection	32
4.3.1	Building training dataset	32
4.3.2	Model	33
4.3.3	Results	36
4.4	Color correction	36
4.4.1	Building training dataset	37
4.4.2	Model	38
4.4.3	Results	40
4.5	Calculation sulfonamides concentration	41
4.5.1	Building training dataset	41

4.5.2 Model	42
4.5.3 Results	44
4.6 Summary	45
5 Conclusions and Future Work	47
5.1 Conclusions	47
5.2 Future Work	47
Bibliography	49

List of Tables

3.1	Standard deviations of the a^* color value for each of the target patches	23
3.2	Standard deviations of the H color value for each of the target patches	23
3.3	Standard deviations of the a^* color value for the red, yellow, violet and magenta target patches	24
3.4	Standard deviations of the H color value for the red, yellow, violet and magenta target patches	24
4.1	Result of patches detection - Traditional Segmentation Techniques (TST) vs Machine Learning Model (MLM)	32
4.2	Result detection disk and color - TST vs MLM	36
4.3	Representation of the layers for each model. All layers have a kernel size of 3x3	39
4.4	Representation of the activation and error function use in our models	40
4.5	Results of the color correction models	42
4.6	Result of color correction - Traditional Correction Model (TCM) vs MLM	42
4.7	Result of the regression models with different type of color components	44
4.8	Result of Decision Tree and Forest Tree with the most important color components	45

List of Figures

1.1	Example of image in the dataset	2
1.2	Detections of the machine learning models	3
2.1	Image of tthe color checker and the sample	6
2.2	Segmentation process of the color checker	7
2.3	Segmentation process of the patches	7
2.4	Bounding boxes of the patches where the color will be extracted	7
2.5	Segmentation process of the patches	8
2.6	Original image(left) and color corrected image (right)	8
2.7	Architecture of the MobileNetV2	9
2.8	Eye vs Camera	9
2.9	Example of automatic white balance options	10
2.10	Diagram of the procedure	12
2.11	Example of a neuro network	12
2.12	Architecture of the model	13
2.13	Camera and dataset	13
2.14	Architecture of the model	14
2.15	Architecture of the model	14
3.1	Example photograph from dataset	18

3.2	Examples of samples of varying sulfonamide concentrations, from 0 $\mu\text{g/L}$ (left) to 150 $\mu\text{g/L}$ (right).	18
3.3	Example photographs from the new dataset. Taken inside with a white light (left), inside with a yellow light (middle) and outside in the sun (right)	19
3.4	Example of the 8 targets of the color correction (top) and the 24 patches used as reference for the color correction (bottom).	20
3.5	Selected patches for each of the 8 targets. Each target indicated on the top left, and the black dots indicating the 13 chosen patches from the 24 color chart	21
4.1	Flowchart Methodology	26
4.2	Example of different lighting condition	26
4.3	TensorFlow Object Detection API	27
4.4	TensorFlow Lite Converter	28
4.5	Detection of the patches	28
4.6	Example of image annotation	29
4.7	Result of the first experience to detect the color checker	30
4.8	Examples of detection results	31
4.9	Disk and color close up	32
4.10	Example of train disk image annotation	32
4.11	Examples of samples of varying sulfonamides concentrations, from 0 $\mu\text{g/L}$ (left) to 150 $\mu\text{g/L}$ (right)	33
4.12	Input and output of the first experiment of the detection model	33
4.13	Input and output of the second experiment of the detection model	34
4.14	Input and output of the third experiment of the detection model	35
4.15	Example of disk and color detection results	36
4.16	Colors taken from a photo (left) vs ground truth colors (right)	37
4.17	Example image of where the color is extracted from the color checker.	37
4.18	Average of a color point	37

4.19 Training images created by a single photo of the color checker	38
4.20 Colors taken from the original photo (top), the true value the colors should had (bottom)	38
4.21 Example images of the color correction models that had troubles learning	40
4.22 Example of color corrected images. From left to right model 2.1, model 1.4, model 4.4, model 3.2 and model 4.5	41
4.23 Example of data array created	43
4.24 Correlation of the color components and concentration value	43
4.25 Graph of the importance of each color component	45

Acronyms

MAE Mean Absolute Error

MLM Machine Learning Model

MSE Mean Square Error

RBF Radial Basis Function

SVM Support Vector Machines

TCM Traditional Correction Model

TST Traditional Segmentation Techniques

Chapter 1

Introduction

Thanks to scientific advances in the mid-1900s, there was an improvement in livestock production that increased the consumption of animal protein per person by 50% from 1961 to 1998 [1]. This scientific knowledge brought new developments in breeding, nutrition, and animal health. An important development in animal health was the increased use of antibiotics to prevent the spread of disease. One of these antibiotics was sulfonamides. These antibiotics were and still are widely used today [2].

Sulfonamides are antibiotics used to treat bacterial infections in humans and animals. When ingested, these antibiotics concentrate in the urine before being excreted. Once this urine enters groundwater, they can remain active for a long period of time [3]. When left in water streams, bacterial populations are exposed to repeated sublethal doses that cause them to become antibiotic resistant. As these new bacteria spread to humans, their infections will be more difficult to treat, cause higher medical costs, longer hospital stays, and increased mortality rates [4]. Thus, the detection and quantification of these sulfonamides is an important task.

Normally, quantification is performed in a laboratory setting, and this makes it difficult to track and monitor compounds in the wild, so it is important to find a portable method that can perform tests immediately.

1.1 Motivation

According to European Centre for Disease Prevention and Control, there has been a rapid and continuing rise in antibiotic-resistant infections [5] and according to the Centers for Disease Control and Prevention, at least 2.8 million people in the U.S. contract an antibiotic-resistant infection each year [6]. Considering that this antibiotic problem is a worldwide problem, it is therefore important to find suitable methods to detect these bacteria so that appropriate methods

are used to prevent and eliminate their spread.

The conventional method of detecting and quantifying sulfonamides in water involves a researcher collecting a sample of water, transporting that sample to a laboratory, and then calculating the concentration using difficult and expensive equipment. This type of method makes it even more difficult to detect and control substances in a wild environment or in remote regions where the nearest laboratory is not as close to the water source.

In 2019, Carvalho et al. [7] developed a new method for the detection of sulfonamides in water. This method is based on digital colorimetry and is simpler and less expensive than standard methods for detecting antibiotics. While their method was a great advance in detecting sulfonamides in a wild, the algorithm used can be outperformed by using machine learning methods. These machine learning models are improving rapidly and can perform faster and obtain more accurate results.

1.2 Objectives

The objective of this dissertation is to create a computer vision algorithm for the determination of sulfonamides in water that improves on the work of Carvalho et al. [7] by using machine learning models that can also be easily adapted in a smartphone application.

Given an image similar to figure 1.1 four machine learning methods will be applied, each one focusing on each of the following areas:



Figure 1.1: Example of image in the dataset.

- **Detection of the color checker patches:** Detection of the position of each color patch in the color checker, so we can use that position to extract the color of each patch (figure 1.1);
- **Detection of the disk and its color:** Similar to the point above, detection of the position of the disk and the color, and use the position of the color to determine its color (figure 1.2b);

- **Color correction of the image using the color checker patches:** When taking pictures with a camera, sometimes the colors are not the same that we can see them with our eyes. Therefore, it is important to correct the image to obtain the true color of the color sample inside the disk. This color correction makes possible a more accurate determination of the concentration value of the sulfonamides. The colors extracted from the color patches are used to color correct the image, and this correction is applied to the color inside the disk;
- **Calculating the concentration of Sulfonamides:** Calculation of sulfonamides concentration using digital colorimetry, which determines the concentration value based on the color of the disk.

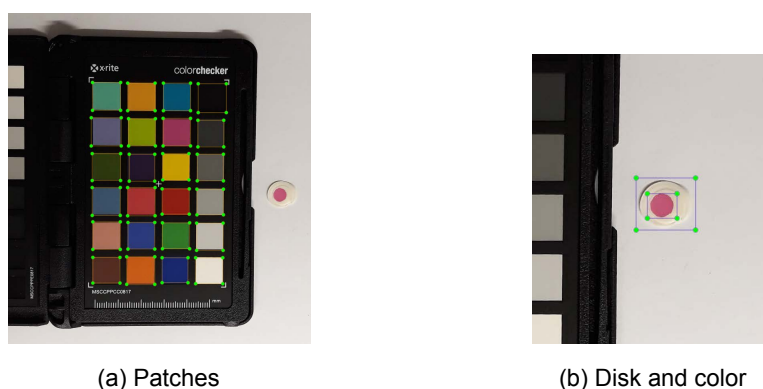


Figure 1.2: Detections of the machine learning models.

1.3 Contribution

At the end of this dissertation, we have a machine learning model for each of the objectives defined above:

- Model for the detection of the color checker patches;
- Model for the detection of the disk sample;
- Model for the color correction of the color of the disk;
- Model for quantification of sulfonamides concentration.

1.4 Organization

This dissertation is divided into the following chapters:

- Chapter 2 - Literature Revision: In this chapter we describe the current state of the art on which our work is based;
- Chapter 3 - Baseline: In this section we establish the baseline against which we will compare our work;
- Chapter 4 - Machine Learning Methods: Here we describe how the machine learning models were created, what datasets they contain and what results they produce;
- Chapter 5 - Conclusions and Future Work: The last chapter where we make an evaluation of the work done and make suggestions for future work.

Chapter 2

Literature Revision

This literature revision will be divided into 3 sections:

- Detection and quantification of sulfonamides - What methods are traditionally used to detect and quantify sulfonamides;
- Object Detection - Comparing machine learning models versus traditional techniques for object detection and segmentation;
- Color Correction - Exploring traditional and machine learning methods currently used to color correct images.

2.1 Detecting and quantifying sulfonamides

There are several methods for the detection and quantification of sulfonamides. The following are the most common and are all performed in a laboratory.

- High Performance Liquid Chromatography (HPLC) [8] - This method is one of the most common methods for the detection and quantification of sulfonamides. The method can be divided into two processes. The first process is to pass a pressurized liquid solvent that separates the components of the sample, and the second process is a mass spectrometric analysis of the separated components;
- Capillary Electrophoresis - Separates the components based on their electrophoretic mobility (atomic radius, viscosity and charge of the molecule) [9]. This method is a better alternative to the previous ones as it can be used for small amounts of a sample, gives faster results and allows a high-resolution separation;

- Microbiological Assays - Microorganisms can only grow and multiply under specific conditions. Therefore, Bilandžić et al. [10] has developed a method using sulfonamide-sensitive bacteria to determine the presence of sulfonamides. Since the bacteria are sensitive to sulfonamides, they are placed in an environment containing milk with suspected sulfonamides to multiply and we measure how much growth has been suppressed compared to a normal environment without sulfonamides. This type of approach serves as a screening method for finding sulfonamides rather than calculating their concentration;
- Spectrophotometric - Is a technique that measures the amount of chemicals present in a solution by measuring the light absorbed by the sample and was developed by El-Dien et al. [11]. It is a simple method that, like the previous method, is best suited for screening sulfonamides;
- Immunoassay - Is a biochemical test that uses an antibody to produce a reaction in which the antibody binds to a specific target structure. Thanks to Li et al. [12], an antibody has been developed that can bind to the sulfonamide structure. This type of test can be performed with small samples and in a short time.

Recently Carvalho et al. [7], develop a method that used digital colorimetry to detect the concentration of sulfonamides concentration. Digital colorimetry is the method of measuring the absorbance of wavelengths of light and correlate them to a concentration of the detected substances of a sample. Their method consisted in:

1. Using a reagent in a water sample to obtained a specific color depending on the concentration of sulfonamides in the sample;
2. Taking a photo of the sample with a color reference target (*x-rite ColorChecker Passport*) next to it to be able to eliminate the illuminations differences when taking a photo;



Figure 2.1: Image of the color checker and the sample Carvalho et al. [7].

3. Segmenting the colors from the reference target to color correct the image - To accomplish this they needed to first segment the color checker (figure 2.2), then segment the patches

(figure 2.3) and finally defining a small bounding box inside each patch where the color will be retrieved (figure 2.4);

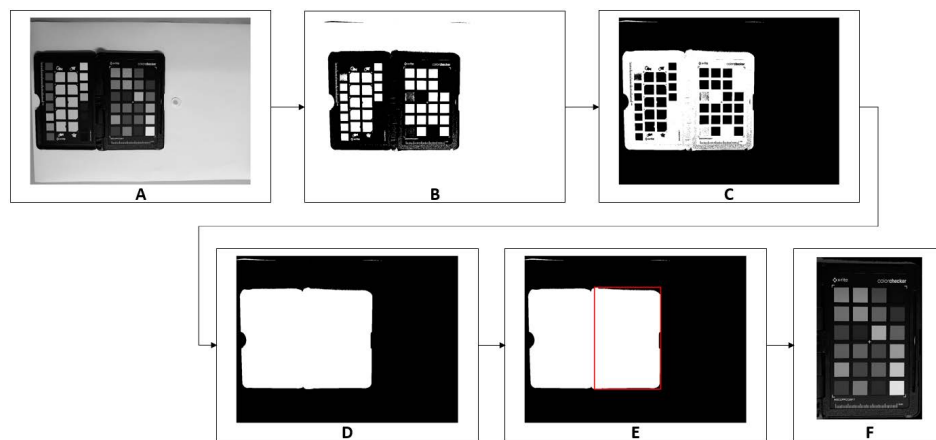


Figure 2.2: Segmentation process of the color checker from Carvalho et al. [7].

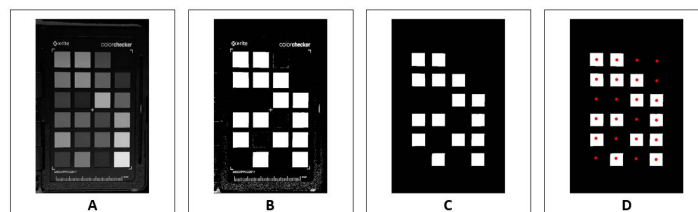


Figure 2.3: Segmentation process of the patches from Carvalho et al. [7].

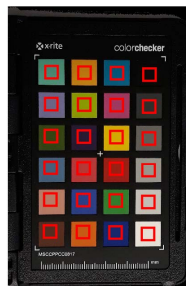


Figure 2.4: Bounding boxes of the patches where the color will be extracted from Carvalho et al. [7].

4. Segmentation of the disk and color (figure 2.5);
5. With the colors extracted they performed a least square regression between their RGB and XYZ values. This regression allow them to color correct the image (figure 2.6);
6. And finally they utilize the G or H color component curves (G for lower concentrations and H for higher concentrations) to calculate its sulfonamides concentration value.

Their method of performing disk and patches segmentation uses traditional computer vision image processing methods such as thresholding, dilation and erosion. These techniques can



Figure 2.5: Segmentation process of the patches from Carvalho et al. [7].



Figure 2.6: Original image(left) and color corrected image (right) from [7].

work well when dealing with very similar images, but they are difficult to adapt to all types of images like images with different light illuminations, quality of the camera, presence of shadows and blurriness or inability of the user when taking a photograph.

According to Mahony et al. [13], deep learning now achieves higher accuracy in object detection and segmentation by overcoming the problem of having different type of images. Since these models are trained rather than programmed, they are more flexible and can be retrained and adapted each time different data is encountered.

2.2 Object Detection

This dissertation was developed alongside another student that was developing the app for this work, and because of that we needed to use models that could be converted into a smartphone application. The model chosen is the model developed by Howard et al. [14].

In Howard et al. [14] they describe a small and lightweight model name Mobile Net that can be used for detection of objects. Instead of using convolutional layers, depth wise separable convolutional layers are used. This type of layer gives almost exactly the same result as the regular convolutional layer, but instead of using only one layer, they split it into two. First, a depth-wise convolutional layer is used, which performs convolution for each channel separately, instead of the usual one channel per pixel. The second and final layer is a pointwise convolution, which sums all channels.

Although this process is a two-step process, the method is actually faster and easier than the normal convolutions because the normal convolutions require more computational work.

In 2018, researchers at Google released version 2 of Mobile Net [15], which introduces two additional features. A linear bottleneck between layers and shortcut connections between bottlenecks (figure 2.7). With this introduction, the models achieve the same accuracy but are faster than the first version.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Figure 2.7: Architecture of the MobileNetV2 from Sandler et al. [15].

2.3 Color correction

Before we talk about color correction algorithms, we must first understand what color is and why we need to correct it. Color is not real and does not exist. In order for our minds to create color, we need our eyes, an object, and most importantly, light. Light is made up of electromagnetic waves, and when they hit an object, that object absorbs some and reflects others. These reflected waves determine the colors we see. To create the color, the light must interact with the light receptors in our eyes, which then send messages to the brain.

Cameras also work similarly to human eyes as seen in figure 2.8. The creation of a digital photograph begins with the opening of the shutter, then the light reaches the sensors and the shutter closes again. The sensors consist of photodiodes that react to the light reflected from objects and convert this information into an electronic form of millions of bits and bytes.

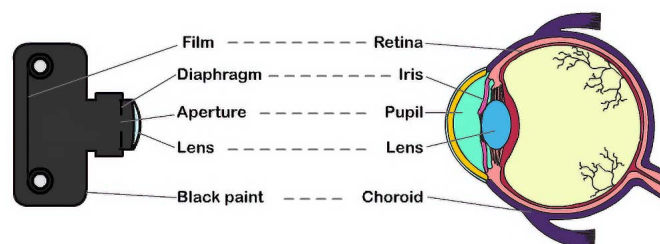


Figure 2.8: Eye vs Camera from ScienceWithMe! [16].

One big difference between cameras and the human eye is that cameras record light as

it actually is, while our brains always color correct colors. If we are in a room illuminated with fluorescent lights that produce a greenish-yellow light, all the reflected light will also have a greenish-yellow tint. Our brain can understand that the reflected colors are not the “true” colors of the image, so it makes some color corrections to remove that greenish-yellow tone (we describe the “true” colors as the colors reflected from white light). If the cameras do not have a color correction algorithm, they always capture the colors with the tone that the light has. This causes a problem because if we shoot the same object with the same camera in two rooms with different light, the colors in the image will be different.

Color correction algorithms help with this problem. They correct colors so that the colors of an object remain consistent, regardless of the light used to capture them. There are several different color correction algorithms. We will refer to learning methods as methods that use machine learning to correct the image, and traditional methods as all other methods.

2.3.1 Traditional methods

2.3.1.1 White Balance

White balance is an algorithm commonly used in image color correction. This algorithm removes color casts so that objects that appear white to our eyes when photographed remain white in the image.

There are several ways to implement this type of algorithm. The simplest are those commonly used in phone cameras, where they have pre-defined options, such as “cloudy”, “fluorescent”, “direct sunlight” and “auto” (figure 2.9), where the user can choose the situation in which to take the photo. The option selected will then add a preset color value to the image to color correct it. Cameras may also have options to manually set a color temperature for the photo. These types of options are good for correcting the photos to look natural but they will not color correct the colors to their true colors.










	Automatic White Balance
	Daylight (5600K)
	Shadow (7000K)
	Cloudy (6000K)
	Tungsten (3200K)
	Fluorescent (4000K)
	Flash (5500K)
	Custom White Balance
	User Defined

Figure 2.9: Example of automatic white balance options from Icon Photography School [17].

Some other more complex than the previous white balance algorithms are:

- **The gray world assumption:** Buchsbaum [18] states that the given a white light source the average of the red blue and green values of a image average to a gray color. So given a new image we can calculate the average color and compare it to the gray color;
- **Gray-Edge:** Van De Weijer et al. [19] developed a similar hypothesis to the above, but instead of using all the scene to calculate the different, it assumes that the edge different from that scene averages to a gray color;
- **White patch:** Land [20] searches for a white pixel and uses the difference between that pixel and a true white pixel and applies that difference to the rest of image.

Normally white balance algorithms are used automatically before others algorithms such as in He et al. [21] where white balance is used before an algorithm of dehazing in Jingchun Zhou and Zhang [22] uses an automatic white balance based on the work of Ching-Chih Weng et al. [23]. White balance can also be adapted to like in Gasparini and Schettini [24]. They propose an unsupervised way to distinguish between a true cast or a predominant color in an image and remove the cast when found. The white balance algorithm is used after the determination of the cast and is not applied only to a single point, to prevent choosing the wrong region. Their algorithm chooses several points that on average are white and applies the white balance to those points.

2.3.1.2 Other Methods

Regression models can also be used to color correct images. Li et al. [25] wanted to create a model to quantify the number of days a bruise has. Their experiment consisted of taking a photo of the color chart, obtaining the color profile information (ICC), taking a photo of the skin, correcting the color of the skin with the ICC profile, and analyzing the color of the bruise shown in figure 2.10. The ICC profile is obtained using a polynomial regression calculation method and the photographs were taken using a LED ring light to obtain more uniform illumination. A dark chamber module was also used between the lens and the sample. While their experiments produced excellent results, they had a very controlled environment. Unfortunately, our model needs to be more robust than this. We need to develop a system that will allow us to photograph in a wild environment and still get excellent results. So, we will have to use a color correction algorithm that is stronger than the proposed one.

In You et al. [26], the authors aim to develop a method to evaluate the quality of chicken meat. Their experiment consisted in taking a photo of the chicken with a color tester next to it, then a color correction algorithm is applied and finally the color is classified into one of the 3

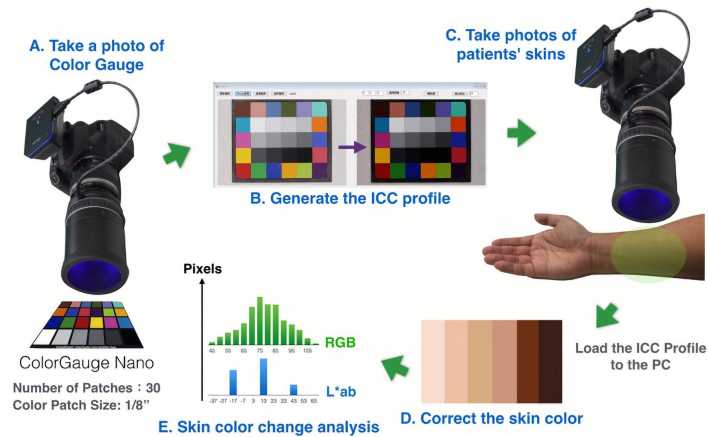


Figure 2.10: Diagram of the procedure from Li et al. [25].

clusters representing the freshness of the meat. A multivariate linear regression model is used in for color correction. The parameters are calculated by comparing the color values of the extractor with the standard colors of the color tester. Since they only want to cluster the colors, there is no need to accurately represent the true color value.

2.3.2 Learning Methods

In recent years, there has been increasing interest in deep learning, a subfield of machine learning in which machines learn by example. This technology involves a multilayered structure of algorithms called neural networks. These networks consist of node layers that contain an input layer, several hidden layers, and an output layer as seen in figure 2.11. Each node has a weight and data is sent through the nodes based on certain thresholds. If this threshold is not met, the data is not forwarded to other nodes. By filtering the data in this manner, neuro-networks function in a similar way to the human brain.

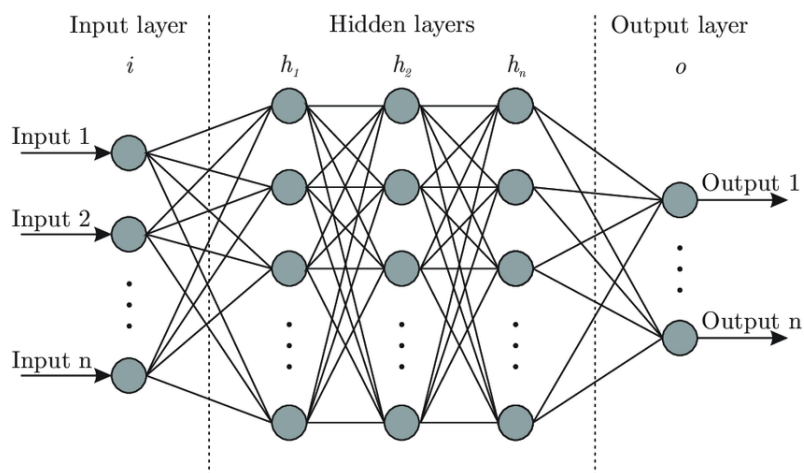


Figure 2.11: Example of a neuro networks from Bre et al. [27].

The reason why this type of learning became so popular is because of the networks ability to perform feature extraction for us. Feature extraction require a very detailed knowledge of the problem domain to extract pertinent features and sometimes it is even impossible to extract some features. Because the layers of the neuro-network learn these features implicitly, we have much more complex features than a human could ever extract. It is this feature that allows these types of learning to achieve excellent results and even outperform humans. In 2019, for example, Ardila et al. [28] developed a Deep Learning algorithm that could accurately predict the risk of lung cancer based on computed tomography scans, outperforming all six expert radiologists.

In Chinese medicine, the patient's tongue is used as a diagnostic method. Therefore, Lu et al. [29] wanted to develop a model that color corrects the images for quantitative analysis or to display in the doctor's computer. They use a convolutional neuro network (figure 2.12) with two phases. In the offline phase, they trained the network with a color checker where it learns how to correct colors in the image, and then in the online phase, they use the network to correct the image of the tongues.

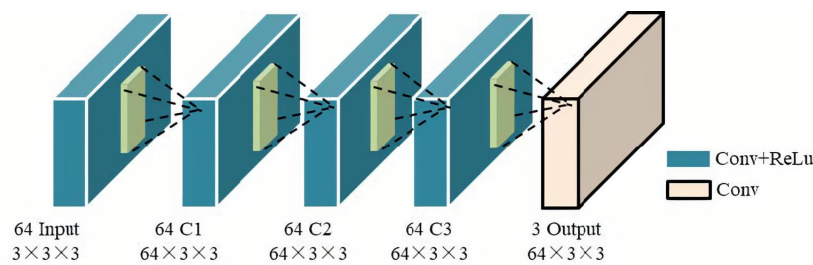


Figure 2.12: Architecture of the model from Lu et al. [29].

The devices used to capture the tongue images are not normal handheld cameras. These devices were developed by the Signal and Information Processing Lab at Beijing University of Technology. They are specially designed so that the patient's head rests on a support and the brightness of the device can be adjusted as show in figure 2.13. This means that their algorithm does not have to be very robust, as there will not be much variation in the photos taken, unlike our problem.

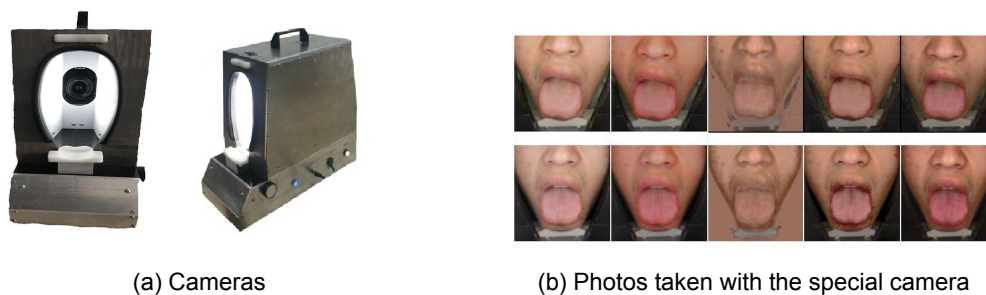


Figure 2.13: Camera and dataset from Lu et al. [29].

Robin Kips and Perrot [30] wanted to estimate the color of the skin of an image taken in a wild environment. They take photos with multiple phones in different lighting conditions, with the participant holding a color checker. Their model, which they called LabNet, is divided into 2 parts. The first part is a neuro-network inspired by the ResNet architecture [31] and the second part is a dense layer with ReLU activations as show in figure 2.14. Their aim with their model is to implicitly learn to color correct and segmentate the skin, so the only preprocessing step used is to do a face alignment. The results were impressive and even outperformed conventional skin color estimation techniques. They concluded that this kind of models can be used for color estimation problems even without explicit color correction.

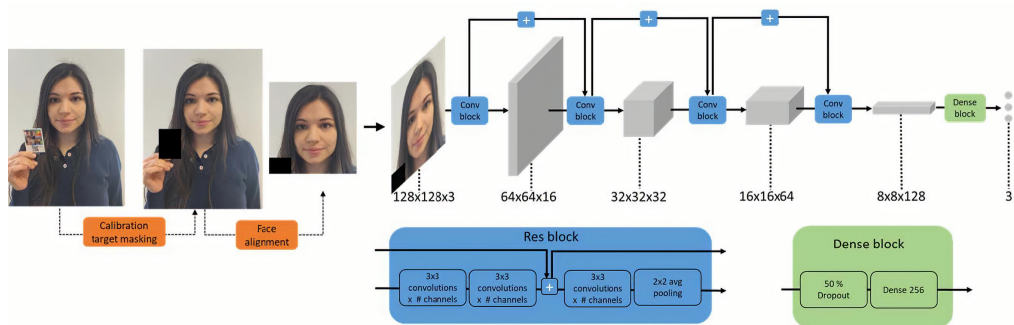


Figure 2.14: Architecture of the model from Robin Kips and Perrot [30].

And finally, Lou et al. [32] proposes a new deep learning model that could estimate the light source. Their model uses Deep Neural Networks (DNNs) with five convolutional layers and three fully connected layers as seen in figure 2.15. Their results show that they were able to obtain accurate results in real world datasets, outperforming the state of the art results by 9%.

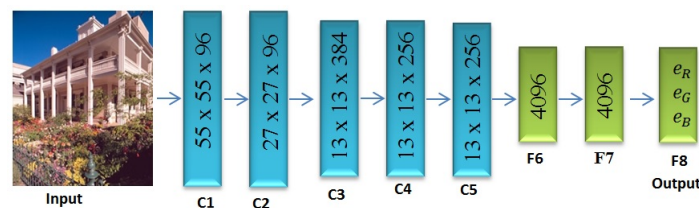


Figure 2.15: Architecture of the model from Lou et al. [32].

2.4 Summary

The detection of sulfonamides in water has become an important issue in recent years because of the health problems that arise when people come into contact with bacteria that are resistant to these antibiotics. Therefore, several methods have been developed to detect sulfonamides in water. These methods can only be performed in a laboratory, which limits their practicality if the environment in which the tests are performed is far from these laboratories. Fortunately, a new practical method has been developed in the form of digital colorimetry. Although this method

gives good results, it can be replaced by machine learning models. Machine learning models have been improving at a rapid rate allowing them to perform detections and color correcting images more accurately than the old algorithms.

Chapter 3

Baseline Method for Color Correction on Uncontrolled Lighting Conditions¹

In Carvalho et al. [7], they proposed a new approach for antibiotic pollution monitoring, using digital colorimetry in conjunction with smartphones for an accessible and mobile application. The study focused on the estimation of sulfonamides (a family of antibiotics) concentration in contaminated water samples. The color of the sample was corrected using a reference target, so that there is color constancy between images with different illuminations and from different devices. To estimate the concentration of sulfonamides, a calibration curve is used, correlating concentrations to a color value. They found that the a^* color value from the *CIE Lab* color space and the hue (H) color value from the *HSV* color space provided the best correlation between color and sulfonamide concentrations [7].

For Carvalho et al. [7], a dataset with photographs of the *x-rite ColorChecker Passport* next to a sample of contaminated water was prepared in a laboratory setting, with the color chart used as the reference for the color correction and the target being the sample. Figure 3.1 is an example of a photograph from this dataset and figure 3.2 shows the colors the samples assume depending on the concentration of sulfonamides.

Although they achieved good results, they still detected a high standard deviation between color corrected images, especially when more contrasting illumination conditions were present. Therefore, we wanted to explore other variations of our color correction method to minimize the influence of different illuminations, providing a more stable color correction even in extreme conditions. Besides, we wanted to validate this approach by testing independently of the samples. Using water samples might contribute to more variance in the results, since it's preparation in a laboratory is more prone to human error. To do this, we used different patches from the *x-rite ColorChecker Passport* as the targets, instead of the samples.

¹Parts of this chapter were published as a scientific paper



Figure 3.1: Example photograph from dataset of Carvalho et al. [7].



Figure 3.2: Examples of samples of varying sulfonamide concentrations, from 0 $\mu\text{g/L}$ (left) to 150 $\mu\text{g/L}$ (right).

3.1 Methodology

The objective of this work is to improve upon the color correction algorithm of Carvalho et al. [7]. Color correction is needed to ensure color is consistent between photographs under different illuminations or from different devices. In Carvalho et al. [7], color correction is done using the classic color chart of an *x-rite ColorChecker Passport*, which is a reference chart with 24 patches of different colors arranged in a 6 by 4 grid, with known ground truth values. With a least squares method, they minimize the difference between the detected color of the 24 patches and the ground truth. Other works have demonstrated different approaches such as:

- **Using only 13 of the 24 color chart patches:** Alsam and Finlayson [33] showed that color correction with only 13 patches is comparable to using the 24 patches;
- **Using more parameters in the least squares minimization:** In Finlayson et al. [34], it is shown that more parameters in the least squares minimization for the color correction leads to better results.

Inspired by these two works, we hypothesize that choosing the 13 patches closest to the color of the target and using more parameters in the least squares minimization will improve upon their previous work.

3.1.1 Data

For this work, a database was built with 24 photographs (see figure 3.3 for examples) of the *x-rite ColorChecker Passport* in various illumination conditions:

- inside with different white lights (9 images);
- inside with different yellow lights (6 images);
- inside by the window (3 images);
- outside in the sun (3 images);
- outside in the shade (3 images).

These new images provide more contrasting conditions of illumination when compared to the first dataset that was captured in a laboratory setting, making the color correction more challenging, but more prepared for real world use. To further simulate practical use, the photographs were captured free hand, with the only restriction being encompassing the reference and target color patches from above, as close to 90 degrees as possible.



Figure 3.3: Example photographs from the new dataset. Taken inside with a white light (left), inside with a yellow light (middle) and outside in the sun (right).

In Carvalho et al. [7], the target of the color correction was the sample of contaminated water. Here, the targets are the 8 colored patches (red, orange, yellow, green, cyan, blue, violet and magenta) in the page opposite of the classic color chart that is used for the color correction, see figure 4.6.

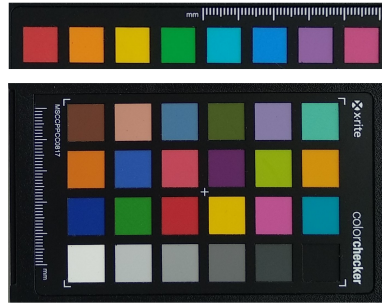


Figure 3.4: Example of the 8 targets of the color correction (top) and the 24 patches used as reference for the color correction (bottom).

3.1.2 Color Correction

A color correction matrix M_{cc} is used to transform the original image I , resulting in a color corrected image I_{cc} , as shown in equation (3.1).

$$I_{cc} = M_{cc}I \quad (3.1)$$

M_{cc} is found by solving the following minimization problem:

$$M_{cc} = \arg \min_T \|C_{XYZ} - M_{cc}C_{RGB}\|^2 \quad (3.2)$$

M_{cc} is calculated using a least-squares method to find the difference between the measured RGB values of the color chart patches, C_{RGB} , in each image and the corresponding ground truth XYZ values, C_{XYZ} , see equation (3.3). Transforming the image using M_{cc} will result in a color corrected image in the XYZ color space, which is a device independent color space.

$$M_{cc} = (C_{RGB}^T C_{RGB})^{-1} C_{RGB}^T C_{XYZ} \quad (3.3)$$

For a 3×3 color correction matrix and using all color chart patches, C_{RGB} and C_{XYZ} are both the same size, 24×3 matrices. Each line represents a patch RGB or XYZ value. In Carvalho et al. [7], all 24 patches were used to find M_{cc} , and for this work, we compare it with using only 13 patches, making C_{RGB} and C_{XYZ} 13×3 matrices.

3.1.2.1 13 Patches

To choose the 13 patches, the RGB values of each patch are assumed as a 3-dimensional coordinate and the euclidean distance between each of the 24 patches and the RGB value of

each of the 8 target colors is calculated, and the 13 closest patches are chosen. Equation (3.4) shows how each distance D is calculated, with P_R, P_G and P_B being the RGB values of each patch and T_R, T_G and T_B the RGB values of the targets. figure 3.5 shows the 13 patches chosen for each target.

$$D = \sqrt{(P_R - T_R)^2 + (P_G - T_G)^2 + (P_B - T_B)^2} \quad (3.4)$$

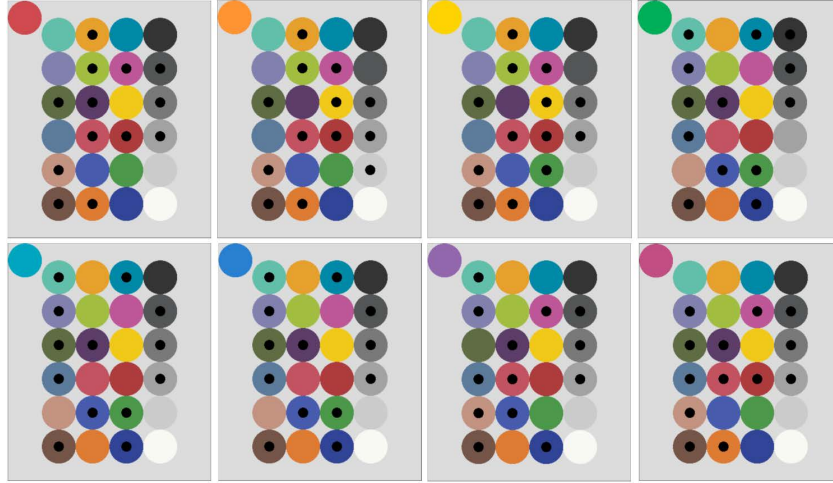


Figure 3.5: Selected patches for each of the 8 targets. Each target indicated on the top left, and the black dots indicating the 13 chosen patches from the 24 color chart.

3.1.2.2 Polynomial Extensions

More complex least squares minimization can be done by adding polynomial terms to C_{RGB} , making it a $24 \times N$ or $13 \times N$ matrix, with N depending on the number of parameters added. In this work, the more commonly used $\{R, G, B\}$ ($P1$) is tested along with the polynomial extension ($P2$) and the root-polynomial extension ($P3$). The terms for these extensions are shown in equations (3.6) and (3.7), respectively.

$$P1 = \{R, G, B\} \quad (3.5)$$

$$P2 = \{R, G, B, R^2, G^2, B^2, RG, GB, RB\} \quad (3.6)$$

$$P3 = \{R, G, B, \sqrt{RG}, \sqrt{GB}, \sqrt{RB}\} \quad (3.7)$$

Other polynomial terms were tested, however the results were consistently worse than these three and, therefore, discarded from this study.

The baseline method employing all 24 patches of the color chart and with $C_{RGB} = \{R, G, B\}$ is compared with the use of only the 13 closest patches to the target and the polynomial and root-polynomial extensions.

3.1.2.3 Metrics

The metric used to compare the results is the standard deviation (*Std*) of the target values between images. Lower values meaning that the colors of the targets in different color corrected images are closer to each other.

3.2 Results

In [7], we found that the color components that can best differentiate sulfonamides concentrations are the a^* component from the *CIE Lab* color space and the hue (H) from the *HSV* color space, therefore the results are focused on these two values.

Tables 3.1 and 3.2 showcase the *Std* of the target values when using 24 or 13 patches and different polynomial terms:

- P1: $\{R, G, B\}$
- P2 the polynomial extension: $\{R, G, B, R^2, G^2, B^2, RG, GB, RB\}$
- P3 the root-polynomial extension: $\{R, G, B, \sqrt{RG}, \sqrt{GB}, \sqrt{RB}\}$

Table 3.1 shows that, for a^* , 13 patches usually provide better results, with only the orange, yellow and violet patches having the best result with 24 patches. The polynomial extension *P2* with 13 patches has the lowest average, with a 2.56% improvement over the baseline (*P1* with 24 patches). The *Std* for this method shows a 32.01% improvement relative to the baseline. Table 3.2 shows that for H , the results are similar, with 13 patches providing lower standard deviations, with the exception of the Red target. On average the best case for H is the simplest, *P1* with 13 patches. Compared to the baseline, the average improves by 18.58%, while the *Std* improves by 59.85%. These results prove that the new methods are more stable, as the *Std* is significantly lower than in the baseline.

With these results, it is clear that using the 13 patches closest to the target's *RGB* values produces a more precise color correction, with less variation between the targets in different

a^*	Patches	Red	Orange	Yellow	Green	Cyan	Blue	Violet	Magenta	Average (Std)
P1	24	2.632	0.613	0.915	0.830	1.100	1.205	0.825	1.242	1.170 (0.628)
	13	2.554	1.035	0.987	0.558	0.675	0.947	1.329	1.72	1.226 (0.648)
P2	24	1.477	1.309	1.275	1.196	1.417	2.046	0.651	0.919	1.286 (0.410)
	13	1.111	0.803	1.382	1.606	0.884	1.848	0.696	0.791	1.140 (0.427)
P3	24	3.552	1.135	1.031	0.826	0.916	1.271	1.007	1.640	1.422 (0.896)
	13	0.905	1.614	1.046	1.751	0.785	1.912	0.936	0.863	1.227 (0.454)

Table 3.1: Standard deviations of the a^* color value for each of the target patches.

H	Patches	Red	Orange	Yellow	Green	Cyan	Blue	Violet	Magenta	Average (Std)
P1	24	0.635	0.947	0.552	1.628	0.439	0.758	1.259	0.759	0.872 (0.396)
	13	0.880	0.796	0.547	0.716	0.436	0.640	0.788	0.880	0.710 (0.159)
P2	24	0.550	0.530	0.562	0.869	0.741	1.233	0.708	0.818	0.751 (0.232)
	13	0.627	0.427	0.651	1.812	0.435	0.900	0.544	0.534	0.741 (0.458)
P3	24	1.265	0.639	0.548	1.021	0.507	0.934	1.518	0.800	0.904 (0.356)
	13	0.561	0.569	0.513	1.532	0.436	1.105	0.611	0.494	0.728 (0.386)

Table 3.2: Standard deviations of the H color value for each of the target patches.

photographs. However, the best polynomial extension varies for each color target, making it a case by case selection.

In our case, the real world samples can vary in color from yellow to a dark magenta, almost red, in the highest sulfonamides concentrations (figure 3.2). We decided to study the results with only the closest targets to our samples. Using the euclidean distance shown in equation (3.4), we calculated the distance of the 8 targets to random samples (with concentrations 0 $\mu\text{g/L}$, 20 $\mu\text{g/L}$, 50 $\mu\text{g/L}$ and 100 $\mu\text{g/L}$) and found the closest targets to our real world application to be the red, orange, yellow, violet and magenta patches. Tables 3.3 and 3.4 show the results focused on these 5 targets.

For the a^* color value, the polynomial extension $P2$ with 13 patches gives the lower average, while, for the H color value, the root-polynomial extension $P3$ with 13 patches is the lowest. In this case, the improvement for a^* is 23.21% on average and 64.81% on the Std . As for H , we can see a decrease of 33.73% on average and 83.33% for the Std . These improvements are proof that these methods are more stable, meaning less influenced by contrasting illumination conditions.

Thus, for our application, the best setup for color correction depends on which color value provides the best correlation between color and sulfonamides concentration.

a*	Patches	Red	Orange	Yellow	Violet	Magenta	Average (Std)
P1	24	2.632	0.613	0.915	0.825	1.242	1.245 (0.807)
	13	2.554	1.035	0.987	1.329	1.721	1.525 (0.645)
P2	24	1.477	1.309	1.275	0.651	0.919	1.126 (0.335)
	13	1.111	0.803	1.382	0.696	0.791	0.956 (0.284)
P3	24	3.552	1.135	1.031	1.007	1.640	1.673 (1.081)
	13	0.905	1.614	1.046	0.936	0.863	1.073 (0.310)

Table 3.3: Standard deviations of the a* color value for the red, yellow, violet and magenta target patches.

H	Patches	Red	Orange	Yellow	Violet	Magenta	Average (Std)
P1	24	0.635	0.947	0.552	1.259	0.759	0.830 (0.282)
	13	0.880	0.796	0.547	0.788	0.880	0.778 (0.136)
P2	24	0.550	0.530	0.562	0.708	0.818	0.634 (0.125)
	13	0.627	0.427	0.651	0.544	0.534	0.557 (0.088)
P3	24	1.265	0.639	0.548	1.518	0.800	0.954 (0.419)
	13	0.561	0.569	0.513	0.611	0.494	0.550 (0.047)

Table 3.4: Standard deviations of the H color value for the red, yellow, violet and magenta target patches.

3.3 Conclusions

Promising results were achieved in this work, with a lower standard deviation between different color corrected images compared to previous work. Confirming that, with fewer but more relevant patches, the color correction has more resolution in the targets color range. This improvement means that smaller differences in sulfonamides concentration can be detected, making digital colorimetry a more reliable method for monitoring environmental water pollution. As for the polynomial expansions, the results are not as straight forward, with each choice (*P1*, *P2* and *P3*) being the best for different targets. This makes it a case by case decision on which to use, depending on the real world application.

Chapter 4

Machine Learning Models

This thesis was developed as part of the *INESC TEC S-MODE* project, so we had some constraints and limitations to consider:

- We could not modify the original dataset - We had to work with the images that were provided to us, and could not change how the images were taken to possibly facilitate the development of the machine learning models;
- We want to be able to measure the concentration of sulfonamides in an environment outside of a lab - So we need to be able to develop a solution that can be converted and run on a smartphone since its portable and their computational power has been increasing;
- To facilitate the conversion to a smartphone app, each step of the algorithm will be modular - So that in the future, as new solutions emerge, we can simply replace one old machine learning model with another.

Our methodology is similar to Carvalho et al. [7]. Since we have an image as a color checker and a disk, we will use the same detection ideas to detect and extract the colors, and we will also use the colors from the color checker to create a model that learns how to color correct them. The thought process to solve our problem started with detecting the color checker and its patches, detecting the disk and its color, and then color correction. This color correction model will be different that was done before. We want to create a machine learning model that receives an image and color corrects it. So, we will create an image with the colors of the patches and the color of the disk, and create a model that receives that image and color corrects it. Then all we have to do is extract the color of the disk and pass it to the model that calculates the sulfonamides concentration. The solution thus developed is shown in figure 4.1.

In order to compare Carvalho et al. [7] results with ours, we converted their MATLAB code to Python to familiarize ourselves with the problem, understand if there are some issues that can

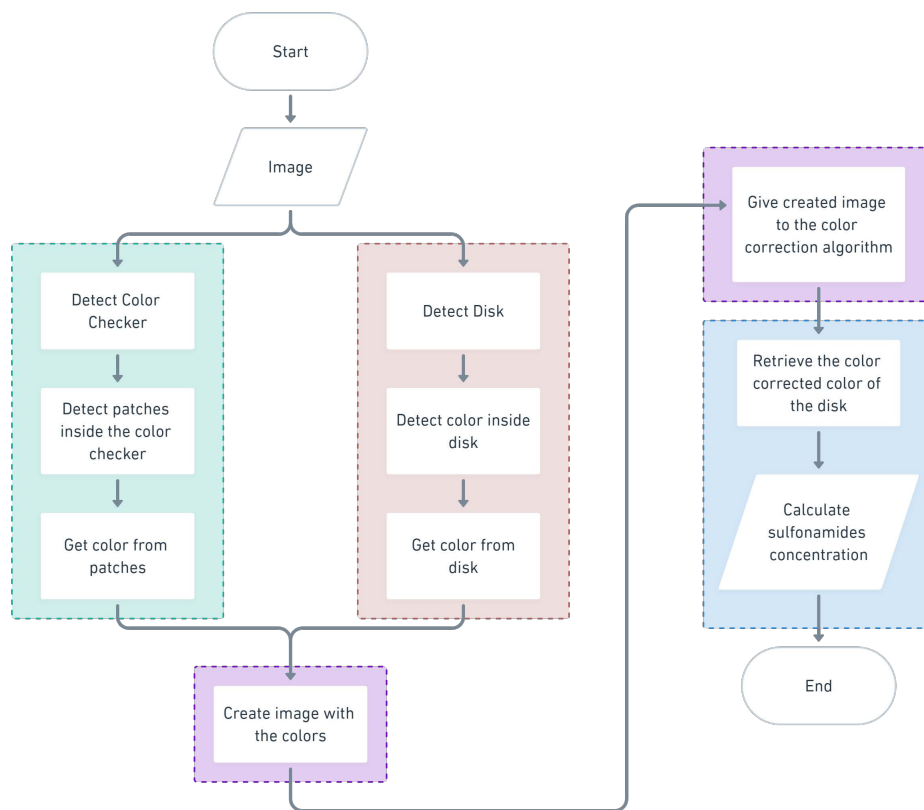


Figure 4.1: Flowchart Methodology.

be solved by machine learning, and help create the datasets between processes. The decision to convert to python was because we were more familiar with this language.

The S-Mode project partners created the original image dataset by photographing the disks and color checkers with different smartphones under different lighting conditions, as shown in figure 4.2.

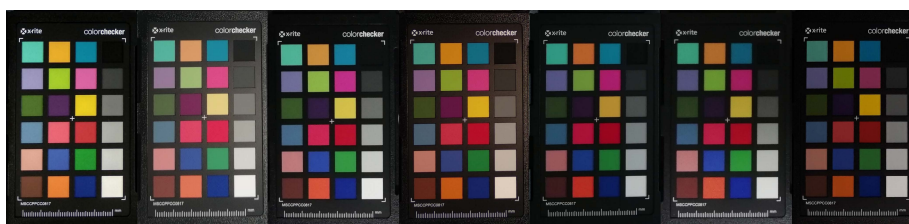


Figure 4.2: Example of different lighting conditions.

And in the following results sections we will refer the machine learning models as Machine Learning Model (MLM), the segmentation algorithms as Traditional Segmentation Techniques (TST) and the algorithm that we develop in chapter 3 as Traditional Correction Model (TCM).

4.1 TensorFlow and TensorFlow Lite

The TensorFlow Object Detection API [35], shown in figure 4.3, was used to build the machine learning models. This framework has several pre-trained models that were trained on the COCO 2017 dataset [36]. These models can be used out-of-the-box if we want to detect objects that appear in the COCO dataset, but we can also train the models from scratch using our own objects. Since the objects we want to detect are not the objects normally used in machine learning, such as cats, dogs, people, cars, and so on. We will train the models from scratch using our own datasets to detect patches and disks.

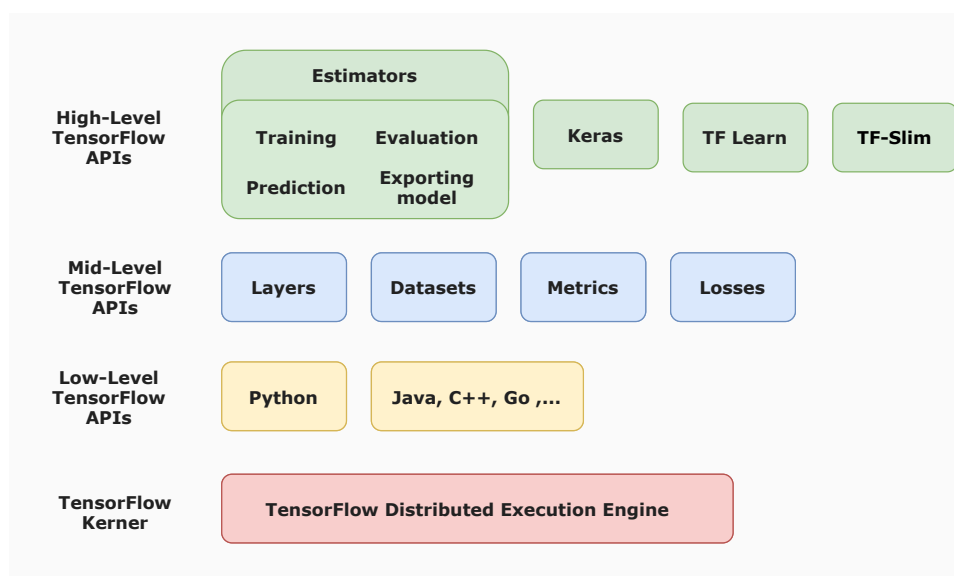


Figure 4.3: TensorFlow Object Detection API.

Since we are using the TensorFlow API, its models use a specific input type. This input is a TFRecords. These TFRecords are a proprietary binary storage format of TensorFlow and it is this binary data that is fed into the model. This conversion to TFRecords improves the training time of our models because binary data is smaller, faster to read, and optimized for use with TensorFlow so that data from larger datasets only needs to be loaded when needed.

Once we have trained our models, we need to convert them into a smartphone-compatible model. To do this, we convert our model into a TensorFlow Lite model, as shown in figure 4.4. These models are specifically designed for use on mobile devices, as they are faster, smaller, and less computationally intensive. When converting to a TFLite model, TensorFlow Lite takes the SavedModel output of our machine learning model as input, compresses it, and interprets it for a mobile application. There are some optimizations and operators that can be used in this conversion to improve the performance of the model on a smartphone, but we will get into that later when describing the machine learning model conversion.

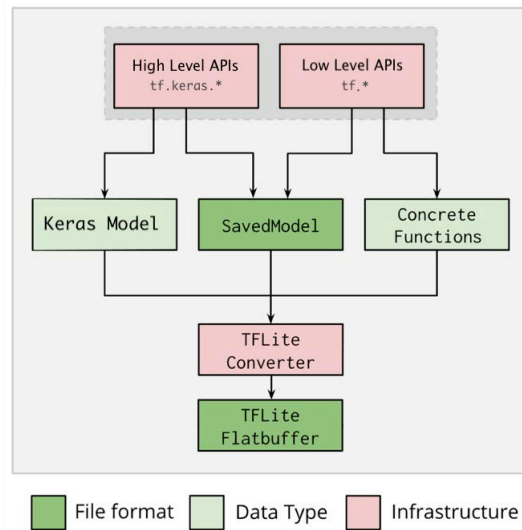


Figure 4.4: TensorFlow Lite Converter from TensorFlow [37].

4.2 Patches Detection

The goal of this model is to be given an image of a color checker and detect the position for each color patch (figure 4.5), so that when integrated into a mobile application, segmentation methods can be applied to obtain the color from the color checker positions.



Figure 4.5: Detection of the patches.

4.2.1 Building training dataset

To make the training phase faster and less computationally intensive, we downsize all images captured by the S-mode project partners to a width of 512 and a height of 384 before processing them.

In order for the machine learning models to learn to detect the patches, we need to annotate each image in the training dataset with the location of each patch. To avoid the need for humans to annotate each image, we modified the code developed by Carvalho et al. [7], which already performs the patch segmentation, to help us annotate our images.

The Carvalho et al. [7] program is written in MATLAB, but we translated it to Python and used the OpenCV library to replace all the functions that perform traditional image processing techniques. The original MATLAB program outputs the mean color of each patch used in the color correction algorithm, but for our model we needed the position, not the color. Therefore, we modified the Python code to stop before extracting the colors and use the positions of the patches to create the annotation files. Since the program could not perfectly segment all images, the images where the segmentation was not correct were manually labeled. Figure 4.6 shows an example of the labeled images.



Figure 4.6: Example of image annotation.

To train the model, we split the images and annotations into a training set (70%) and a test set (30%) and converted them to TFRecords.

4.2.2 Model

The model used was the SSD Mobilenet V2, described in chapter 2. This model was already built using the TensorFlow API, so we only need to change the input in the configuration file to make it work. So as a first experience, we only changed the input to use our dataset. You can see the result in the figure 4.7

Unfortunately, as we can see, the model had difficulty detecting only the 24 colors and detected the same color several times in different positions. So, we decide to experiment more with the configuration file.

Since the model is already created, only some changes were made to the configuration file:

- Limit the number of classes to detect to 24 - We only had 24 different patches and these patches show up every time, so we changed the model to always detect all 24 patches.
- Limit the model to only one detection per class - The model detected the same colors more than once, so we limit the model to only detect one patch for each color.



Figure 4.7: Result of the first experience to detect the color checker.

- Due to the memory available for training the model, we had to reduce the batch size to 32;
- And finally, we change the type of the fine - tuning checkpoint from classification to detection.

With these changes, the new model began to detect all 24 patches in the correct location without any of them appearing in an incorrect location. After the model completed training, we converted it to a TFLite model. It was necessary to add two post-processing operators, the TensorFlow Lite operators and the TensorFlow operator. When converting the model to the TFLite model, we did not use optimizers because the model would not run on a IOS smartphone.

4.2.3 Results

In the model TST, three operations must be performed to segment the patches:

- Detection and segmentation of the color checker;
- Detection and segmentation of each patch of the color checker;
- Output of an Excel file with the coordinates for each patch;

The MLM performs the patches detection in one step and its output does not consist of an Excel file, but of three components:

- Bounding boxes - the position of the detected patches;
- The class of the bounding box - the name of the detected color;
- The score of the this prediction - how sure is the model that this bounding box belongs to this class.

The three components can be seen in figure 4.8. This MLM model only performs the detection of the patches and it is the smartphone application that receives these components and performs the segmentation of the patches.

Since we restricted the MLM model to detect only one box for each of the 24 classes, we sometimes have prediction values of less than 30% confidence, as in figure 4.8b. If the confidence value is greater than 20%, we do not consider this a problem. Even if the prediction is lower, the model will detect the correct patch corresponding to this class. If the prediction is so low that the model detects the patch in the wrong place, as in figure 4.8c, the user can use the mobile application to check that the model has made a mistake and perform a manual segmentation.

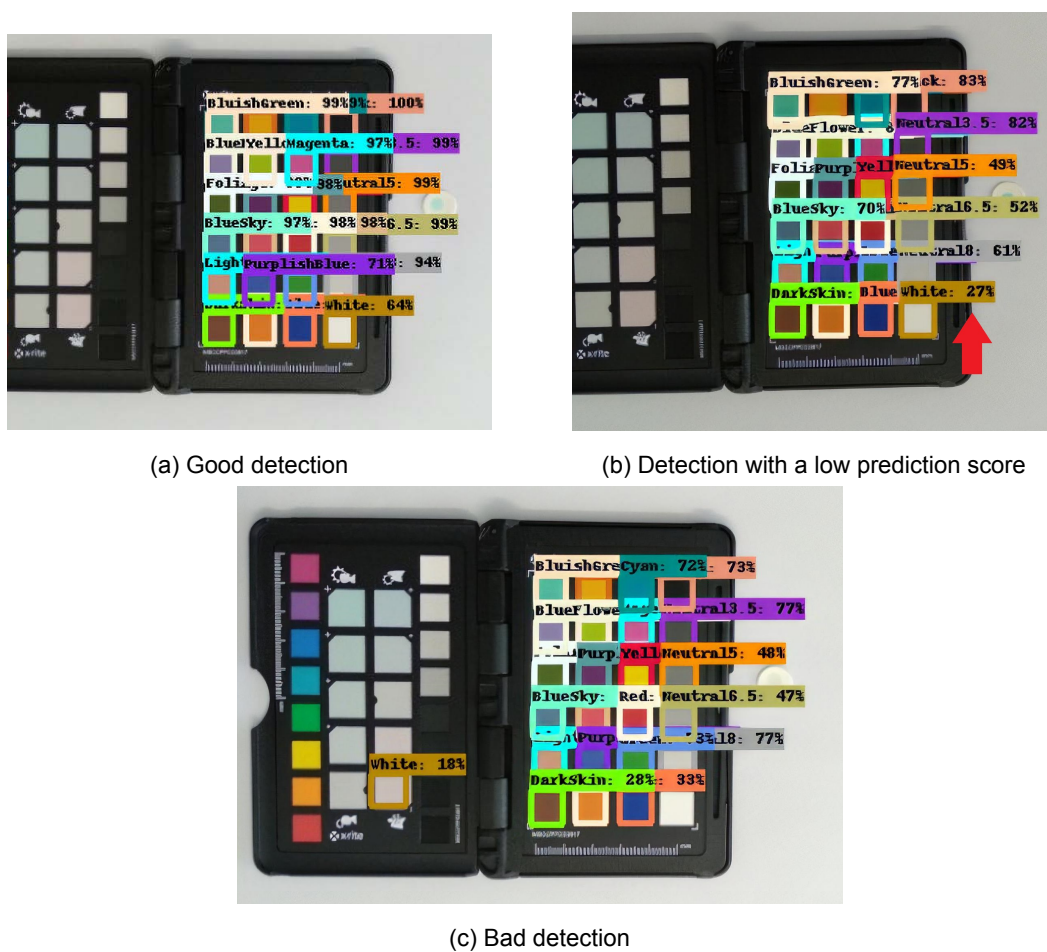


Figure 4.8: Examples of detection results.

To test which model was the best, we took 930 images and ran the two models to see how many images were poorly detected. The results are shown in table 4.1.

As illustrated in table 4.1, the method that performed with fewer errors was the MLM. When running this model on a smartphone, will allow the user to perform an automatic segmentation and only needing to intervene on 0.5% of the images.

Models	Percentage of wrong detection
TST	12%
MLM	0.5%

Table 4.1: Result of patches detection TST vs MLM.

4.3 Disk and Color Detection

This model works similarly to the patches detection model. Given an image, we want to detect the position of the disk and, if possible, the position of the color within the disk (figure 4.9).

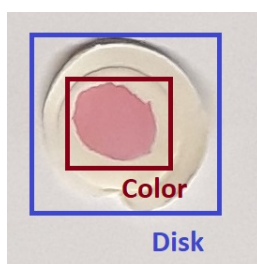


Figure 4.9: Disk and color close up.

4.3.1 Building training dataset

This dataset was created in a similar manner to the patches detection dataset. Using the same images taken by the S-Mode project partners, they were resized and annotated using the modified translated Python code (which stops at disk and color detection) from the MATLAB disk and color segmentation program. Figure 4.10 shows an example of the annotated images.

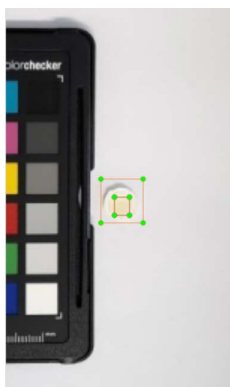


Figure 4.10: Example of train disk image annotation.

This annotation process was more difficult than the one for the patches. As can be seen in figure 4.11, we have a color gradient that gets more pink the higher the concentration of sulfonamides is and at a sulfonamides concentration of 0 $\mu\text{g/L}$ it is sometimes difficult to dis-

tinguish what is the disk and what is the color of the sample being tested. The Python code detected and annotated each image and a manual annotation was done especially for this lower concentration.

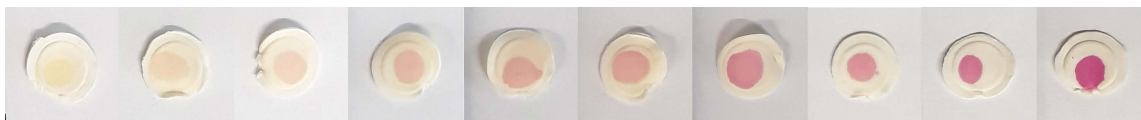


Figure 4.11: Examples of samples of varying sulfonamides concentrations, from 0 $\mu\text{g/L}$ (left) to 150 $\mu\text{g/L}$ (right).

The images and their annotations were split into a training set (70%) and a test set (30%) and converted to TFRecords that were used to train the model.

4.3.2 Model

This detection model was not as easy to build as the patch detection model. Several experiments were conducted, which eventually led to the choice of a different TensorFlow model for the detection of the patches.

The first experiment was conducted using the same training images in the format used for the patch detection (figure 4.12a) and the same SSD Mobilenet V2 model template. This model had the same changes to the configuration file that we had discovered in the patches detection model, except for the option to detect the 24 classes, which was changed to detect only 2: Disk and Color. After training, this model could not detect the disk or the color in any of the images (figure 4.12b).

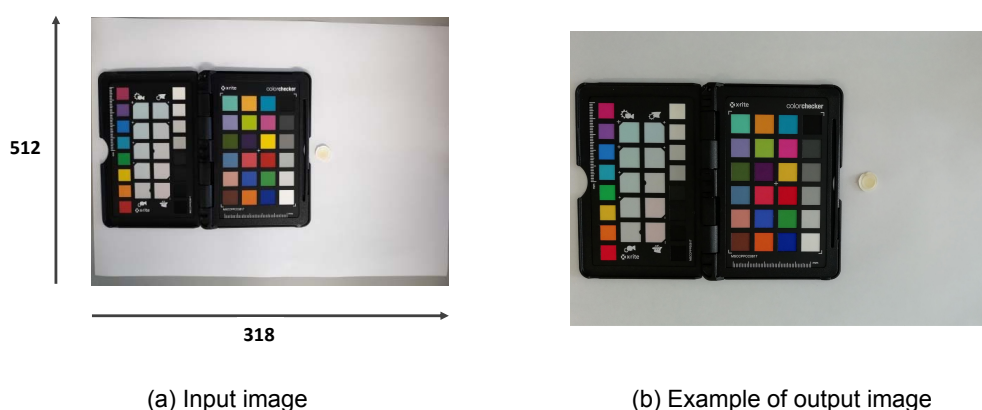


Figure 4.12: Input and output of the first experiment of the detection model.

We did not know if the model had problems detecting the disk and color because the color checker next to it, so for the second experiment, we split the training image in half and selected the right part of the image (figure 4.13a). We can split the image in half because the color

checker is always on the left side of the target, and the color checker is large enough to cover the left part of the image. Thus, we know that the right part of the image contains part of the color checker and the disk. This model configuration was the same as in experiment 1 and again the model could not detect the disk or the color (figure 4.13b).

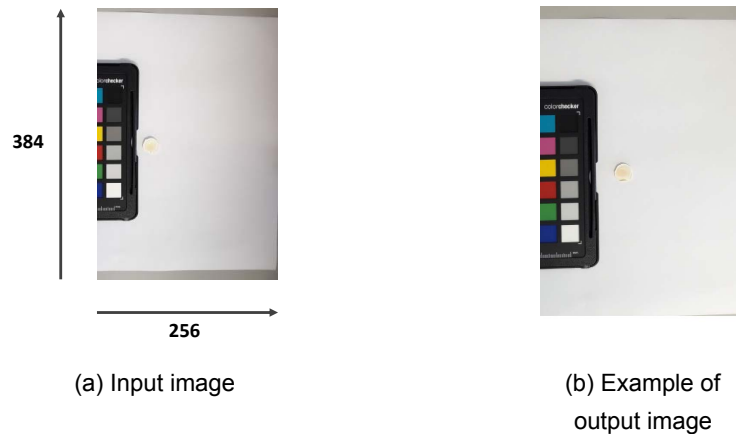


Figure 4.13: Input and output of the second experiment of the detection model.

In the third experiment, we try to find out if the failed trials are related to the objects in the image being too small to be detected by the model. To test this, we split the original image taken by the S-Mode in two and reduced it to a height of 576 and a width of 384 (figure 4.14a). The result was an image that was almost as large as the images used in the first experiment, but now the disk and color were larger. This model configuration was also the same as the past experiments. With this model, there was a small percentage of images where the model could detect the disk (figure 4.14b), others were the disk was detected in the wrong place (figure 4.14c) and there were even fewer images where the color was detected. We trained the model with a higher number of steps, but the accuracy did not improve.

For the fourth and final experiment, we decided to use a new model SSD MobileNet FPNLite with the half images used in the second experiment (figure 2.13b). This model is also part of the TensorFlow API, so we can modify its configuration file and training in the same way as the SSD Mobilenet V2. We decided to experiment with this model because it uses a feature pyramid network, which increases accuracy for small objects [38]. This feature extractor accepts an image of arbitrary size as input and produces appropriately sized feature maps. Using this model and the same configuration file as the previous models, we were finally able to detect both the disk and the color inside these disks.

One drawback of this model is that it is 2 seconds slower than the SSD MobileNet V2, but at this point we were willing to sacrifice a little time for accuracy because although the model is slower when it is running, it is faster than a user performing a manual segmentation when a model performs a bad detection,

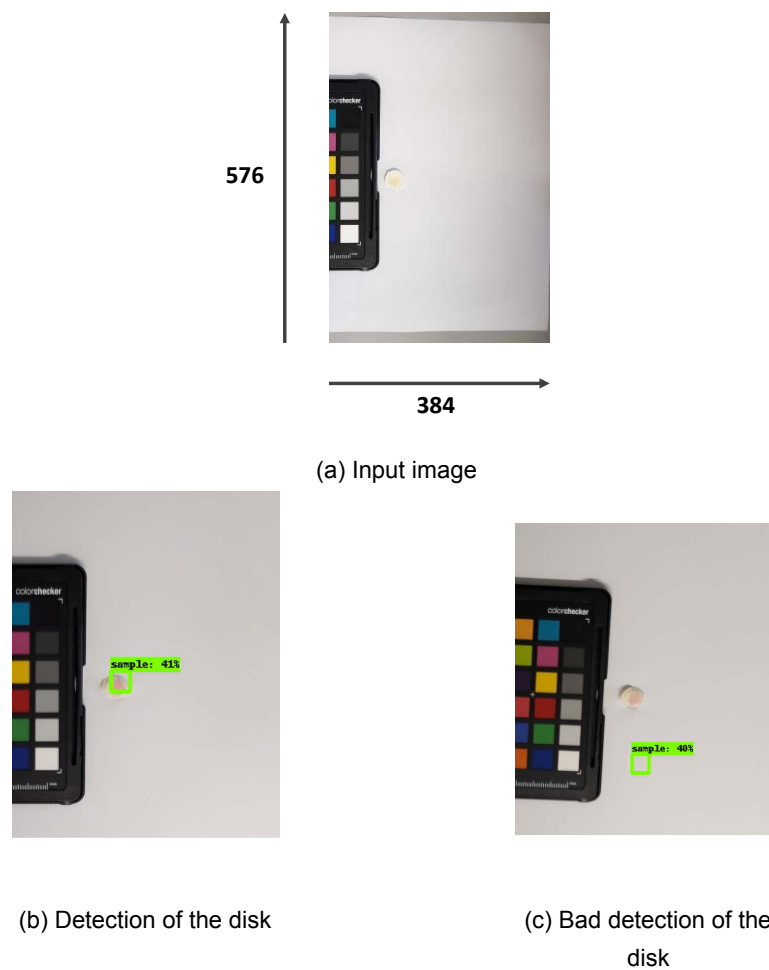


Figure 4.14: Input and output of the third experiment of the detection model.

The final disk detection model was trained with the following changes to its configuration:

- Limit the number of classes to detect to 2 - Detect the disk and its color;
- Limit the model to only one detection per class;
- Due to the memory available for training the model, we had to reduce the batch size to 32;
- Change the type of the fine tuning checkpoint from classification to detection;
- Adding a data augmentation option - Due to the issues we were having with the models, we decided to add a data augmentation option that flips the image vertically to help the model learn to detect the disk and color.

After completing the training, we use the same operators as for the patch detection model to convert it into a TFLite model.

4.3.3 Results

The two disk and color detection models TST and MLM for disk and color detection work similarly to patches detection counterpart, and their outputs are the same as the previous models. The SSD MobileNet FPNLite model is a different model than the one used in the patches detection, but since we used the same TensorFlow API, the input, training and their output type are identical for both models. The result of the detection can be seen in figure 4.15.



Figure 4.15: Example of disk and color detection results.

Similar to the results of the patches detection models, we also tested the two disk and color detection models with the same 930 images. Table 4.2 shows the results of these tests.

Models	Percentage of wrong detection	
	Disk	Color
TST	12.7%	15.5%
MLM	0.0%	0.86%

Table 4.2: Result detection disk and color - TST vs MLM.

It is important to note that the goal of this model was not to have a model that could detect the color of the disk 100% of the time. Rather, it was more important to first detect the disk and then see if we could detect the color. Fortunately, our model performs very well in detecting the disk and the color, as it detects the disk in all images and failed to detect the color only 0.86% of the time.

4.4 Color correction

The objective of this model is to receive an image filled with the colors from the color checker patches and the color of the sample, and then correct it to get the correct colors as shown in figure 4.16.

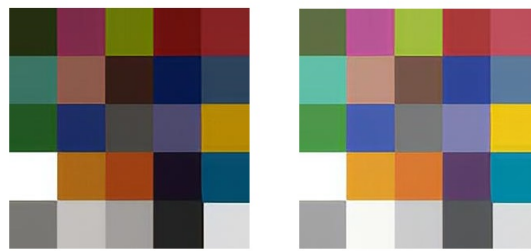


Figure 4.16: Colors taken from a photo (left) vs ground truth colors (right).

4.4.1 Building training dataset

To create this dataset, we used the result data from the color patch detection model. Based on the position of the color patches, we calculate the position of 5 points within the patch. These points are top left, top right, bottom left, bottom right, and center, as shown in figure 4.17. Five points were chosen instead of just the center point to add to the data and cover the possibility that the colors were not captured evenly. Once these points are calculated, we calculate the mean of 25 pixels around that point, as shown in Figure 4.18.



Figure 4.17: Example image of where the color is extracted from the color checker.

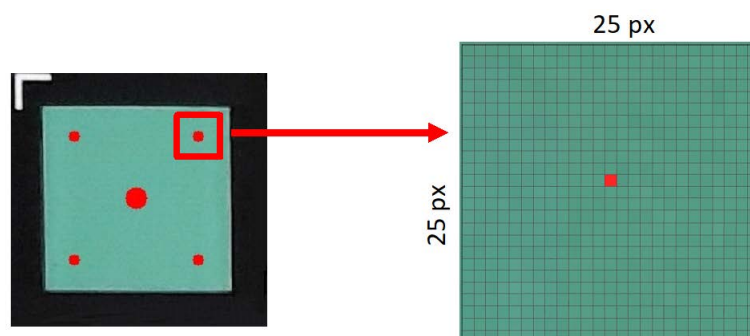


Figure 4.18: Average of a color point.

After collecting all the colors from the 24 patches, we combine them into 5 images. Each

image contains an arrangement of colors from the position where they were retrieved. That is, no colors are repeated in any of the images created and all images will have all 24 colors. Each color occupies an area of 20x20 pixels and the position of the colors within the created images is chosen randomly. An example of images created from a single photo of the color checker is shown in figure 4.19.

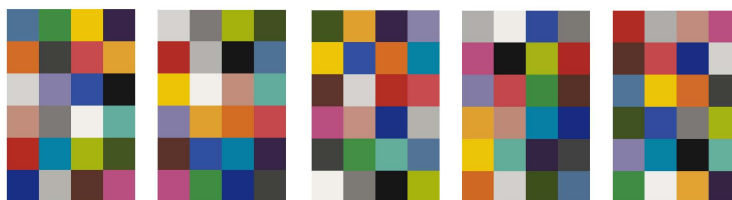


Figure 4.19: Training images created by a single photo of the color checker.

In the first iteration of the dataset, the training images had the format of 6 rows and 4 columns, as shown in the figure 4.19. Since we have 24 colors in the color checker, all 24 colors in the image appeared in the training images, but when we wanted to correct the color of the disk, one of those colors had to be excluded to make room for the color of the disk.

Since we wanted to give the model the best change in color correction, we decided to change the format of the training image so that we could add the disk color without sacrificing any of the other colors. This new format consisted of the same 20x20 pixels for each color, but now we have 5 rows and 5 columns. In the training images, we only have 24 colors to fill the image. Therefore, the area where the color of the disk would be located is filled with a white square in the training phase and the corresponding disk color in the testing phase. The new training images are shown in figure 4.20.



Figure 4.20: Colors taken from the original photo (top), the true value the colors should had (bottom).

4.4.2 Model

We could not build this model in the same way as the previous one because there is no color correction model in the TensorFlow repository. So, we had to create our own model from scratch.

To create a deep learning model, we need to select several objects such as the shape of input and output, the number of layers, the type of layers used (convolution, pooling, dense, etc.) and the type of arguments used (filters, kernel size, padding, activation, etc.) and finally we need to select an optimizer and a loss function to compile and run our model. To find out which model is best, we experimented with several models, all of which had the same following characteristics:

- **Input** - RGB image (the square images created in the dataset creation phase) with a size of 100 px width and 100 px height and with 3 color channels;
- **Optimizer** - The Adam optimization [39] was used. This optimizer is an extension of stochastic gradient descent and is widely used in deep learning due to the good and fast results that can be obtained with it;
- **Sequential** - Models are built in a sequential mode where each layer has only one input and one output;
- **Padding** - All convolutional layers are padded with zeros around the input so that the output has the same dimension as the input.

The architecture of the models can be seen in table 4.3. Each smaller model has 6 variants in which we experiment with three activation layers and two loss functions. The three activation layers are: Rectified Linear Unit activation function (Relu), Sigmoid activation function and the Softmax activation function. The two loss functions are Mean Square Error (MSE) and Mean Absolute Error (MAE). The larger models have 4 variants with two activation layers and the same two loss functions used in the smaller models. The two activation layers used are the Relu and Softmax activations.

Models	Layers								
	2D Convolution					Dense			
Model 1	32	32	32	32	32	-	-	-	
Model 2	64	64	64	64	64	-	-	-	
Model 3	92	92	92	92	92	-	-	-	
Model 4	32	64	256	196	196	1024	128	3	
Model 5	64	128	512	392	392	1024	128	3	

Table 4.3: Representation of the layers for each model. All layers have a kernel size of 3x3.

Model 1, 2, and 3 were based on Lu et al. [29] models, and we decided to test with 3 different sized filters to see if the number of filters affected the accuracy of the model. Model 4 and 5 were based on Lou et al. [32]. We did not have the computational power to train our models

with the filters used in their model, so we reduced them to two sizes. The description of how the functions are distributed can be found in table 4.4.

Models	Activation Function								Error Function
	I1	I2	I3	I4	I5	I6	I7	I8	
Model [1-3].1			Relu			-	-	-	MSE
Model [1-3].2									MAE
Model [1-3].3			Softmax			-	-	-	MSE
Model [1-3].4									MAE
Model [1-3].5			Sigmoid			-	-	-	MSE
Model [1-3].6									MAE
Model [4-5].1					Relu				MSE
Model [4-5].2									MAE
Model [4-5].3			Relu				Softmax		MSE
Model [4-5].4									MAE

Table 4.4: Representation of the activation and error function use in our models.

4.4.3 Results

Before we look at the results of the color correction models, we would like to point out that there were some models that had problems during training, resulting in models where all the lighter colors were corrected to the same color (figure 4.21b) or the image was corrected to all the same color (figure 4.21c). These models were the ones where softmax was enabled in all layers. They are not considered here in the discussion of the results.

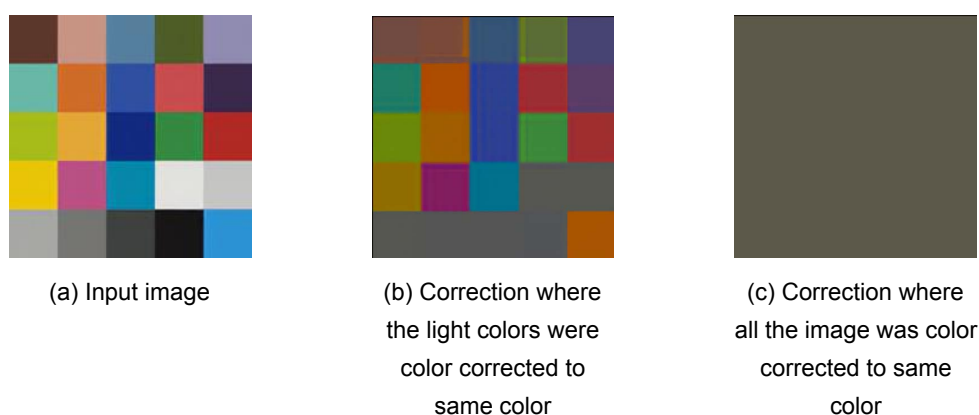


Figure 4.21: Example images of the color correction models that had troubles learning.

To avoid overfitting, the training of models was stopped if they did not improve during 20 epochs. There were 16000 images used for training and the batch size was 32. In figure 4.22 we see some of the color corrected images from the models described in the above subsection.

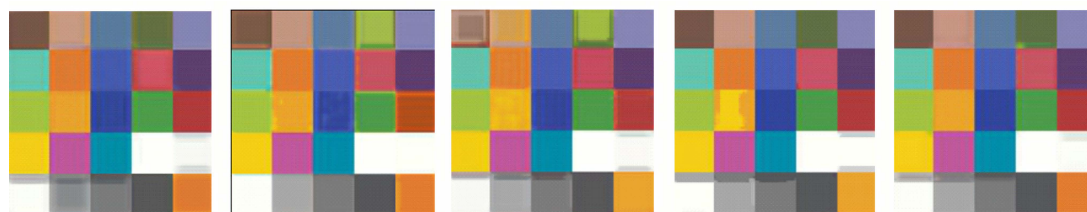


Figure 4.22: Example of color corrected images. From left to right model 2.1, model 1.4, model 4.4, model 3.2 and model 4.5.

To compare the machine learning models with the baseline results in chapter 3, we calculated the standard deviation of the 8 colors used to test the color correction. These results can be found in the table 4.5, where we can see the standard deviation of all models. The color components that obtained the best results were: G from the RGB color space, Z from the XYZ color space, H and S from the HSV color space, and finally a^* and b^* from the CIELab color space. These were the components we chose to present the results. And as we can see, the best model is the model 5.4 which had 5 layers with a relu activation function, 3 layers with the softmax activation and a MAE error function. This model has the lowest standard deviation for the color components H, S and a^* .

In the table 4.6 we can see the comparison between model 5.4 (MLM) and the model we develop in chapter 3 (TCM). In the previous work and in this model, the color components a^* and H have the lower standard deviation, so we will only compare the models with these color components. The machine learning model could achieve lower standard deviation for both H component and a^* .

4.5 Calculation sulfonamides concentration

The aim of this model is to receive a color of the disk represented in 4 color spaces (RGB, XYZ, HSV and CIELab) and predict its sulfonamide concentration.

4.5.1 Building training dataset

This model differs from the others in that the model does not receive an image. An array is received. This training dataset is created using the results of the color correction model. All the images that have passed through the models are divided into a training set (70%) and a test set (30%). For each set, we created a csv file containing the concentration of the sample and the color corrected color of the disk represented in 4 color spaces: CIELab, RGB, HSV and XYZ (figure 4.23). With the csv file created, we can easily convert it into an array and pass it to the model

Model	G	Z	H	S	a*	b*
Model 1.1	2.1497	2.0757	0.5473	1.8480	0.4802	1.0700
Model 1.2	2.8118	2.8815	2.5995	2.6460	0.8119	1.4581
Model 1.5	2.4033	1.7207	0.5664	1.9299	0.3270	0.9388
Model 1.6	2.6246	2.5797	1.1405	2.0511	0.4132	1.2231
Model 2.1	2.6796	2.5860	0.8420	1.8355	1.0328	1.1412
Model 2.2	1.6824	1.1741	1.5910	3.0009	0.8244	1.9689
Model 2.5	2.0249	2.0405	0.6804	2.3352	0.1435	1.5046
Model 2.6	2.1255	2.2037	1.9798	2.1895	0.1671	1.4284
Model 3.1	1.4871	1.0947	7.5292	2.7160	0.4932	1.1856
Model 3.2	2.0287	1.6808	0.7427	2.3291	0.9389	1.2867
Model 3.5	2.7216	3.4469	0.0981	2.1541	0.2914	1.4219
Model 3.6	1.8577	1.4204	2.2497	3.3458	0.4355	1.4179
Model 4.1	1.6777	1.1974	0.8322	1.9214	0.9100	1.5009
Model 4.2	1.0503	0.5331	0.7815	2.1650	0.8590	1.4768
Model 4.3	1.4471	1.5678	1.7126	2.3819	1.9107	1.7216
Model 4.4	1.4223	0.9488	1.5257	2.4584	1.4453	1.6503
Model 5.1	1.6784	0.9657	0.1123	1.2657	0.5678	1.6347
Model 5.2	1.4158	0.4782	0.0898	0.7205	0.0245	1.7893
Model 5.3	1.2790	0.6355	0.0686	0.8534	0.2680	1.5678
Model 5.4	1.9657	0.5236	0.0457	0.6895	0.1985	1.8324

Table 4.5: Results of the color correction models.

Models	Standard Deviation	
	H	a*
TCM	0.047	0.284
MLM	0.046	0.199

Table 4.6: Result of color correction - TCM vs MLM.

4.5.2 Model

Carvalho et al. [40] used linear and polynomial regression to find the best color component curve that can be used to calculate the sulfonamides concentration. Instead of using only one component, we wanted to find out which components can be combined to get better accuracy.

To know which other components could be used, we tested the correlation between the components. The result is shown in figure 4.24. The color components that better correlate with the sulfonamides concentration values are (in order of importance): a*, S, V, R, X. This

Concentration	R	G	B	X	Y	Z	H	S	V	L	a*	b*
5	86,34483	86,40953	84,37064	0,08767	0,092798	0,097169	0,678906	0,02454	0,339112	36,67543	-0,45777	1,178232
10	89,4202	85,05162	82,65319	0,08882	0,09188	0,093469	0,558145	0,076238	0,350667	36,50074	1,284093	2,033871
0	86,06465	86,09554	84,9648	0,087397	0,09226	0,098299	0,670705	0,013679	0,33778	36,57328	-0,25354	0,655162
150	132,1977	30,07557	94,85173	0,119591	0,06619	0,113389	0,401734	0,778926	0,518422	31,05022	48,55933	-13,2658
50	123,4108	76,09387	57,62033	0,114668	0,096392	0,051778	0,547186	0,53754	0,483964	37,34857	17,90598	19,34967
10	87,47261	87,78021	81,87218	0,088855	0,095152	0,092546	0,676466	0,068185	0,344356	37,11833	-1,34813	3,39034

Figure 4.23: Example of data array created.

correlation helps us to get an idea of which group of components we could use to calculate the concentration.

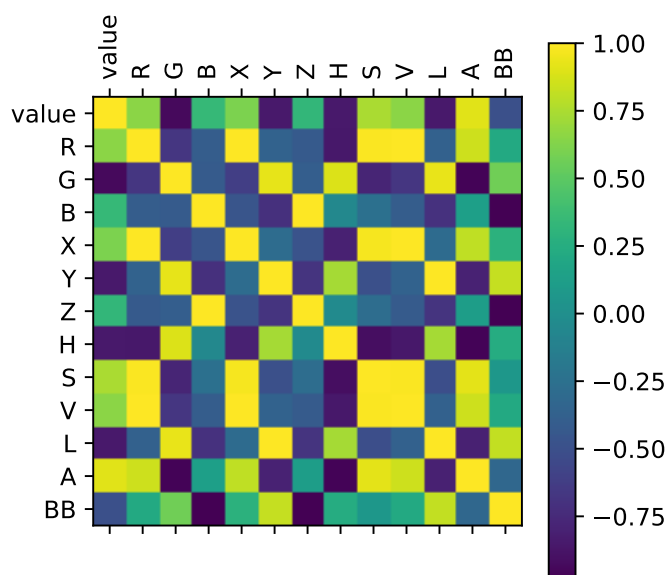


Figure 4.24: Correlation of the color components and concentration value.

The models used were traditional regression models:

- Linear, Quadratic, Cubic and Quartic Regression - These types of regressions are equations that combines a set of input values to a set of predicted values;
- Ridge and Lasso Regression - These models prevent overfitting and reduce the complexity that can be generated by a simple linear regression. In Ridge regression, a penalty equal to the square of the coefficients is added to the cost function. And lasso regression uses shrinkage, where the data values are shrunk to a central point as the mean;
- Linear, Polynomial and Radial Basis Function (RBF) Support Vector Machines (SVM) - These algorithms use a set of N features to find a hyperplane in an N-dimensional space that uniquely classifies the data points;
- Elastic Net - It uses two types of penalties: the L2 penalty, where the model is penalized based on the sum of squared coefficient values, and the L1 penalty, where penalties are applied based on the sum of absolute coefficient values;

- Decision Tree - These are models that use a tree structure to predict the data;
- Forest Tree - This model combines the prediction of multiple decision trees to make a more accurate prediction.

4.5.3 Results

The models we trained with several times with several variables and we chose the best results from each model. To obtain the score of the models, we run each model 1000 times. In each run, we split the data into a training (70%) and a testing (30%) dataset and average all the scores.

In table 4.7 we show the results of testing the models with all the color components and the color components in order of greater correlation.

Models	Score					
	All components	a*	a* S	a* S V	a* S V R	a* S V R X
Linear Regression	-82.4%	78.8%	85.0%	84.2%	84.0%	83.3%
Ridge Regression	84.3%	79.0%	83.1%	83.7%	84.2%	84.3%
Quadratic Regression	-76.9%	85.3%	84.3%	85.9%	86.7%	86.4%
Cubic Regression	-54575.1%	84.4%	85.3%	76.9%	-9.6%	-9.3%
Quartic Regression	-815368.6%	84.3%	45.1%	25.6%	-264.0%	-476.2%
Linear SVM	82.0%	76.1%	76.2%	76.0%	81.5%	82.1%
Polynomial SVM	13.4%	76.9%	77.5%	77.3%	28.3%	18.8%
RBF SVM	-11.5%	0.2%	1.4%	1.6%	-10.8%	-10.6%
Lasso	84.7%	79.0%	84.7%	84.7%	84.4%	84.2%
Elastic Net	84.6%	79.1%	84.6%	84.5%	84.1%	84.8%
Decision Tree	90.1%	71.0%	89.6%	89.7%	89.5%	87.5%
Forest Tree	90.3%	76.8%	90.1%	90.2%	88.8%	90.5%

Table 4.7: Result of the regression models with different type of color components.

As we can see, the regression forest tree scores best with the 5 color components that correlate best with its concentration value. Because of this discovery we decided to focus the forest tree regression more to see if the result can be improved. So, we created a forest tree model with the same training options as before, but with all the color components. Once the model was trained, we extracted the importance of each color component. The results are shown in figure 4.25.

With this data, we repeated the above process to obtain the scores of the models, but we focused only on the decision and forest tree models, since these models are similar and

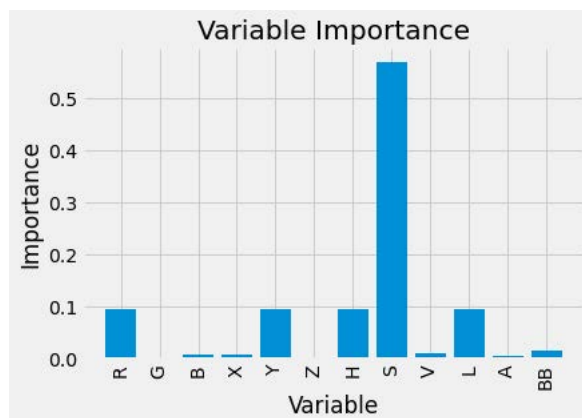


Figure 4.25: Graph of the importance of each color component.

performed well before. The results are shown in the table 4.8.

Models	Score			
	S, Y	S, Y, R	S, Y, R, L	S, Y, R, H
Decision Tree	90.7%	91.0%	91.3%	91.0%
Forest Tree	91.5%	91.1%	91.4%	91.2%

Table 4.8: Result of Decision Tree and Forest Tree with the most important color components.

Looking at the results, we concluded that using a forest tree with color components S, Y provides the best accuracy in predicting the concentration value of sulfonamides.

4.6 Summary

In this chapter, we have described how the machine learning models were created and trained, and we also show how these models produce better results compared to the traditional methods developed so far.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The detection of sulfonamides is an important task for which several methods have been investigated using laboratory equipment. The work of Carvalho et al. [7] introduces a new mobile work where the detection can be performed wherever it is needed. We have shown with this work that their traditional methods have been outperformed by using machine learning models. We had some limitations in the form of the original dataset, but even with that our models were able to produce good results. And since the models are modular, when new models appear, they can be easily replaced and we continue to have a working application.

5.2 Future Work

Since our work was limited by the original dataset, we propose to develop a new dataset in the future that could be developed specifically for machine learning. One of the changes we would make would be to use a color checker that has a larger number of color patches. As Lu et al. [29] suggests in their paper, color correction works better when we have a larger number of colors, especially when we do color correction with machine learning because the more data it has, the better it learns. Another suggestion for future work is to test the models with contaminated water, especially the color correction model, to see if the models can adapt to water that is not uniformly crystalline.

Bibliography

- [1] A. Nardone. Evolution of livestock production and quality of animal products. 2002.
- [2] Karel Hruska, Milan Fránek, et al. Sulfonamides in the environment: a review and a case report. *Vet Med*, 57(1):1–35, 2012.
- [3] Wojciech Baran, Ewa Adamek, Justyna Ziemiańska, and Andrzej Sobczak. Effects of the presence of sulfonamides in the environment and their influence on human health. *Journal of hazardous materials*, 196:1–15, 09 2011. doi:10.1016/j.jhazmat.2011.08.082.
- [4] C Lee Ventola. The antibiotic resistance crisis: part 1: causes and threats. *PubMed*, 40: 277–83, 2015.
- [5] European Centre for Disease Prevention and Control. Antimicrobial resistance in the eu/eea (ears-net) - annual epidemiological report 2019, 2019.
- [6] U.S Department of Health and Human Services. Antibiotic resistance threats in the united states, 2019.
- [7] Pedro Carvalho, Sílvia Bessa, Ana Silva, Patricia Peixoto, Marcela Segundo, and Helder Oliveira. *Estimation of Sulfonamides Concentration in Water Based on Digital Colourimetry*, pages 355–366. 09 2019. ISBN: 978-3-030-31331-9. doi:10.1007/978-3-030-31332-6_31.
- [8] Malgorzata Gbylik-Sikorska, Andrzej Posyniak, Tomasz Sniegocki, and Jan Zmudzki. Liquid chromatography–tandem mass spectrometry multiclass method for the determination of antibiotics residues in water samples from water supply systems in food-producing animal farms. *Chemosphere*, 119:8–15, 2015.
- [9] Rodrigo Hoff and Tarso BL Kist. Analysis of sulfonamides by capillary electrophoresis. *Journal of separation science*, 32(5-6):854–866, 2009.
- [10] Nina Bilandžić, Božica Solomun Kolanović, Ivana Varenina, Giampiero Scortichini, Loredana Annunziata, Matko Brstilo, and Nevenka Rudan. Veterinary drug residues determination in raw milk in croatia. *Food control*, 22(12):1941–1948, 2011.

- [11] Faten A Nour El-Dien, Gehad G Mohamed, Elmersy Khaled, and Eman YZ Frag. Extractive spectrophotometric determination of sulphonamide drugs in pure and pharmaceutical preparations through ion-pair formation with molybdenum (v) thiocyanate in acidic medium. *Journal of Advanced Research*, 1(3):215–220, 2010.
- [12] Chenglong Li, Xiangshu Luo, Yonghan Li, Huijuan Yang, Xiao Liang, Kai Wen, Yanxin Cao, Chao Li, Weiyu Wang, Weimin Shi, et al. A class-selective immunoassay for sulfonamides residue detection in milk using a superior polyclonal antibody with broad specificity and highly uniform affinity. *Molecules*, 24(3):443, 2019.
- [13] Niall O’ Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Adolfo Velasco-Hernández, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. *CoRR*, abs/1910.13796, 2019.
- [14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [15] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.
- [16] ScienceWithMe! Learn about the human eye.
- [17] Icon Photography School. What is white balance?, 2021. [Online; accessed August 2, 2021].
- [18] Gershon Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin institute*, 310(1):1–26, 1980.
- [19] Joost Van De Weijer, Theo Gevers, and Arjan Gijsenij. Edge-based color constancy. *IEEE Transactions on image processing*, 16(9):2207–2214, 2007.
- [20] Edwin H Land. The retinex theory of color vision. *Scientific american*, 237(6):108–129, 1977.
- [21] R. He, Z. Wang, H. Xiong, and D. D. Feng. Single image dehazing with white balance correction and image decomposition. In *2012 International Conference on Digital Image Computing Techniques and Applications (DICTA)*, pages 1–7, 2012. doi:10.1109/DICTA.2012.6411690.
- [22] Dehuan Zhang Jingchun Zhou and Weishi Zhang. Multiscale fusion method for the enhancement of low-light underwater images. *Mathematical Problems in Engineering*, 2020:1–15, 2020. doi:10.1155/2020/7095248.

- [23] Ching-Chih Weng, H. Chen, and Chiou-Shann Fuh. A novel automatic white balance method for digital still cameras. In *2005 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3801–3804 Vol. 4, 2005. doi:10.1109/ISCAS.2005.1465458.
- [24] Francesca Gasparini and Raimondo Schettini. Color balancing of digital photos using simple image statistics. *Pattern Recognition*, 37(6):1201 – 1217, 2004. ISSN: 0031-3203. doi:https://doi.org/10.1016/j.patcog.2003.12.007.
- [25] C. Li, H. Tsai, C. Yang, M. Lin, K. Huang, and Y. Lin. Quantifying the color changes in bruised skin using a color-calibrated imaging system. In *2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pages 1–5, 2020. doi:10.1109/MeMeA49120.2020.9137217.
- [26] M. You, J. Liu, J. Zhang, M. Xv, and D. He. A novel chicken meat quality evaluation method based on color card localization and color correction. *IEEE Access*, 8:170093–170100, 2020. doi:10.1109/ACCESS.2020.2989439.
- [27] Facundo Bre, Juan M Gimenez, and Víctor D Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158:1429–1441, 2018.
- [28] Diego Ardila, Atilla P. Kiraly, Sujeeth Bharadwaj, Bokyung Choi, Joshua J. Reicher, Lily Peng, Daniel Tse, Mozziyar Etemadi, Wenxing Ye, Greg Corrado, David P. Naidich, and Shravya Shetty. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature Medicine*, 25:954–961, 2019. doi:10.1038/s41591-019-0447-x.
- [29] Yunxi Lu, Xiaoguang Li, Zhaopeng Gong, Li Zhuo, and Hui Zhang. Tdccn: A two-phase deep color correction network for traditional chinese medicine tongue images. *Applied Sciences*, 10(5), 2020. ISSN: 2076-3417. doi:10.3390/app10051784.
- [30] Emmanuel Malherbe Robin Kips, Loïc Tran and Matthieu Perrot. Beyond color correction : Skin color estimation in the wild through deep learning. *Electronic Imaging*, 2020(82-1-82-8), 2020. doi:doi:10.2352/ISSN.2470-1173.2020.5.MAAP-082.
- [31] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.
- [32] Zhongyu Lou, Theo Gevers, Ninghang Hu, Marcel P Lucassen, et al. Color constancy by deep learning. In *BMVC*, pages 76–1, 2015.
- [33] Ali Alsam and Graham Finlayson. Integer programming for optimal reduction of calibration targets. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of*

- Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 33(3):212–220, 2008.
- [34] Graham D Finlayson, Michal Mackiewicz, and Anya Hurlbert. Color correction using root-polynomial regression. *IEEE Transactions on Image Processing*, 24(5):1460–1470, 2015.
- [35] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.
- [36] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [37] TensorFlow. Tensorflow lite converter, 2021. [Online; accessed August 2, 2021].
- [38] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [40] Pedro H. Carvalho, Inês Rocha, Fabio Azevedo, Patricia S. Peixoto, Marcela A. Segundo, and Helder P. Oliveira. *Cost-efficient Color Correction Approach on Uncontrolled Lighting Conditions*. 2021.