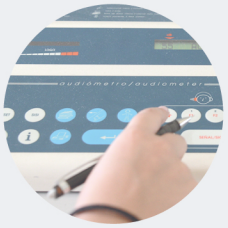
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA BIOMÉDICA






Utilização da câmara smartphone para monitorizar a aderência à terapia inalatória

SOFIA ALEXANDRA GONÇALVES FERRAZ
Outubro de 2021

P.PORTO

Use of the Smartphone Camera to Monitor Adherence to Inhaled Therapy

Sofia Alexandra Gonçalves Ferraz

Biomedical Engineer by Instituto Superior de Engenharia do Porto

"Dissertation for the partial fulfillment of the requirements for the master's degree in Biomedical Engineering by Instituto Superior de Engenharia do Porto"

Supervisors: Dr. Rute Almeida, Dr. Pedro Marques and Dr. Nuno Escudeiro

January 2020

Acknowledgments

Throughout the writing of this dissertation I have received a great deal of support and assistance.

First and foremost, I am extremely grateful to my supervisors, Dr. Rute Almeida and Dr. Pedro Marques for their invaluable advice, continuous support, and patience during my thesis.

In addition, I would like to thank my parents for offering me the opportunity to pursue studies. I hope that this stage that I now finish can, somehow, compensate for all the support, affection and friendship they always unconditionally offered me.

Furthermore, I could not have completed this dissertation without the support of Alexandre Lima, who provided a sympathetic ear as well as words of comfort and motivation.

Finally, this research was part of the project "mINSPIRE—mHealth to measure and improve adherence to medication in chronic obstructive respiratory diseases - generalization and evaluation of gamification, peer support and advanced image processing technologies", POCI-01-0145-FEDER-029130, supported by the ERDF (European Regional Development Fund), COMPETE2020 (Programa Operacional Competitividade e Internacionalização), Portugal 2020 and by Portuguese Funds through FCT (Fundação para a Ciência e a Tecnologia).

Abstract

Self-management strategies can lead to improved health outcomes, fewer unscheduled treatments, and improved disease control. Compliance with inhaled control drugs is essential to achieve good clinical outcomes in patients with chronic respiratory diseases. However, compliance assessments suffer from the difficulty of achieving a high degree of trustworthiness, as patients often self-report high compliance rates and are considered unreliable. This thesis aims to enable reliable adhesion measurement by developing a mobile application module to objectively verify inhalation usage using image snapshots of the inhalation counter.

To achieve this, a mobile application module featuring pre and post processing techniques and a default machine learning framework was built, for inhaler and dosage counter numbers detection. In addition, in an effort to improve the app's capabilities of text recognition on a worst-performing inhaler, a machine learning model was trained on an inhaler image dataset. Some of the features worked on during this project were incorporated on the current version of the app InspirerMundi, a medication management mobile application, planned to be made available at the PlayStore by the end of 2021.

The proposed approach was validated through a series of different inhaler image datasets. The carried-out tests with the default machine learning configuration showed correct detection of dosage counters for 70% of inhaler registration events and 93% for three commonly used inhalers in Portugal. On the other hand, the trained model had an average accuracy of 88 % in recognizing the digits on the dose counter of one of the worst-performing inhaler models.

These results show the potential to explore mobile and embedded capabilities to gain additional evidence for inhaler compliance. These systems can help bridge the gap between patients and healthcare professionals. By empowering patients with disease self-management and drug adherence tools and providing additional relevant data, these systems pave the way for informed disease management decisions.

Key words: optical character recognition; medication adherence; mHealth; remote monitoring.

Index

List of Figures

List of Tables

CHAPTER 1 – INTRODUCTION

1. Introduction

1.1. Context, problem and motivation

Asthma exacerbations can be reduced with appropriate regular therapy and patient education. Despite this, asthma affects about 300 million people globally and accounts for 1 in every 250 deaths. In Europe alone, approximately 30 million people have asthma and 15,000 people die yearly from this disease [1]. The economic burden of asthma is substantial, in 2010, adult asthma accounted for over 2 % of the total healthcare expenditure in Portugal (3% if children and adults are included). On average, each adult cost 708.16€ a year, with direct costs representing 93%. Furthermore, uncontrolled patients' costs are more than double than those of controlled asthma patients. Thus, improving asthma control in patients is critical to diminish this burden [2].

Treatment adherence is generally low among patients with asthma. As a matter of fact, some studies show that adherence is less than 50% in children and as low as 30% in adults [3]. This low adherence may be due in part to misinformation or confusion regarding complicated treatment regimens. Additional barriers such as high prescription cost, taste of medication, and uncertainty about the safety of inhaled corticosteroids may contribute to poor adherence to inhaled asthma medications.

Poor medication adherence is concerning, since it is shown to increase risk of asthma exacerbations, leading to higher mortality, greater financial burden for the patient and health system, as well as decreased quality of life [3]. Numerous adherence-improvement interventions have been introduced, but most have been only moderately successful with little evidence of long-term sustainability or reduction of health care utilization and cost [4].

Mobile Health (mHealth) technologies can improve disease outcomes and may be an especially powerful tool to deliver effective behavioral health interventions that are dynamic, user-centric, and continuously adapted [5]. Medication-use monitoring can provide important information for patients, researchers, and health professionals, with the aim of facilitating improved adherence and of improving treatment prescribing, but available monitoring methods vary in quality. Patient self-report and clinician assessments of medication adherence are notoriously unreliable [6].

Although subjective, self-reports are still considered one of the preferred methods to continuously monitor adherence as they are simple, cheap and minimally intrusive. One self-report measure of medication adherence is Visual Analogue Scale (VAS)[1]. Nevertheless, reliance on VAS also has its limitations: patients tend to overestimate their level of adherence and physicians have been found to be inaccurate in estimating patients' adherence when using VAS [7].

Regarding inhaled medication, current mHealth applications require the user to manually enter the readings from the dose counters of these medical devices. This process is slow and prone to error. As the internet becomes more embedded into medical monitors through Wi-Fi and Bluetooth technologies, more sophisticated systems transmit the values from the connected devices to the smartphone. However, this adds costs to the manufacturing of the device and brings connectivity issues. Moreover, requiring a reading to be transmitted over Bluetooth is not applicable to devices that are not Bluetooth-compatible [8]. People who cannot afford to upgrade to these expensive devices will fail to receive the benefits [9].

In the United States, smartphones are owned and regularly carried by approximately 50% of 12–17 year-olds and 75% of adults ages 30–49 [5]. The advantages of smartphones over other devices is not only the fact that they are affordable, but also that they are very powerful, with most models nowadays integrating several cores in their main processor. They are also standalone devices with a camera, a battery, and audio output and an Internet connection [10]. Therefore, these devices show high potential to be explored as a relevant mHealth tool.

1.2. InspirerMundi App

The InspirerMundi app aims to provide a verified monitoring of treatment adherence through gamification and social interaction and is currently available in stores free of charge. The focus of the app is to support patient's medication management, while transforming the process of adherence to treatment into a positive experience. InspirerMundi's was developed through a highly iterative process incorporating input/feedback from patients and physicians throughout [11].

[1] The VAS for medication adherence was developed as an adjunct self-report measure of medication adherence. The VAS asks individuals to mark a line at the point along a continuum showing how much of each drug they have taken in the past month [60].

The app stores the prescribed therapeutic plan (current medications: name, dose, medication bar code, treatment duration, treatment dosage and schedule) inserted by the patient and uses it to trigger related reminders and the record of performed inhalations. Other reminders present are related to symptoms and burden questionnaires and Control of Allergic Rhinitis and Asthma Test CARAT. The scheduling of events is derived from the therapeutic plan taking in consideration the periodicity prescribed for intakes and from predetermined (symptoms and burden questionnaires and Control of Allergic Rhinitis and Asthma Test CARAT) which have a fixed periodicity. The therapeutic plan includes the registration of the specific medications name, posology, frequency, and duration of the treatment. The app includes, as the main interaction interface, a timeline where expected events of monitoring and medication intake are depicted providing a quick reference when a medication is due [11].

Inspirer Mundi, by combining features of inhaler usage detection and a gamification approach based on peer support, delivers an innovative way to measure and improve the adherence to inhaler [10]. Versions 1.x of the app were made available on the App store and Playstore, and were used in feasibility studies of the InspirerMundi app to monitor medication adherence in adolescents and adults with persistent asthma (treated with daily inhaled medication) [12].

1.2.1. Inhaler Usage Detection

According to the scheduled events, users are prompted to register their medication intake. These can be of three kinds: inhalation (referred throughout as inhalers), pills and others (such as a nasal spray or an oral solution). When the event is for an inhaler, the real-time inhaler medication module is triggered, and the user is requested to collect images from the inhaler and register the value of the inhaler dosage counter after the use.

The module uses the smartphone camera; and provides, through image processing techniques and machine learning tools, confirmation of the inhaler presented to the camera. A more recent version developed only for iOS is also able to access dosage values inferred from the acquired dose counter image [10][13]. The tool can thus be used as a pervasive low-cost means of collecting data on patients' adherence to inhalers. This functionality is triggered from the timeline when a new inhalation is registered.

The developed detection tool is based on computer vision methods and key visual features which are common on dose trackable devices: i) a contour or outer shape; ii) a specific written label/canister; and iii) a dose counter indicating the remaining doses. The

detection of each dose in the dose counter uses numerical dose OCR (optical character recognition) standard techniques together with character positioning in addition to object (foreground) and dial (background) color, when available, for partial number correction. The detection tool is at a proof-of-concept stage, currently working with virtually all models of inhalers on the European market with a numerical dosimeter that corresponds to the number of doses available in the device.

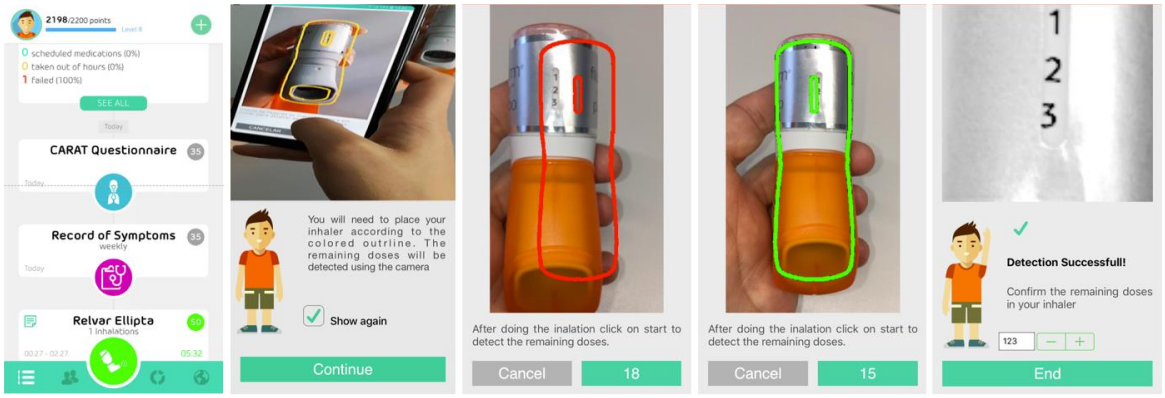


Figure 1 - Workflow for dosage detection counter.

1.3. Aim and Main Objectives

The final goal of the presented work is to improve an inhaler detection module for the Android version of the Inspirer's Mundi app. To this end, an isolated module was built in Android Studio, with text recognition features, in order to detect the digits on the dose counter of commercially available inhalers.

The main objectives of this thesis are:

1. Build an Android module that allows for the text recognition on the inhaler’s dosage counters.
2. Applying pre-processing and post-processing techniques to improve the algorithm performance.
3. Enhance the inhaler detection module of the Android version of the Inspirer's Mundi app.
4. Develop a machine learning algorithm to optimize the efficiency of the application OCR algorithm.

1.4. Document Structure

This thesis is divided into six chapters. Chapter 1 offers a brief contextualization as well as the motivation for the problem we try to solve. Moreover, this chapter includes an overview of the app where the proposed solution will be incorporated. The second chapter examines the literature on OCR and presents possible approaches for text recognition in mobile apps, as well as challenges that may be posed during development. Case studies are also analyzed in Chapter 2. The methodology is outlined in the third chapter. The next chapter addresses the obtained results. Chapter 5 discusses the limitations of the current work. The conclusions are drawn in the final chapter.

CHAPTER 2 – LITERATURE REVIEW

2. Literature Review

In order to collect objective data on patients' adherence to treatment, an inhaler usage detection tool based on imaging processing technologies is proposed to measure and improve the adherence to inhalers. This feature allows to recognize the remaining doses on the inhaler's meter, and thus monitor medication adherence. The problem of recognizing digits or characters in real-world applications is encompassed in the study of optical character recognition (OCR).

There are several aspects to consider in conducting a literature review on the topic of OCR. The first is to understand the current technology as well as the existing limitations. The second focus on the OCR engines available to the development of this method in smartphones. Next, the third consideration consists of various types of preprocessing steps that can be added, and which can increase the accuracy of our system. Lastly, there have been several publications on this subject whereby an overview of similar work to the current case will be presented.

2.1. Optical Character Recognition

Optical character recognition (OCR) is a powerful tool for bringing information from our analog lives into the increasingly digital world [14]. OCR belongs to the family of machine recognition techniques performing automatic identification. Automatic identification is the process where the recognition system identifies objects automatically, collects data about them and enters data directly into computer systems i.e. without human involvement [15].

In short, OCR process converts scanned images of typewritten or hand-written text into machine-readable text [16]. OCR has evolved and became more and more mature with the advancement of technologies and contributions of well-known companies such as IBM, HP, Microsoft, Google and etc. through ongoing researches [17]. Although many commercial systems for performing OCR exist for a wide variety of applications, the available machines are still not able to compete with human reading capabilities with desired accuracy levels [15].

The whole process of an OCR algorithm includes several stages as shown in Figure 3. These stages are as follows:

- **Image Acquisition:** This step consists in capturing the image from an external source (e.g. a smartphone's camera).
- **Location Segmentation:** Segmentation is a process that determines the constituents of an image, locating the regions of the document where characters have been printed and distinguish them from figures and graphics.
- **Preprocessing:** Once the image has been acquired, different preprocessing steps can be performed to improve the quality of image. Since processing color images is computationally more expensive, most of the applications in character recognition systems utilize binary or grey images, thus, conversion of color images is performed in this step [18].
- **Character segmentation:** The characters in the image are separated so that they can be recognized. Character segmentation can be categorized into three strategies: top-down, bottom-up and hybrid. The top-down approach (e.g. projection profile, filtering techniques, Hough transform) takes as input the entire image of text and attempts to divide it into different text-lines images. In contrast, the bottom-up strategies start by searching for interest pixels and then groups interest pixel level. They then manage those interest pixels into connected components that constitute characters which are then combined into words, and lines or text blocks. The integration of both top-down and bottom-up methods is called hybrid approaches [18] [19].
- **Feature extraction:** The segmented characters are then processed to extract different features. Based on these features, the characters are recognized. Extraction of representative and essential features from an input image is the main key to improving the performance of a recognition system.
- **Character classification:** This step maps the features of segmented image to different categories or classes. There are different types of character classification techniques. Structural classification techniques are based on features extracted from the structure of the image and use different decision rules to classify characters. Statistical pattern classification methods are based on probabilistic models and other statistical methods to classify the characters [20].
- **Post processing:** After classification, the results are not 100% correct, especially for complex languages. Post processing techniques can be performed to improve the accuracy of OCR systems. One of the approaches is to use more than one classifier in cascading, parallel or hierarchical fashion. The results of the

classifiers can then be combined using various approaches. Moreover, contextual analysis can also be performed, i.e. the geometrical and document context of the image can help in reducing the chances of errors. Lexical processing based on Markov models and dictionary can also help in improving the results of OCR [21].

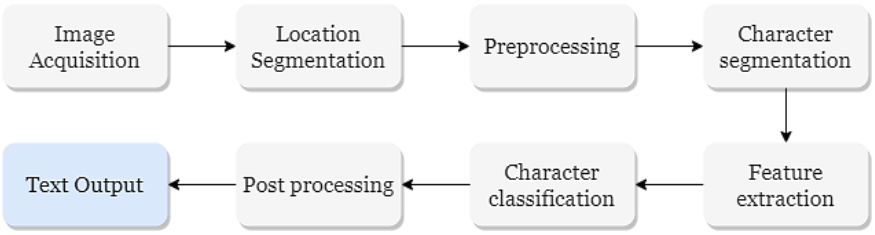


Figure 2 - General process of optical character recognition.

2.2. Mobile Approaches to OCR

Although OCR has been in research for several years, it is still not customized for the cameras that come on modern day smartphones. There has been intensive research on building mobile systems for text extraction and recognition both in the commercial space and in the academia world. Both the industry as well as the research and opensource communities offer comparable systems like:

- **Tesseract:** The Tesseract is probably one of the most widely used and accurate open source OCR engines available. It was initially created by HP and is currently developed by Google. Tesseract works on Linux, Windows and Mac OSX. The source can also be compiled for other platforms, including Android and iPhone. It supports around 149 languages which come as different packages [22].
- **Azure cognitive services:** Azure's Computer Vision API includes Optical Character Recognition (OCR) capabilities that extract printed or handwritten text from images. It can extract text from images, such as photos of license plates or containers with serial numbers, as well as from documents - invoices, bills, financial reports, articles, and more [23].
- **ABBYY:** In industry, ABBYY provides a powerful mobile OCR engine which is claimed to provide real time processing with a very high accuracy and is compatible with several mobile platforms such as Windows Mobile, Nokia Symbian, iPhone, and Android [14]. ABBYY Mobile Capture is an SDK which offers automatic data capture within your mobile app, providing real-time recognition and capturing photos of documents for on-device or back-end processing [24].
- **Dynamsoft Camera SDK:** is compatible with iOS and Android and is designed for programming of mobile document imaging. It provides document boundary detection, intelligent cropping, trapezoid distortion correction and image enhancement for the quality of captured documents. In addition, it can be used to can capture contracts, ID cards, presentations, receipts, passports, driving licenses, or any other documents [25].
- **Anyline SDK:** Anyline is a mobile text recognition SDK, natively developed for iOS, Android, and UWP, that enables developers to build text recognition apps with ease [26].
- **ML Kit:** ML Kit SDK is a relatively new product from Google that was presented in 2018. ML Kit is a software development kit (SDK) that makes it possible for

developers to simplify the integration of machine learning models into their mobile apps, including OCR. This kit provides a variety of highly skilled and accurate pre-trained models developed on deep learning strategies. It also incorporates Google ML technologies, such as: Google Cloud Vision API, TensorFlow Lite, Android Neural Networks API in a single SDK to apply ML techniques easily in your apps (Figure 3) [27].

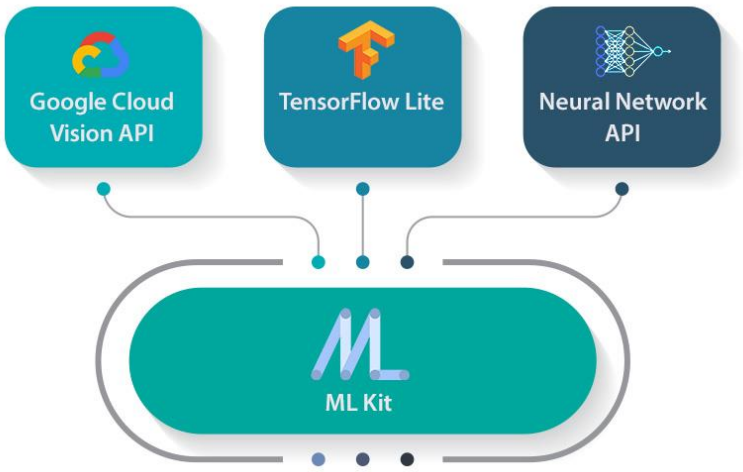


Figure 3 - Technologies incorporated on ML Kit SDK [28].

Different OCR engines tend to make different mistakes. This remark can be explained by the fact that different engines use different approaches for classifying characters, are possibly trained on different data sets with different character and word distributions and are possibly using different language models [29].

Table 1 - Comparison of Optical Character Recognition Mobile Approaches.

Name	Latest Version	Release Year	Mobile Platforms	License	Main Features
Tesseract	4.1.1	2019	iOS, Android.	OpenSource	OCR.
Azure Cognitive Services	3.1	2020	iOS, Android, Windows Phone.	Open Source	OCR, Image Understanding, Spatial Analysis, Flexible Deployment.
ABBYY Mobile Capture	15	2019	Windows Mobile, Nokia Symbian, iPhone, Android.	Commercial	OCR, Customizable Data Capture, Automatic Document Detection.
Dynamsoft Camera SDK	8.2.	2012	iOS, Android.	Commercial	OCR, Document Boundary Detection, Intelligent Cropping, Trapezoid Distortion Correction, Image Enhancement.
Anyline SDK	-----	-----	iOS, Android.	Commercial	OCR, Continuous Scanning, Automatic Torch for Low Light Conditions, Auto & Manual Focus Settings.
ML Kit	17.0.0	2020	iOS, Android.	Open Source	OCR, Barcode Scanning, Face Detection, Image Labeling.

2.3. Optical Character Recognition Challenges

With the increase in the number of portable cameras, smartphones, and surveillance cameras, the detail in images captured with these devices have considerably increased. As a result, the information in these image details cannot be analyzed by humans. Thus, efficiently and automatically analyzing this information is an important challenge faced by image processing and pattern recognition [30].

The performance and accuracy of OCR is directly dependent upon the quality of input images. Thus, most OCR engines require the input image to have a clean background and clear foreground. However, the source images may be taken under various conditions, which increase exponentially the difficulty of recognizing the target characters [31].

In general, the processing of an image obtained by a smartphone is affected by several factors, such as:

- **Tilt:** the perspective distortion that can take place when the text image plan to capture is not parallel to the smartphone's camera plane, results on the smaller appearance of the characters located farther and the hypothesis of parallel lines of the edges of the document page no longer fits to those in the captured image [19].
- **Lighting conditions**: the variation in illumination is common, due to the physical environment such as shadows or reflective surfaces and lack of controlled lighting [19].
- **Text properties**: from one document text to another, the text properties may vary, such as: variety of sizes, fonts, styles, colors [19].
- **Complex background**: The most notable and probably an inherent challenge faced by almost all vision-based systems is scene complexity. Background cluttering and noise can often create outliers and mislead a machine learning system into classifying non-textual information as text [32].
- **Blur distortion:** When using a mobile camera in real world scenarios, there is a tendency to capture out of focus, shaken or blurry images. Compression and decompression can also produce their own noise. In those cases, character (and feature) sharpness is generally affected [32].

2.4. Standard Image Pre-processing

The aim of pre-processing is to eliminate undesired characteristics or noise in an image without missing any significant information. It enhances the image and prepares it for the next phases in OCR. Most of the challenges listed in the previous section need to be addressed in preprocessing stage. Some common operations of pre-processing can be listed as follows: binarization, noise reduction, skew correction, morphological operations, slant removal, filtering, thresholding, smoothing, compression, and thinning [18]. Table 2 presents some of these pre-processing methods in more detail.

Table 2 - Description of some pre-processing methods.

Processes	Description
Binarization	Most OCR systems are designed to work with binarized images and good binarization is crucial for reliable performance. A satisfactory choice of binarization algorithm appears to be dependent on the application domain and experimentation on relevant data sets [33]. Nonetheless, binarization has long been recognized as a standard method to solve uneven lighting in OCR. The goal of the binarization process is to classify image pixels from the given input grayscale or color image into either foreground (text) or background. In general, the binarization process for grayscale images can be grouped into two broad categories: global binarization, and local binarization. Global binarization methods (e.g. Otsu's algorithm) try to find a single threshold value for the whole image. Each pixel is then assigned to either foreground or background based on its grey value. On the other hand, local binarization methods (e.g. Niblack's algorithm, Sauvola's algorithm) compute thresholds individually for each pixel using information from the local neighborhood of that pixel.
Filtering	Filtering aims to remove noise and diminish spurious points usually introduced by uneven writing surface and poor sampling rate of the data acquisition device. Various spatial and frequency domain filters have been designed for this purpose. The basic idea is to convolute a predefined mask with the image to assign a value to a pixel as a function of the gray values of its neighboring pixels. Several filters have been designed for smoothing, sharpening, thresholding, removing slightly textured or colored background and contrast adjustment purposes [34][35].

Skew Correction	Due to inaccuracies in the capturing process and writing style the writing may be slightly tilted or curved within the image. This can hurt the effectiveness of the algorithms and thus should be detected and corrected [35].
Thresholding	To reduce storage requirements and to increase processing speed it is often desirable to represent gray scale or color images as binary images by picking a threshold value. The two important categories of thresholding are viz global and local. The global thresholding picks one threshold value for the entire character image which is often based on an estimation of the background level from the intensity histogram of the image. The local or adaptive thresholding use different values for each pixel according to the local area information. A comparison of common global and local thresholding techniques is given by using an evaluation criterion that is goal directed keeping in view of the desired accuracy of the OCR system. It has been shown that Niblack's locally adaptive method produces the best result [34][35].

2.5. Related Work

As previously mentioned, nonadherence to asthma medications is a well-recognized problem. Nonetheless, over the period of 30 years, the effectiveness of adherence interventions has not progressively improved. Thus, the potential for mobile communication technology, such as mobile health apps and inhaler-based devices, to improve the management of asthma has gathered abruptly growing interest [4].

The development of electronic sensing devices that attach to inhaled respiratory medications has mitigated this problem. The inhaler compliance assessment device, commercially referred to under the trademark of INCA® (Inhaler Compliance Assessment), has the advantage of being an automatic and objective measure of both adherence and inhaler technique [36]. The aforementioned device, when attached to an inhaler, allows to identify and record the time and technique of inhaler use, thereby providing objective longitudinal data on an individual's adherence to inhaled medication [37]. Actuation sensors are now available worldwide, and their use has been associated with lower overall costs for asthma care and better clinical outcomes [38].

In relation to mobile health applications, there are more than 500 asthma-related apps currently available, whether standalone or paired with sensors on inhalers, offering functions that span health education, symptom tracking, environmental alerts, and medication reminders [39]. Standalone asthma apps are less costly to design, create, and maintain than their interactive counterparts and are frequently downloaded from the App Store and Google Play Store [38]. Although standalone apps often allow to collect symptoms and inhaler usage data, these tools rely on patient self-report.

With all this in mind, combining features of inhaler usage detection and a gamification approach based on peer support, delivers an innovative way to measure and improve the adherence to inhaler, by using a smartphone camera to register the value of the inhaler dosage counter after the use. The developed detection tool is based on computer vision techniques, namely OCR (optical character recognition) to detect remaining doses according to the dose counter [11].

Several academic projects and commercial products have tried to use mobile phone cameras to build interesting OCR applications. The recent developments regarding OCR have led to numerous applications of this technology in daily life, such as helping vision-impaired people understand text information, automatic document generation, or even in surveillance systems [10].

The following review focused on finding articles that apply OCR strategies in real-world scenarios with applications in the medical field. However, papers from other application fields, whose methodology proved to be potentially useful for the work at hand, were not excluded. The search was conducted manually in Science Direct, IEEE, and Google Scholar databases.

Rusyn et al. proposes an approach to the automatic digit recognition of exposures from medical devices, such as blood pressure meters, glucometers, substance content harmfulness analyzers. The authors proposed solving the problem of optical character recognition using methods of deep learning. To do so, the researchers used a database of 4500 images to train and validate the created network. Their team found the deep learning model suitable for detecting and recognizing low contrast and defocus images. In addition, their results show these methods to be more accurate than more classical approaches, such as k-nearest neighbors and decision tree [40].

Tangtisanon suggested a medicine alert system for the elderly population that can operated in an Android smartphone with Internet connection. The proposed application allows to convert a medicine label picture into a text file, so the user can set up the right medicine name. The author used tesseract to implement this functionality, along with some pre-processing techniques (e.g. adaptive thresholding). The results show that the program output 100% correctly for typewriter font in local-based testing. On the other hand, for server-based testing only 90% of the output text match with the uploaded picture label correctly, due to a higher picture sampling before the upload to the server [41].

In contrast, Čakić et al. [42] used Tesseract OCR to get additional information about a specific bottle of wine (i.e. type, vintage, origin, ratings), based on existing wine labels that contain unique serial numbers, using only a smart mobile phone equipped with camera. The results obtained by the authors highlighted the importance of image pre-processing, since the success rate varies significantly depending on the application of this stage in the OCR (around 62 % of the images were correctly recognized without pre-processing; this value increases to 87.5% after performing pre-processing).

An increasing number of studies have attempted to develop software to detect and classify digits from images captured from medical devices, namely from seven-segment displays. Many of these systems follow a process of locate a region of interest (ROI), binarize the ROI, locate digits within the binarized region and finally classify the digits by their numeric value. Significant differences in these systems occur for the methods used for region extraction and digit classification.

In the healthcare sector, Finnegan and co-workers [8] presented an automated process for the detection and reading of seven-segment digits from images of medical devices that used LCD screens, from two frequently-used medical devices: blood glucose meters and blood pressure monitors. The algorithm took in consideration unfavorable illumination effects by applying Retinex theory using bilateral filters. Moreover, a multi-layer perceptron was built using Matlab's 'Neural Network Pattern Recognition' toolbox. The model achieved 93% accuracy on digits found on the medical devices. However, this system was not yet implemented on a smartphone.

Similarly, Shenoy et al. [9] developed a smartphone based system (running on iOS 8 and above), to automatically recognize and record biometric measurements captured from medical devices. In the search for an algorithmic model to implement, the researchers concluded that while Tesseract is strong at reading regular text on a page, it has difficulty to accurately read seven segment displays. Alternatively, the authors implemented an algorithm in Python 2.7, that accurately read the monitor 98.2% of the time. Packages such as scikit-learn and OpenCV were used to execute Random Forest Classifier and feature extraction along with image normalization, respectively.

In [31] the authors presented an accelerated optical character recognition approach of seven-segment display digits found on digital medical devices. Using the proposed system, patients are expected to scan a medical instrument with a smartphone to automatically extract measurements related to their health. The authors overcame the challenges posed by changes in illumination through an adaptive thresholding method, and achieved an OCR accuracy of 96,22 %.

Outside the medical field, Ghugardare et al. [43] propose a generalized module for automatic calibration of any measuring instruments using OCR, in order to better replicate and objectify measuring, as well as significantly minimize the quantity of work for calibration. This paper mainly suggests algorithms for recognition of seven segment display characters present on digital multi-meters. Despite the high accuracy of 95%, the proposed method focuses on serial execution and the algorithm is neither optimized for speed, nor performance is the focus of this research.

More recently, Kanagarathinam and Sekar [16] conducted text recognition in energy meters with a seven segment display. Their dataset included images captured under challenging text recognition conditions such as tilted position, blurred, day and night light. Their findings suggest that existing open source OCR software could not recognize the text of seven-segment numerals because of the discontinuity in the digit

representation. As such, a MSER algorithm was used to detect the text regions in an image and an OCR function available in MATLAB was used to increase the recognition rate. Their results showed a recognition rate of 93,17 % in the test set.

In the last few years, much more information on OCR has become available as well open-source engines that allow the application of this technology in mobile apps. Although ML Kit is a relatively recent product (presented by Google in 2018), it has already been incorporated in some commercial apps, namely ZYL and Lose It!.

Lose It! is an app that helps users manage their diets through food logging. Their team used ML Kit to scan nutrition labels from the camera view in real time, and instantly fill in the nutrition information for any new food in the app. The implementation of ML Kit significantly reduced the image analysis time for nutrition label reading. Moreover, since ML Kit allows to host models in Firebase, enables the seamlessly update of models on device without updating the app, reduces the app size, and allows A/B test model versions [44].

The InspirerMundi's app also incorporates Google's ML Kit technology to monitor the adherence to inhaled control medications. In a pilot implementation of a detection tool for this mobile app for iOS, the dose in the dose counter of the inhaler is recognized to assist with the adherence monitorization. This functionality uses template matching to locate the dose counter for a set of commonly used inhaler devices. Only then the app performs the recognition of each numerical dose in the dose counter. The counter numbers were detected in 42% of the 101 images. To improve these results, an additional step was added where if no numbers are detected in the region of interest (ROI), a crop is performed around this area and submitted to text analysis again. The cropping of the image's ROI was tested on 20 images subset and allowed to retrieve correctly the doses of 15 images (75%) [42].

In a more recent study, the tests performed with the InspirerMundi application resulted in the correct value identification for the dosage counter in 79% of the registration events with all inhalers and over 90% for the three most widely used inhalers in Portugal [13].

CHAPTER 3 –METHODOLOGY

3. Methods

The goal of the presented work is the development of a module dedicated to recognizing the text of inhaler counters. As such, a module was built, in Android Studio, with ML Kit and tested on several datasets. Pre and post processing techniques were applied where it was deemed necessary. Subsequently, certain features of the project were integrated in the most current version of the app, planned to be made available at the PlayStore by the end of 2021 (Figure 4).

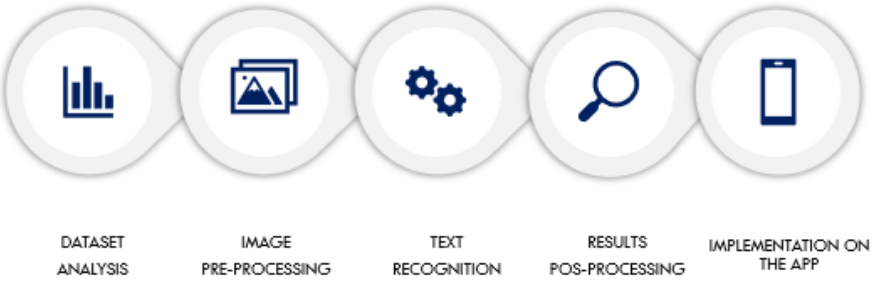


Figure 4 – Stages for the implementation of the default MLKit model.

The second phase of the project aimed to improve the performance of the text recognizer by creating a TensorFlow model, trained on images of inhalers. This stage encompasses the steps described in Figure 5 and will be addressed later on. It should be noted that this process was only performed for one inhaler type: the *seretaide*.

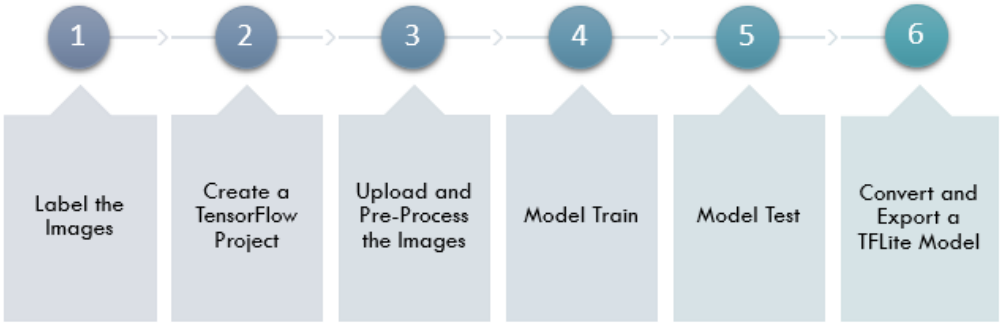


Figure 5 – Steps for creating a custom MLKit model.

3.1. Datasets

To test the many variables that can affect the performance of the text recognizer, four datasets were considered, each with a different set of characteristics (Figure 5).

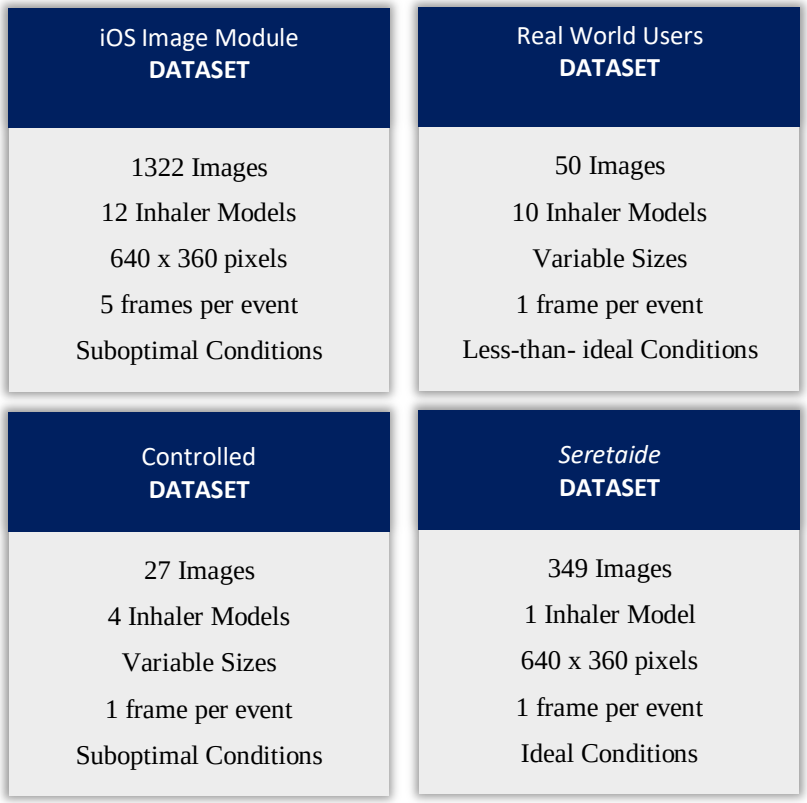


Figure 6 - The four datasets used on this work and their specifications.

A subset of inhaler images, used in previous studies of the iOS inhaler detection module, was made available for text recognition testing. The database consists of 1322 RGB images, in PNG format, with dimensions of 640 x 360 pixels. This dataset contains all the currently marketable inhaler types, with numerical dose counters at Portugal and Spain (Figure 7). However, the distribution of each inhaler in the dataset is variable (Table 3).

Table 3 - Inhaler distribution for the several datasets used in the present work.

Inhaler Model	**No. of images**			
	iOS Image Module Dataset	**Real World Users Dataset**	**Controlled Dataset**	***Seretaide* Dataset**
Diskus	229	3	6	---
Easyhaler	146	4	9	---
Ellipta	53	7	---	---
Flutiform	95	2	---	---
K-haler	232	4	5	---
Mdi3m	79	---	---	---
Nexthaler	62	2	---	---
Novolizer	82	4	7	---
Seretaide	152	4	---	349
Spiromax	47	8	---	---
Turbohaler	69	12	---	---
Twisthaler	76	---	---	---
TOTAL	1322	50	27	349

Figure 7 - Sample images from the iOS Image Module dataset; photos captured under suboptimal conditions.

The images provided were acquired by 24 volunteers under controlled conditions using an iPhone 6S and a Swift implementation of method [13] described at section 2.5 Related Work. It's also important to note that the acquisition of these images was performed with a template matching tool, which allowed for the inhalers to remain similarly positioned on the screen. In addition, this dataset many times contained frames of the same event for each inhaler.

A second dataset containing images collected from v1.x app users (iOS and Android) was provided to test the text recognition in even more realistic conditions. This dataset of 50 RGB images in PNG format, includes photos acquired by testers but also real patients, recruited as participants of the Inspirers Studies [12]. Additionally, the photos collected for the real-world user's dataset were acquired after the process of template matching. Contrary to the iOS image module database, in this dataset there is only one frame per inhaler event.

The images were collected under less-than-ideal conditions (poor lighting, camera blur, background noise, etc.; Figure 8). The size of these images varies depending on the camera used to obtain the photo. This dataset contained images representative of all the inhalers available, except for the *mdi3m* and *twisthaler* models.

Figure 8 - Sample representative of the images present in the real-world user's dataset; less than ideal acquisition conditions (background noise, poor lighting, etc.).

A dataset acquired under controlled conditions was provided specifically to test the interference of using different cameras in the text recognition results. For this purpose, photos from three different devices: a tablet (Lenovo TB-7504F) and two smartphones (Huawei P8 Lite and Redmi Note 8T) were obtained. The specifications for each instrument can be found in Table 6. These devices were used to collect photos of different inhaler models through the InspireMundi version 1.2.2. The captured photos form a dataset containing a total of 27 frames, however the distribution of images for each device is non-uniform and not all succeeded in the process of template matching.

Table 4 - Inhaler distribution of the controlled dataset.

Device Model	**Number of Images per Inhaler Model**				
	Diskus	*Novolizer*	*Easyhaler*	*K-haler*	Total
Lenovo TB-7504F	4	2	5	2	13
Huawei P8 Lite	2	3	3	2	10
Redmi Note 8T	---	2	1	1	4

Table 5 - Smartphone model's specifications.

Model	**Inches**	**Resolution (pixels)**	**Ratio**	**Operative System**
Lenovo TB-7504F	7''	720 x 1280	16:9	Android 7
Huawei P8 Lite	5.2''	1080 x 1920	16:9	Android 8
Redmi Note 8T	6.3''	1080 x 2340	19.5:9	Android 10

At last, the *seretaide* database consists of 349 images of the *seretaide* inhaler model in PNG format, with dimensions of 640 x 360 pixels, collected under ideal conditions using a LG-V700 (Android) camera app. These images show a wide representation of digits in the dose counter; additionally, the background varies between black, white, and multicolor, and the lighting source of the photo varies between natural light and artificial light.

3.2. Counter Detection

3.2.1. Image Pre-Processing

Before processing the images through the text recognition model, it's important to improve the image data by suppressing undesired distortions and enhancing some relevant image features. With this goal in mind, several steps were followed during preprocessing, namely, the application of filters, image rotation and cropping of the region of interest (Figure 9).

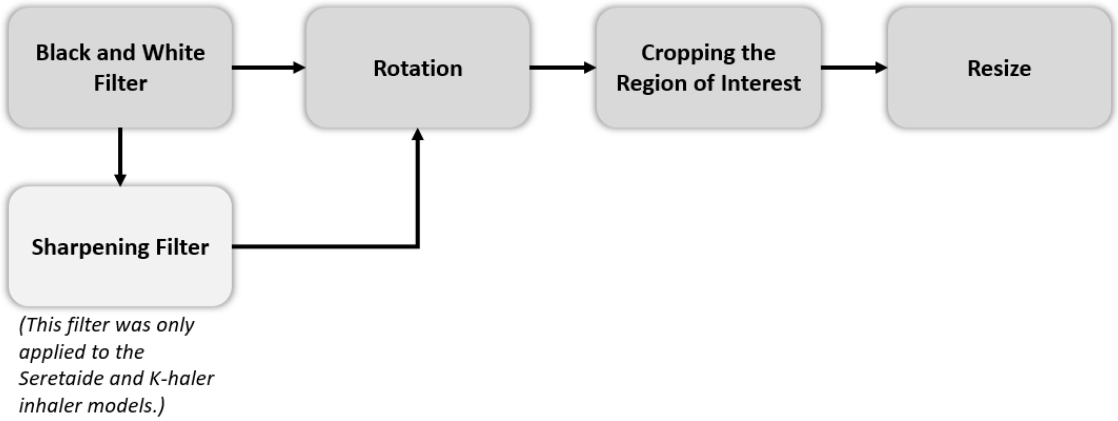


Figure 9 – Methods used for image preprocessing.

Step 1: Black and White Filter

In order to facilitate text recognition, a black and white filter was applied to all images. The reason for this pre-processing step is that less information needs to be provided for each pixel (a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image), thus making the algorithm more efficient.

Step 1.5.: Sharpening Filter

A sharpening filter was applied to some key inhaler models, to improve the definition of fine detail and sharpen edges in the original image. The filter works by creating a fine highlight on the darker side of the edge, and a tiny lowlight on the lighter side of the edge.

This intermediate step was only applied to the *seretaide* and *k-haler* inhaler models, due to the poor performance of the text recognition algorithm in the images of these inhaler types [13].

Step 2: Rotation

The pictures of the inhalers were captured horizontally by the camera of a smartphone. However, the ML Kit algorithm does not recognize horizontal text, so all the collected images were rotated vertically.

In most cases, the images were rotated 90 degrees to the right, with the exception of the *mdi3m* inhaler model (rotated 180 degrees right). This process was automatically made in Android Studio.

Step 3: Cropping the Region of Interest

The main goal of this work is to recognize the digits that are represented on the dose counter of an inhaler. However, the text recognizer in the ML Kit recognizes not only the text on the dose counter, but also text in the background or on the labels present in the inhalers. Therefore, to ensure that the text collected does indeed belong to the dose counter, an image cropping centered on the dose counter display of the inhaler was performed. For this process, it had to be considered that the location on the image of the dose counters varies according to the type of inhaler. Therefore, image cropping was performed with different measurements and different coordinates based on the inhaler model (Table 6).

Table 6 - Measurements used to perform the cropping of the region of interest: x and y represent the first pixels of the region of interest; width and height are the measures of the rectangular crop.

Inhaler Model	x	y	Width	Height
Diskus	170	240	140	100
Easyhaler	100	350	140	80
Ellipta	160	370	110	80
Flutiform	120	200	140	130
k-haler	140	140	100	100
Mdi3m	430	125	140	100
Nexthaler	180	300	120	80
Novolizer	120	100	140	80
Seretaide	130	440	140	80
Spiromax	140	280	140	80
Turbohaler	180	220	140	80
Twisthaler	120	450	140	80

Step 4: Resize

Resizing images is an essential part of image processing. In this case, the image resizing is performed by scaling down the image, to speed the processing phase. To create a resized image, the original image is divided by a scale factor. The scale factor is determined by selecting the greater of two values $\left(\frac{Image\ Width}{Target\ Width}, \frac{Image\ Height}{Target\ Height}\right)$.

3.2.2. Text Recognition with Default Model

Machine learning-based technologies were utilized to solve the problem of reading the dose counter. In particular, the MLKit SDK, which provides an off-the-shelf, simple solution for mobile developers to incorporate machine learning into their apps without having to fully grasp the entire model training process. The inhaler detection module was modified to include the ML Kit text mining API for dose counter value number extraction in order to provide objective verification of adherence.

In essence, the code configures a text recognition detector, parses the results and displays them in the app, showing the text recognition results and bounding boxes overlaid on top of the original image.

The ML Kit's Text Recognizer segments text into blocks, lines, and elements (Figure 10):

- a **Block** is a contiguous set of text lines, such as a paragraph or column,
- a **Line** is a contiguous set of words on the same axis, and
- an **Element** is a contiguous set of alphanumeric characters ("word") on the same axis in most Latin languages, or a character in others.

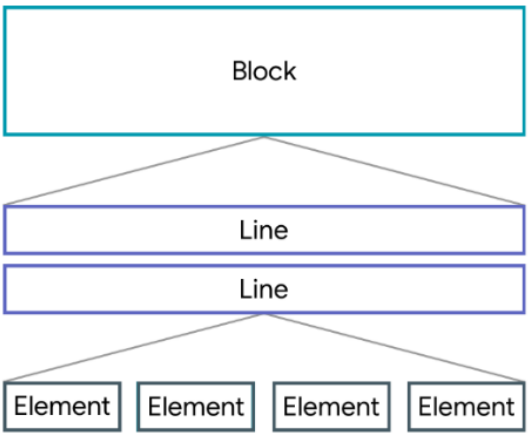


Figure 10 – Segmentation of text in ML Kit's Text Recognizer (Source: Google Developers).

If the text recognition operation succeeds, a Text object is passed to the success listener. A Text object contains the full text recognized in the image and zero or more TextBlock objects. Each TextBlock represents a rectangular block of text, which contains zero or more Line objects. In turn, each Line object contains zero or more Element objects, which represent words and word-like entities such as dates and numbers [45].

3.3. Results Post-Processing

After processing the images and obtaining the results, certain post-processing techniques were applied to the outcomes, in order to avoid errors in the results. Some examples of preventable errors include the presence of non-numerical characters in the final result or detected values that do not fall within the dosage range of the inhaler. Thus, the method represented in Figure 14 were implemented by the illustrated order.

Figure 11 - Post-processing techniques applied to the results.

3.3.1. Removing Invalid Characters

The dose counter of an inhaler can only display numerical characters. Thus, any detection of a non-numerical character could be handled one of two ways: either correct the recognized text to match a number (change the 'o' to 0, or 'I' to 1) or exclude any character that does not match a digit from the result. From these two different approaches, the latter was chosen, as it is simpler and the conversion from text to number would imply studying the dataset.

Hence, all characters that were not digits, including alphabetic and special characters, were excluded from the recognized text.

3.3.2. Admissibility Range

The value obtained from the text recognition can be considered inadmissible for several motives. These reasons may be that there is no recognized text, no character remaining after removing invalid characters, or that the text does not fit into the range of admissible values.

It is worth noting that the values that are illustrated on the inhaler counter display are limited by ranges. That is, each inhaler has a maximum dose value and a minimum value (0). Thus, for each character recognition result it is important to verify whether the result was admissible within the known value range for each inhaler (Table 7).

Table 7 - Maximum dose values for each inhaler model.

Inhaler Model	Maximum Dosage
Diskus	60
Easyhaler	200
Ellipta	30
Flutiform	120
K-haler	124
Mdi3m	120
Nexthaler	120
Novolizer	200
Seretaide	124
Spiromax	120
Turbohaler	200
Twisthaler	120

However, it is worth noting that in certain instances the text recognizer has simultaneously detected two dose values (particularly among the *easyhaler* and *turbohaler* inhaler models). The ML Kit aggregates the text blocks detected in the image in order to generate the final result. Therefore, there were some cases in which the results were erroneously considered to be outside the admissibility range. Thus, for these occurrences, the admissibility of the results was amended manually.

3.3.3. Voting System

In order to increase the effectiveness of the algorithm, a voting system was implemented. This system consists of collecting 5 frames from the same inhaler. These frames are temporarily stored in the cell phone. For each frame, text recognition is performed. From the five results obtained, the text that was recognized most often is selected. This is then considered the correct one. If there are two most common numbers, the algorithm selects the one with the lower value.

3.4. Saving Data

Each time an image is processed all the information regarding that same image and the detected text are stored temporarily in the device. Towards access the saved values an export is performed, to load the data obtained by the app in Google Drive under the form of a .csv file. This data encompasses: the timestamp, the inhaler model, the file name, the detected text, the corrected detected text, whether the value is in the admissibility range, and the most common detection among five frames.

3.5. Model Optimization

Following data processing and analysis of the results, a discrepancy in the performance of the text recognizer was observed between different inhaler models. In fact, the results for certain inhalers were lower than expected when compared to the others.

By default, ML Kit's APIs make use of Google trained machine learning models. These models are designed to cover a wide range of applications. However, some use cases require models that are more targeted. That is why some ML Kit APIs now allow you to replace the default models with custom TensorFlow Lite models.

Thus, it was proposed to train a TensorFlow Lite model on images belonging to a specific type of inhaler, to improve the accuracy of the text recognizer. The inhaler models with worst results (in iOS) were *Twisthaler* and *Seretaide* [13], however the *Twisthaler* model is rarely used among Portugal users, according with medical specialists of the team Inspirers; hence, the *Seretaide* model was selected for this study.

In order to create a TensorFlow Lite model the following steps were completed:

1. Annotate the dataset.
2. Get crops for each image where the dose counters are located.
3. Split the seretaide dataset into train and validation datasets.
4. Train the model.
5. Make prediction on cropped images of the test dataset.
6. Evaluate the model.
7. Convert the model to TFLite format.

3.5.1. Annotations

For the purpose of training a machine learning model on inhaler pictures, a manual annotation process was performed on all images present in the *Seretaide* inhaler dataset. This task was accomplished with the help of the VGG Image Annotator (VIA) tool. VIA is an open source standalone manual annotation software for image, based solely on HTML, JavaScript and CSS (no dependency on external libraries)[46]. This software was selected among other annotation tools due to the simplicity of its interface (Figure 11) and the fact that there is no installation required (runs in a web browser).

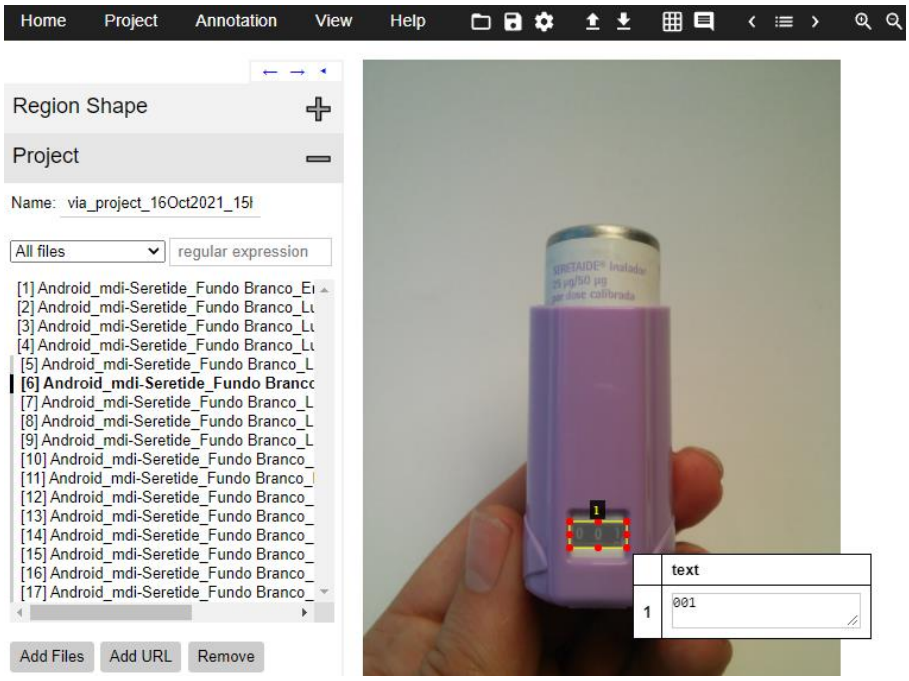


Figure 12 - Interface of the VIA software.

With the aid of this software, it was possible to collect annotations regarding the position and dimensions of the dose counter and the corresponding digits that appear on the display. The annotations were saved under a csv file with information regarding the filename, the width and height of the dose counter, the coordinates for the first pixel corresponding the dose counter and the number of dosages displayed in the dose counter. The data referent to the location and dimensions of the dosage counter allowed to get crops of each image at a later stage.

3.5.2. Tools

Google Colaboratory, or "Colab" for short, is a web integrated development environment (IDE) for python that allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis

and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including Graphic Processing Units (GPUs) [47].

A good GPU is indispensable for machine learning. Training models is a hardware intensive task, and a proper GPU will make sure the computation of neural networks goes smoothly. Hence, the use of Google Colab in this case was indispensable given the limited hardware resources available. The types of GPUs that are available in Colab vary over time. The GPUs available in Colab often include Nvidia K80s, T4s, P4s and P100s. Nevertheless, there is no way to choose what type of GPU you can connect to in Colab at any given time [47].

Since, the goal is to train a model to be integrated in MLKit API, the model has to be in a TensorFlow Lite format. TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks, enabling developers to build and deploy ML-powered applications. In addition, TensorFlow offers intuitive high-level APIs that enable immediate model iteration and simplified debugging and can be executed in Colab notebooks [48].

3.5.3. Dataset Pre-Processing

The dataset contains 354 files as jpg images. All images suffered four pre-processing steps: rotation, cropping, convert to grayscale and resizing. The images in the dataset were rotated 90º right and cropped according to the dimensions of the dose counter indicated on the annotations file. Since the cropping measurements were not consistent across the dataset, the cropped-out images did not have the same size; hence, all images were resized to standardize the data (Figure 12).

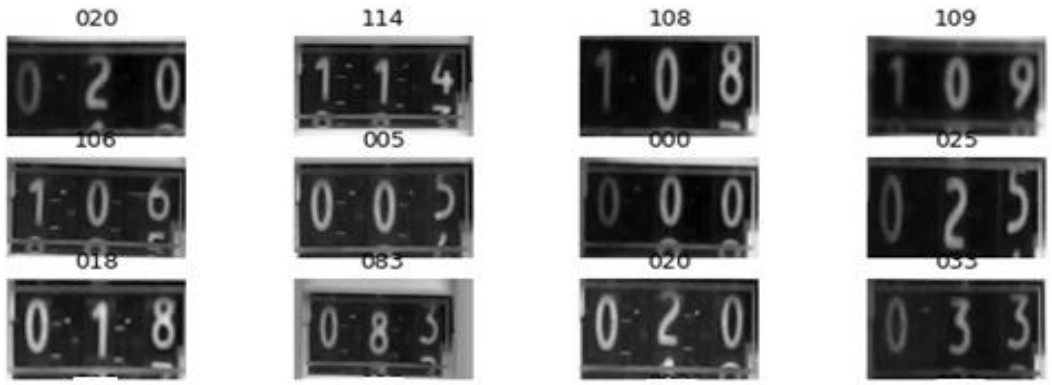


Figure 13 – Sample of the pre-processed *seretaide* dataset.

The label for each sample is a string with the characters illustrated in the image of the dose counter. Each character in the string was mapped to an integer for training the model. Similarly, the predictions of the model were mapped back to strings. For this purpose, two dictionaries were maintained, mapping characters to integers, and integers to characters, respectively. The characters that are present in the dose counter, and therefore are needed to encode and posteriorly decode for predictions, are digits from 0 through 9 . All transcriptions of the dose counter digits are 3 characters long.

3.5.4. Split the Dataset

In order to test the model, a training and validation set must be generated. The network can be trained using the training set and verified using the validation set. For this purpose, the dataset was shuffled randomly, so that each time the dataset is split a new training and validation dataset are created. Additionally, the dataset split into 90 % training set and 10 % validation set. That is, of total images present in the *seretaide* dataset, 90 % were used for training the model and 10 % for validating the model's performance:

- Training Dataset Size: 318 images;
- Validation Dataset Size: 36 images.

The classes found in the seretaide dataset are digits that range from 0 to 9, and the frequency of each one in the dataset can be seen in Table 8.

Table 8 - Frequency of the characters found in the seretaide dataset.

Character	Frequency (%)
0	33,90
1	26,08
2	10,92
3	8,47
4	3,95
5	3,39
6	3,39
7	3,11
8	3,39
9	3,39

3.5.5. Model Architecture

When attempting to recognize characters, neural networks (NN) are a good choice as they outperform all other approaches at the moment. The NN for such use-cases usually consists of convolutional layers (CNN) to extract a sequence of features and recurrent layers (RNN) to propagate information through this sequence. The network outputs character-scores for each sequence-element, which simply is represented by a matrix [49]. In this case, the model combines a CNN and an RNN, and it instantiates a new "endpoint layer" for implementing CTC loss. The former enables using unsegmented pairs of images and corresponding text transcriptions to train the model without any character/frame-level alignment [50].

More specifically, the architecture of the NN consists of an input layer, two convolutional layers each followed by a pooling layer, two bidirectional layers, a CTC layer and finally an output layer (Figure 13). The input layer consists of an array of floats that represent the pixels in the image of a dose counter. It is important to notice that the value of a pixel is the intensity of a symbol at a given position, with 255 being a black pixel with ink, and 0 being a white pixel without ink. The final layer is the output layer and it has 11 nodes.

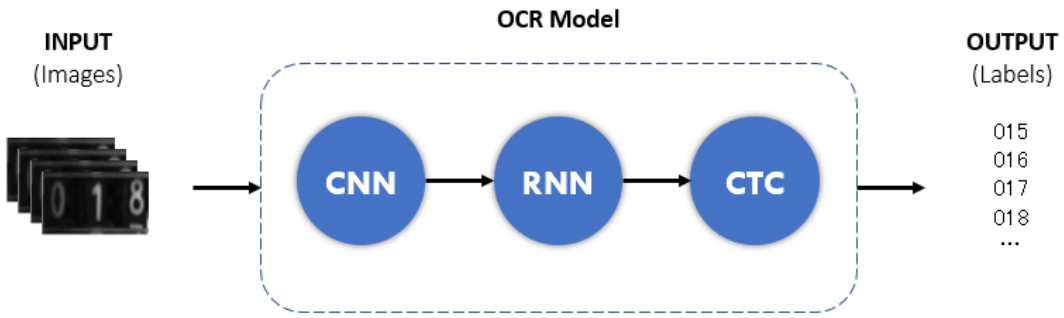


Figure 14 - Shape of the OCR Model.

3.5.6. Model Train

Training is the machine learning stage where the model is gradually optimized, or the model learns the dataset. The goal is to learn enough about the structure of the training data set to make predictions about unseen data. If you learn too much about the training data set, the predictions will only work for the data it has seen and will not be generalizable (i.e. overfitting) [51]. The problem in question (OCR of dose counter digits in inhalers) is an example of supervised machine learning, in particular, a multi-label

classification problem: the model is trained from examples that were previously labelled with the corresponding class; in this case a class is a label from the set {0, 1, 2, …, 9}).

During the training stage the model needs to calculate loss. This measures how much a model's predictions are off the desired label, in other words, how bad the model is doing. In this case, the model computes its loss using the CTC Loss which calculates a loss between a continuous (unsegmented) time series and a target sequence. It does this by summing over the probability of possible alignments of input to target, producing a loss value which is differentiable with respect to each input node [52].

In addition, TensorFlow has many optimization algorithms available for training. This model uses the optimization algorithm Adam which is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. The Adam algorithm combines the best properties of the AdaGrad and RMSProp algorithms. Adam is relatively easy to configure where the default configuration parameters do well on most problems [53].

3.5.7. Evaluate the Model

The critical step after implementing a machine learning algorithm is to find out the effectiveness of the model based on metrics. Different available performance metrics are used to evaluate machine learning algorithms. In OCR tasks, we can use multi-label classification performance metrics such as accuracy, precision, recall and F-measure [54].

The evaluation of a multi-label classification algorithm is difficult mostly because multi-label prediction has an additional notion of being partially correct. One trivial way around would be just to ignore partially correct (consider them as incorrect) and extend the accuracy used in single label case for multi-label prediction. This is called Exact Match Ratio (MR) and it can be described by the following expression:

$$ExactMatchRatio, MR = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i) \qquad (1)$$

where, I is the indicator function. Clearly, a disadvantage of this measure is that it doesn’t distinguish between complete incorrect and partially correct [54].

In order to account for partially correctness, Godbole et. at in [55] proposed following set of definitions for accuracy, precision, recall, and F1 measure. As in single label multi-class classification, the higher the value of referred metrics, the better the performance of the learning algorithm.

The accuracy determines the fraction of correct predictions. If $\hat{y}_i$ is the predicted value of the *i*-th sample and y_i is the corresponding true value, then the fraction of correct predictions over $n_{samples}$ is defined as [56]:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i) \qquad (2)$$

Another evaluation metric is precision (P), which corresponds to the proportion of predicted correct labels to the total number of actual labels, averaged over all instances:

$$Precision, P = \frac{1}{n} \sum_{i=1}^{n} \frac{|Y_i \cap Z_i|}{|Z_i|} \qquad (3)$$

On the other hand, recall (R) is the proportion of predicted correct labels to the total number of predicted labels, averaged over all instances:

$$Recall, R = \frac{1}{n} \sum_{i=1}^{n} \frac{|Y_i \cap Z_i|}{|Y_i|} \qquad (4)$$

F1-Measure (F) is the definition for precision and recall naturally leads to the following definition for F1-measure (harmonic mean of precision and recall).

$$F_1 = \frac{1}{n} \sum_{i=1}^{n} \frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|} \qquad (5)$$

Hamming Loss (HL) reports how many times on average, the relevance of an example to a class label is incorrectly predicted. Therefore, hamming loss considers the prediction error (an incorrect label is predicted) and the missing error (a relevant label not predicted), normalized over total number of classes and total number of examples.

$$HammingLoss, HL = \frac{1}{kn} \sum_{i=1}^{n} \sum_{l=1}^{k} [I(l \in Z_i \wedge l \notin Y_i) + I(l \notin Z_i \wedge l \in Y_i)] \qquad (6)$$

where I is the indicator function. Ideally, we would expect hamming loss, HL = 0, which would imply no error; practically the smaller the value of hamming loss, the better the performance of the learning algorithm [54].

3.5.8. Convert to TF Lite

In order to replace the default models on ML Kit APIs with a custom model, the existing TensorFlow OCR Model had to be converted to a TensoflowLite format. To do so, the has to meet some compatibility requirements must be met [57]:

- The model must have only one input tensor with the following constraints:
  - The data is in RGB pixel format.
  - The data is UINT8 or FLOAT32 type. If the input tensor type is FLOAT32, it must specify the Normalization Options by attaching Metadata.
  - The tensor has 4 dimensions: BxHxWxC, where:
    - B is the batch size. It must be 1 (inference on larger batches is not supported).
    - W and H are the input width and height.
    - C is the number of expected channels. It must be 3.
- The model must have at least one output tensor with N classes and either 2 or 4 dimensions:
  - (1xN)
  - (1x1x1xN)

3.6. Implementation on the app

All previously mentioned steps were developed and implemented over in an Android Studio project independent of the InspirersMundi app. Therefore, after establishing that the image processing and text recognition worked, the goal is to integrate this work in the current version of the Android app Inspirers Mundi. The pre-processing features and the text recognition were included on the app through a team collaboration, and over the next course of weeks the voting system will be implemented to improve the app's performance.

CHAPTER 4 – RESULTS

4. Results

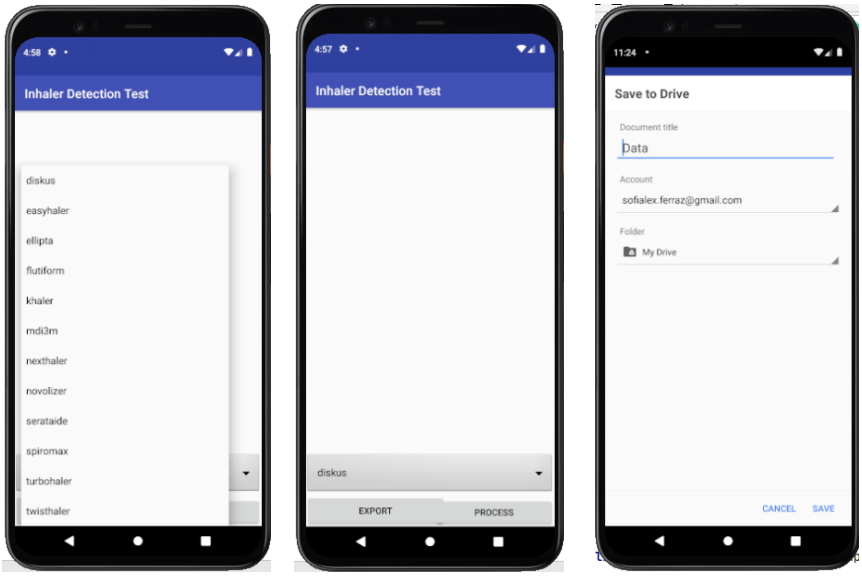
4.1. Testing Module with Default ML Kit's Model

In order to test app's operation, Android Studio emulator was used to simulate the virtual device Pixel 4 API 28. The specifications for this device can be found in Table 9.

Table 9 - Specifications of the android virtual device.

Name	Resolution	API	Target	CPU/ABI
Pixel 4 API 28	1080 x 2280: 440dpi	28	Android 9.0	x86

As shown in Figure 15 (a), the mobile app module has a drop-down field menu that allows the user to select which of the inhaler model is going to be processed. The menu offers 12 options corresponding to the commercially available inhalers with numerical dose counters. In addition, the app has a 'PROCESS' button that starts the image processing and OCR, and a 'EXPORT' button (Figure 15 (b)). The former allows loading to Google Drive all the data that has been temporarily saved on the device, in a cvs format for future analysis (Figure 15 (c)). After the text recognition is complete, the obtained values are displayed on the screen alongside the bounding boxes.



(a) **(b)** **(c)**

Figure 15 – (a) Display of the drop-down field menu, with the inhaler model options, on the app; (b) Display of the layout of the app with all the existing buttons; (c) Display of the layout for saving the data on Google Drive.

In a preliminary analysis, the text recognition program often picks up much more text than the one present in the dose counter, namely text from the inhaler labels and text present at the background of the photo (Figure 16).

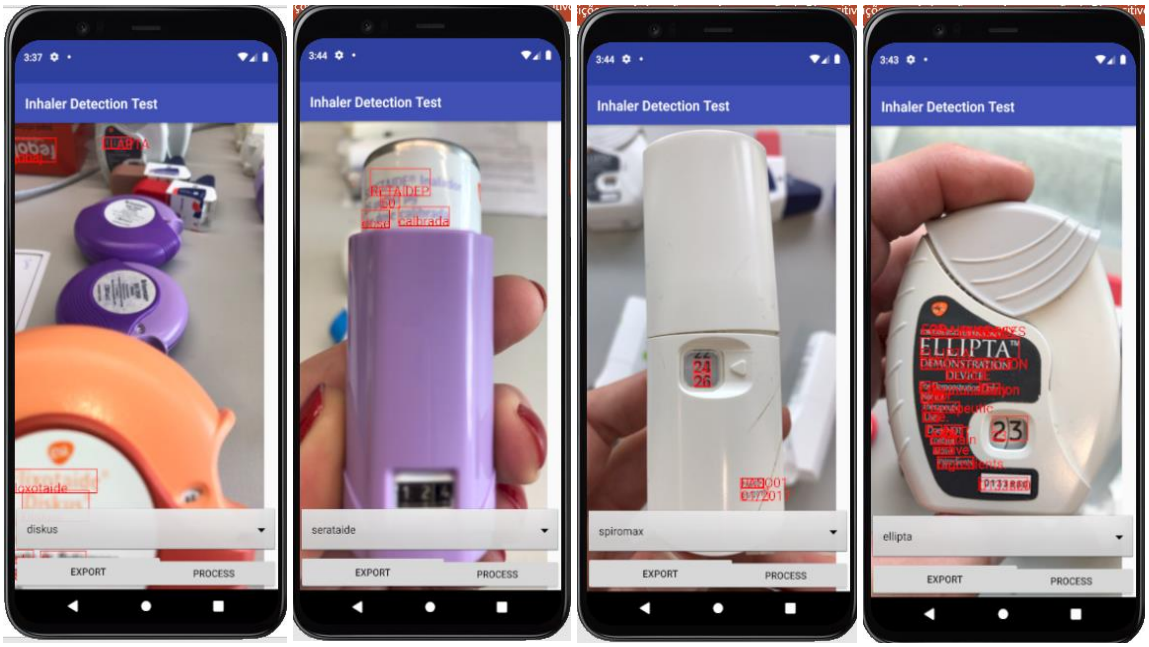


Figure 16 - Outputs of the preliminary phase of testing.

4.2. Results with the Default ML Kit's Model over iOS Image Module Dataset

To evaluate the performance of the text recognizer, a manual annotation of the text expected to be recognized by the app was performed. This way, all the results obtained were compared to the expected text, according to the success rate. The success rate is measurement corresponding to the number of correct results divided by all the images that were processed (Equation 7).

$$Success\ rate\ (\%) = \frac{Number\ of\ correct\ results}{Total\ number\ of\ images\ processed} \qquad (7)$$

To verify the impact of preprocessing on the quality of the system performance, data was collected on the text recognition app before cropping and afterwards (Table 10). This study was performed on the iOS Image Module dataset.

Table 10 - Text recognizer success rate before and after cropping in the iOS image module dataset.

Inhaler Models	**Number of Images (n)**	**Success Rate (%)** (before cropping)	**Success Rate (%)** (after cropping)
Diskus	229	49,34	77,83
Easyhaler	146	36,30	76,71
Ellipta	53	62,26	77,36
Flutiform	95	16,84	51,58
Khaler	232	12,5	24,14
Mdi3m	79	79,74	86,08
Nexthaler	62	40,32	96,77
Novolizer	82	80,48	92,68
Seretaide	152	19,10	38,16
Spiromax	47	31,91	55,32
Turbohaler	69	28,99	81,16
Twisthaler	76	2,63	9,21

Similarly, to study the effect of post-processing on the output quality, the results obtained before and after post-processing were compared. As such, a pre-processed iOS image module dataset was used for this analysis.

As mentioned previously, the results underwent three stages of post-processing: the removal of invalid characters, the fitting into the admissibility range, and the voting system. In order to evaluate the influence of these methods on the outcomes, the results were analyzed for each post-processing step. In this regard, Table 11 presents the success rate of the algorithm before and after the removal of invalid characters. In Table 11, it's also possible to verify which percentage of the results were considered admissible after the exclusion of non-numerical character.

Table 11 - Success Rate before and after removing invalid characters, and admissibility of the results.

Inhaler Models	**Events (n)**	**Success Rate (%)** (before exclusion of invalid characters)	**Success Rate (%)** (after exclusion of invalid characters)	**Admissible Results (%)**
Diskus	229	77,83	80,43	66,52
Easyhaler	146	87,67	89,04	75,34
Ellipta	53	77,36	79,25	66,04
Flutiform	95	51,58	51,58	48,42
Khaler	232	24,14	24,14	64,66
Mdi3m	79	86,08	87,34	81,01
Nexthaler	62	96,77	96,77	74,19
Novolizer	82	92,68	93,90	68,29
Seretaide	152	38,16	38,16	25,66
Spiromax	47	55,32	57,45	48,94
Turbohaler	69	82,61	82,61	79,71
Twisthaler	76	9,21	9,21	34,21
Total	1322	64,95	65,82	61,10

It is important to note that only a percentage of the corrected results were considered admissible. Thus, the causes for a result to be considered inadmissible (image quality, default model malfunction, etc.) were investigated. As such, a study was performed on the iOS image module dataset, in which the images were inspected one by one to determine the cause of the algorithm's failure. In Table 12, the causes for inadmissibility can be verified according to the inhaler model. On the other hand, in Figure 17, the causes for inadmissibility can be viewed without distinguishing between inhaler types.

Table 12 – Number of admissible events and the causes for error in inadmissible results.

Inhaler Type	Causes for inadmissibility					Admissible	Total of Images
	Template Matching	Cropping	Blurring	Admissibility Range	Unknown		
Diskus	37	8	8	10	14	152	229
Easyhaler	22	3	7	1	3	110	146
Ellipta	0	9	0	0	9	35	53
Flutiform	1	13	4	2	29	46	95
K-haler	35	6	3	0	38	150	232
Mdi3m	12	0	2	0	1	64	79
Nexthaler	10	3	3	0	0	46	62
Novolizer	10	8	3	1	4	56	82
Seretaide	33	1	21	1	57	39	152
Spiromax	9	0	0	0	15	23	47
Turbohaler	12	0	0	0	2	55	69
Twisthaler	15	0	0	0	35	26	76

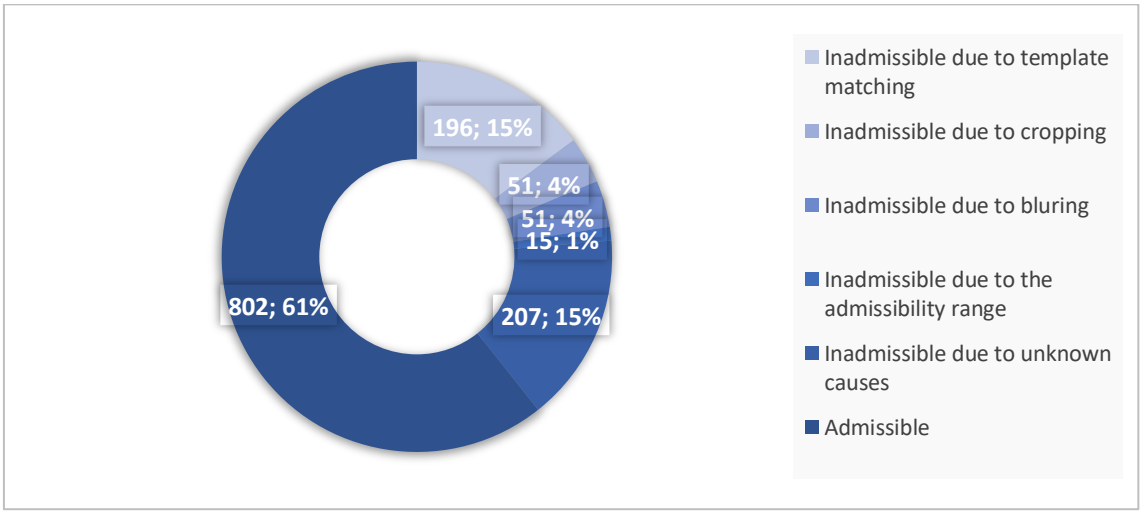


Figure 17 - Distribution of the causes for inadmissibility and admissible results.

Thereby, Table 13 shows the success rate of the algorithm only on results considered admissible.

Table 13 - Success rate of the algorithm considering only admissible and successful results.

Inhaler Models	**Number of Admissible Results**	**Number of Successful and Admissible Results**	**Success Rate (%)**
Diskus	153	147	96,07
Easyhaler	110	97	88,18
Ellipta	35	33	94,29
Flutiform	46	28	60,87
Khaler	150	13	8,67
Mdi3m	64	55	85,94
Nexthaler	46	45	97,83
Novolizer	56	56	100
Seretaide	39	4	10,26
Spiromax	23	23	100
Turbohaler	55	45	81,82
Twisthaler	26	0	0
Total	803	546	68,00

Finally, the results obtained after the implementation of the last post-processing step can be found in Table 14. As mentioned previously, the voting system considers five admissible frames of the same inhaler and saves the most common result of the text recognizer among them. In Table 14, the success rate after the deployment of this method can be found, as well as for the number of events, in which an event corresponds to the processing of five admissible frames.

Table 14 -Results of the inhaler detection module after applying the voting system.

Inhaler Models	**Events (n)**	**Success Rate (%)** (after applying the voting system)
Diskus	31	96,67
Easyhaler	21	90,47
Ellipta	6	100
Flutiform	9	77,78
Khaler	30	3,33
Mdi3m	12	100
Nexthaler	9	100
Novolizer	11	100
Seretaide	7	0
Spiromax	4	100
Turbohaler	11	81,82
Twisthaler	5	0
Total	156	69,65

In addition, it was also imperative to evaluate if the performance of the Android text recognizer is comparable to the performance of the iOS app. Therefore, Table 15 presents the success rates of both Android and iOS projects, when applied to the iOS module study. It's worth noting that all data from the iOS study went through similar pre and pos processing mechanisms to the ones used on the Android project. In addition, the results presented in Table 15 are only from images where template matching was successful.

Table 15 – Success rate of the text recognizer on the iOS app *versus* the Android app.

Inhaler Models	**Success Rate (%)** iOS version	**Success Rate (%)** Android version
Diskus	85,64	85,08
Easyhaler	87,50	89,17
Ellipta	77,27	79,55
Flutiform	62,16	41,89
Khaler	46,33	10,17
Mdi3m	83,61	83,61
Nexthaler	90,38	96,15
Novolizer	92,06	92,06
Seretaide	5,13	27,35
Spiromax	100	61,11
Turbohaler	94,44	79,63
Twisthaler	0	0
TOTAL	68,71	62,15

4.3. Results with the Default ML Kit's Model over the Real-World Users Dataset

A second database was also studied in an effort to analyze the quality of the results under suboptimal conditions. All images from this dataset have undergone all stages of pre-processing. However, due to the limitations of the number of available photos, the voting system step of post-processing was not performed. The results from this analysis can be found in Table 16.

As it can be seen in Table 16, there are 4 inhaler models whose results were deemed inadmissible. In fact, a significant percentage of the results (30 %) weren't considered admissible due to problems related to the image, more specifically: incorrectly performed cropping.

Table 16 – Success rate of the text recognizer in images acquired under less than ideal conditions by app users.

Inhaler Models	**Number of images**	**Admissible Results (%)**	**Success Rate (%)** among admissible results
Diskus	3	0	Non-applicable
Easyhaler	4	50,00	0
Ellipta	7	57,14	100
Flutiform	2	0	Non-applicable
K-haler	4	0	Non-applicable
Nexthaler	2	0	Non-applicable
Novolizer	4	50,00	100
Seretaide	4	75,00	0
Spiromax	8	37,50	33,33
Turbohaler	12	16,67	50
TOTAL	50	28,63	47,22

4.4. Results with the Default ML Kit's Model over the Controlled Dataset

The results in Table 17 were acquired by applying the text recognizer on the controlled dataset. The study of these results will allow us to understand if the model of the smartphone with which the photo was acquired may interfere with the quality of the results. Given the scarcity of images in this database the voting system method of pos-processing was not applied.

Table 17 – Success rate of the text recognizer among different devices.

			Inhaler Model				
			Diskus	Novolizer	Easyhaler	K-Haler	Total
Smartphone Model	Lenovo TB-7504F	**Number of Images**	4	2	5	2	13
		Admissible Results (%)	0	0	0	0	0
		Success Rate (%)	0	0	0	0	0
	Huawei P8 Lite	**Number of Images**	2	3	3	2	10
		Admissible Results (%)	0	100	100	0	50,00
		Success Rate (%)	0	100	66,67	0	91,67
	Redmi Note 8T	**Number of Images**	---	2	1	1	4
		Admissible Results (%)	---	100	100	0	66,67
		Success Rate (%)	---	50,00	100	0	50,00

Due to the lack of admissible results in some devices, the possible causes for these outcomes were studied. The findings of this analysis are shown in the graph in Figure 18.

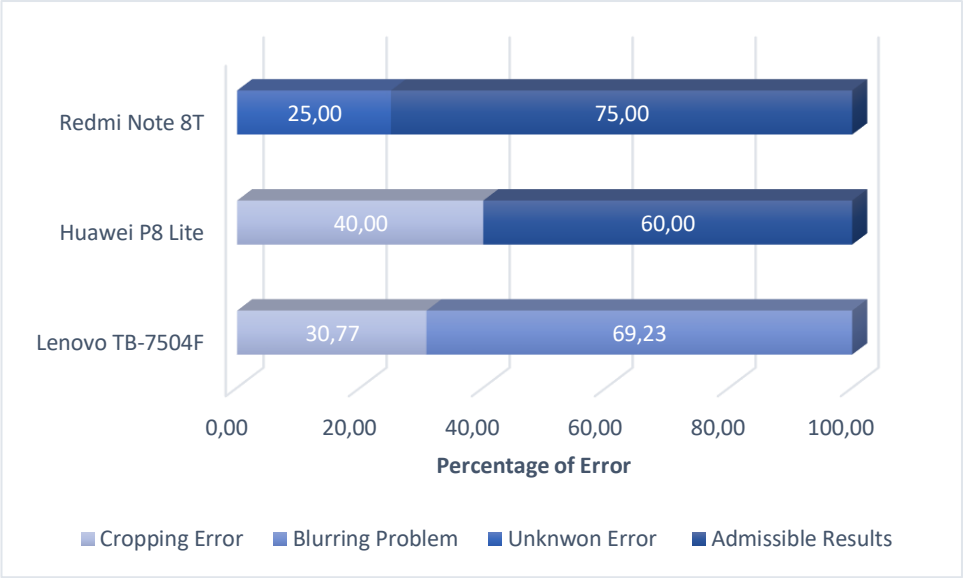


Figure 18 - Causes for error in the controlled dataset.

4.5. Trained Model

As mentioned previously, this project runs using the Google Colab environment and the network was built using TensorFlow 2. 6.. The model was trained in 200 epochs and a batch size of 5.

A sample of the outputs of the model (predicted text) on the validation dataset are presented in Figure 19, along with the input images. Moreover, as it can be seen in Figure 19, if the output string is less than 3 characters, the model predicts an unknow character ([UNK]).

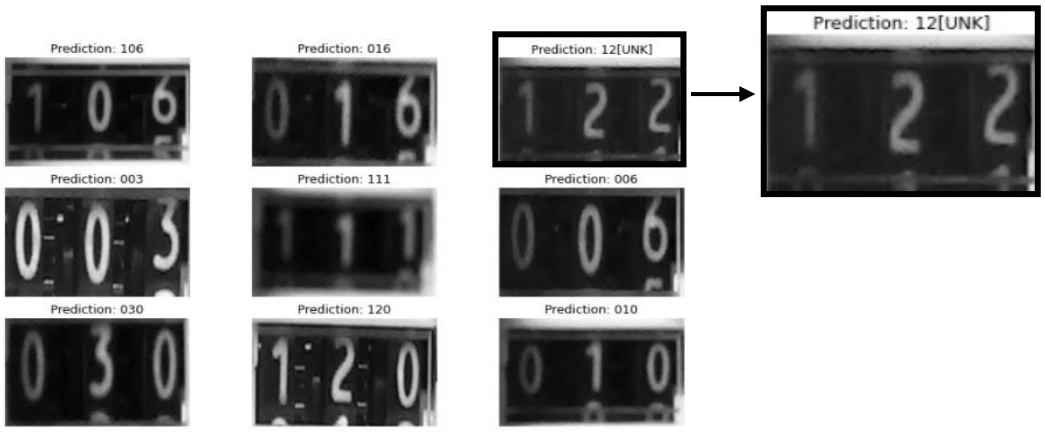


Figure 19 - Sample of the predicted text of the trained model, accompanied by the original images of the validation dataset.

4.5.1. Losses

During the training of a machine learning model, the current state of the model at each step of the training algorithm can be evaluated. In this case, the metrics training loss and validation loss were used to evaluate the model over time (Figure 20). This measures how much a model's predictions are off the desired label. The training loss indicates how well the model is fitting the training data ("learning"), while the validation loss indicates how well the model fits new data ("generalizing") [58].

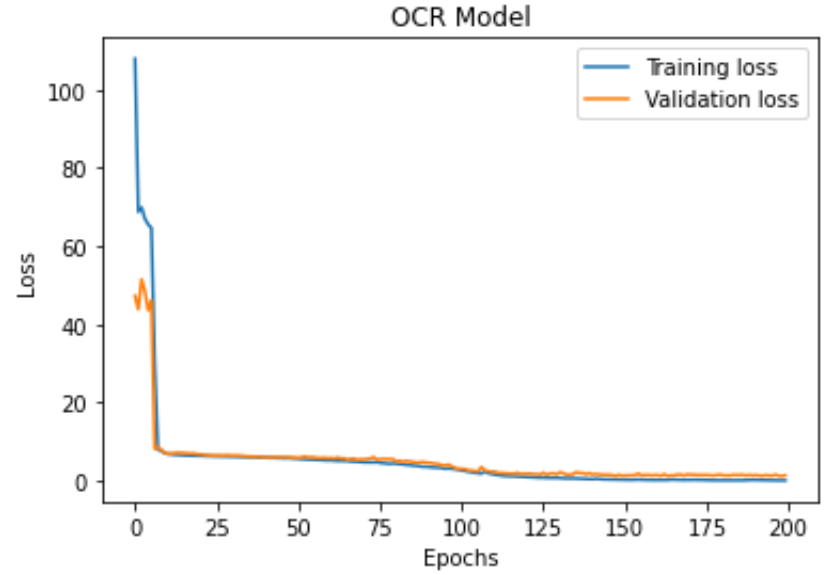


Figure 20 - Training and Validation Loss over time.

4.5.2. Evaluation Metrics

To evaluate the model, several metrics were taken into consideration to assess the model's performance (Table 18).

Table 18 - Evaluation Metrics for the Trained Model.

Metrics	Results
Exact Match Ratio	83,34 %
Hamming Loss	0,17 %
Recall	84,72 %
Precision	83,34 %
F1-measure	83,34 %

To evaluate the model's accuracy, the model was tested in 5 different training and validation datasets of the same size (Table 19). Recall that all the images used belonged to the *seretaide* dataset.

Table 19 – Validation accuracy for different training and validation datasets.

	Test					**Average**	**Standard Deviation**
	1	2	3	4	5		
Accuracy (%)	80,56	94,44	83,33	94,44	91,67	88,89	6,51

CHAPTER 5 – DISCUSSION

5. Discussion

In this thesis, a study designed to validate the inhaler dosage counter value identification module through an Android solution is introduced and the results presented. The module allows for an objective assessment of inhaler usage, which can enable and facilitate remote monitoring of patient adherence by a healthcare professional. A broad set of inhaler devices, with heterogeneous forms and dosages, were used for testing purposes.

In this chapter, the results of all performed experiments are summed up and discussed. First, the results of preprocessing are discussed. Followed by the outcomes of the performed OCR in the several datasets with MLKit default model, and finally the results of the trained model are discussed. Each step is briefly explained, followed by a summary of the results for each topic.

5.1. Pre-processing

During the processing of the images, it was possible to conclude that the ML Kit algorithm was not able to recognize the numbers contained in the dose counter when the picture was positioned horizontally. As such, the image rotation proved to be one of the most important preprocessing steps.

Among all the preprocessing steps, only cropping the region of interest was tested separately. The ML Kit algorithm can recognize text in images, however this entailed that the preliminary results presented all text captured in the image and not only the numbers in the dose counter. Thus, cropping the region of interest proved to be a critical step in noise removal, and without which it would not be possible to achieve an objective dosage value.

As it can be seen in Table 11, the cropping of the region of interest greatly impacted the results of the success of the algorithm to detect correctly the number on the dose counter. The inhaler models whose results seemed to benefit more from the cropping step were the *Nexthaler*, *Turbohaler* and *Easyhaler*.

5.2. ML Kit Text Recognition

5.2.1. Results from the iOS image module dataset

Regarding the iOS image module dataset, the exclusion of non-numerical characters seemed to slightly improve the algorithm’s performance in some inhaler

models. However, the addition of this pre-processing step did not have a considerable impact on the overall results, since the success rate before and after removing non-numerical characters differs only 0,9 % (Table 12).

The results of the ML Kit algorithm can be unsuccessful for several reasons: the number detected may be wrong, the result may contain more characters than those contained in the dose counter, or it may simply not find the number to detect. Some of these errors can be corrected by post-processing the results, however, others may derive from errors made in previous steps.

After the inspection of the images used for text extraction, it was possible to identify some of reasons for results to be deemed inadmissible (Figure 15). Three of the causes were due to the inadmissibility of the image: template matching, cropping errors and blurring. The images in the iOS module study dataset were acquired through the use of the app InspirersMundi, meaning the images went through a template matching process to guarantee that the inhaler was on the right position in the moment of the image capture. However, not all images were successful in the template matching step, and consequently not all inhalers were in the correct position during the cropping of the region of interest. Thus, a small percentage of the images (15 %) used did not frame the dose counter of the inhalers and were considered inadmissible due to template matching errors. On the other hand, there were some images where the template matching was successful, but the cropping was still poorly performed. This may be due to the measurements used for the cropping of the region of interest that despite working for most images, cannot be generalized for all cases. Moreover, in cases where the image was too blurred for the algorithm to recognize any digit, the error occurred during the photo acquisition and the images were considered inadmissible due to blurring.

However, a small percentage of inadmissible results are due to unknown causes of error (15 %), as visual verification of the cropped image does not allow to identify an evident problem. The most likely explanation for these unknown causes of error is the failure of the MLKit algorithm to detect the numbers, as the default ML Kit's Model may be insufficient.

The *Seretaide* model has one of the lowest admissibility rates, this is mainly due to unknown motives (37,5 %), although template matching errors (21,71 %) and out-of-focus pictures (13,81 %) also occur. One of the unknown causes contributing to the low admissibility rate, might be a problem with the ML Kit library recognizing the type of font present in the counter. Another possible explanation for so many inadmissible results,

is the small size and deep dosage counter in the *Seretaide* model, which creates additional difficulties.

Moreover, although the *diskus* inhaler model had a relatively high rate of admissible results, it was, among all the models, the one with the highest number of inadmissible images due to dose counter framing errors in the template matching phase. In fact, around one quarter of the images were found to be inadmissible for this reason (24, 34 %).

Additionally, the *K-haler* and *Twisthaler* inhaler models also had some of the highest number of events of picture inadmissibility due to unknown causes, with 16,38 % and 46,05 % of their images considered inadmissible for this reason, respectively. On closer inspection of these images, it's possible to discern that the orientation of the digits in the dose counters of these inhaler models is arranged vertically, whereas on other inhaler types the values are displayed horizontally. However, there is nothing in the ML Kit's documentation that may explain this discrepancy, since it clearly states that the model should be able to detect a contiguous set of words no matter the axis. Therefore, the problem may reside in the angle in which the numbers of the photo are lined, making it hard for the default model to detect them at all.

It was also observed that the text recognizer can be sensitive to strong reflexes or shadows that are able to obscure the numbers. These effects prevent the acquisition of an image of enough quality to allow text detection, even for human eyes. Such is the case for the *Flutiform* inhaler model, where it was found that many of the failed results are due to light reflected in the dose counter display. In fact, flashes of light prevented the detection of one digit in most cases and in some instances the light completely obstructed the vision of the dose counter. These observations were perceived during the image processing phase.

Among the results that were deemed admissible, in 68 % of the events it was possible to obtain a valid identification of the numeric values in the dosage counter. The inhaler models with higher rates of success were the *Novolizer*, *Spiromax*, *Nexthaler*, *Diskus* and *Ellipta* (all above 90%; Table 13). The inhaler model with the worst results is the *Twisthaler*, whose success rate is 0% among results considered admissible (Table 13). Nevertheless, *twisthaler* is rarely used in clinical practice, so the impact of the performance of the tool for this type of inhaler, in clinical practice, will be minimal [13]. Overall, by considering only the admissible images, the success rate of the algorithm improved by 2,2 %.

These results are in line with the findings of the iOS study [13], which indicates that the Diskus and Spiromax inhaler models are among the best performing inhalers. Similarly, the twisthaler inhaler model is also among the worst performing inhalers according to the InspiresMundi study.

The goal of the developed voting system is to make the algorithm less prone to errors. By collecting five frames instead of one, even if the algorithm does not on occasion predict correctly the numbers displayed in the dose counter, it is still possible to correctly identify the values form the majority. The number of frames to collect (5) and was chosen in accordance with the previously developed system in iOS by the Inspirers team. Although the number of frames could be higher or lower, five frames seemed to suffice the needs of the developed work.

Compared with the initial tests, the voting system phase performance improved the results for most inhalers. Nevertheless, this was found to be untrue for the inhaler model *Seretaide*. The cause for this disfavoring of the results was due to failures to correctly detect the text in the first place, i.e. the ML Kit failed to correctly recognize the numbers in the individual frames of the image, so the most voted text among the five frames will not match the expected text.

When comparing the performance of the ML Kit engine in the iOS (68.71 %) and Android (62,15 %) version of the image module on the same image subset, there is a difference of approximately 6,56 % among the success rates of the algorithm (Table 14). Although an attempt was made to implement all the pre-processing and post-processing steps met in the study performed on the iOS version [59], the differences between the base functions and algorithms in Android Studio and Swift may have contributed to this discrepancy.

5.2.2. Results from the Real-World Users Dataset

Furthermore, it’s possible to observe a contrast between the rate of admissible results of the iOS Image module dataset and of the Real-World Users dataset. By considering only the inhaler models present in both datasets, the rate of admissible results is, respectively, 61 % and 29 % (approximately).

This discrepancy may be due to the conditions under which the datasets were acquired: while the iOS Image module dataset was attained in a study with volunteers (sub-optimal conditions), the real-world user’s dataset was acquired by patients that use the iOS app in their daily life (less than ideal conditions). It’s also important to note that

the sizes of the dataset are very distinct in size (the images range from 50 to 1322 among the datasets) which may affect the differences among the rates of admissible. In addition, the heterogeneity of the equipment used in the real-world user's dataset could be one of the factors for the low admissibility rate in this dataset.

5.2.3. Results from the Controlled Dataset

Additionally, the images that have been acquired in a more controlled environment had more end to end results. Nevertheless, this was to be expected considering the small size of the dataset (27 images). All images acquired with the Lenovo TB-7504F were found to be inadmissible. This is justified by the fact that the expected result was to not detect text, since the quality of the image wouldn't allow it. Around 30% of the pictures the cropping did not frame the dose counter, and nearly 70% of the pictures were too blurry to detect any text (Figure 18). In relation to the incorrect cropping, this was only confirmed for an inhaler model: *Diskus*; thus, after verifying the images it was shown that, the dose counter was on the opposite side of the image. Hence, the cropping error derived from a previous one: the improper positioning of the inhaler model *Diskus*. After a closer inspection, it was found that half the images were not successful in the template matching phase. However, even the images were the template matching had been successful, were found in an incorrect position. This was due to the initial orientation of the image. While all images used so far were acquired by a smartphone and consequently were captured in a vertical position, the images acquired from the tablet were acquired in a horizontal position. This variation at the time of the image capture, invalidated the pre-processing steps performed, i.e. the rotation 90º degrees to the right did not allowed for the correct position of the dose counter, during the cropping of the region of interest.

With regards to the high rate of blurred pictures, the explanation may be the weaker resolution of the camera of this device (tablet), in comparison with the other two studied equipment (smartphones; Table 17).

5.3. Trained Model

5.3.1. Losses and Representativeness

As it can be seen in Figure 20, the training and validation loss decreases over time to a point of stability, achieving low error values. In addition, it's also possible to observe that the gap between the two final loss values is minimal. All the previously mentioned

aspects are indicators that the model has a good fit, and that continued training would likely lead to an overfit.

The curves on Figure 20 can also be used to diagnose properties of a dataset and whether it is relatively representative, i.e., if it reflects proportionally statistical characteristics in another dataset from the same domain. From the observation of the graph in Figure 20, the training dataset seems to be representative in relation to the validation dataset and vice-versa. Some of the signals that may indicate unrepresentative datasets are a large gap between the curves, noisy movements of the validation loss around the training loss and a validation loss that is lower than the training loss. None of these cases seems to apply for this instance.

5.3.2. Evaluation Metrics

The exact match ratio can be considered a challenging metric since it doesn't support the notion of being partially correct. In other words, this metric only considers the outputs as being correct if the whole sequence of characters corresponds to the ground truth. As it can be seen in Table 18, the exact match ratio is 83, 34%, which indicates that a large part of the predicted results were entirely correct, and consequently reflects a good model performance.

As mentioned in subsection 3.4.6., the Hamming Loss informs how many times on average, the relevance of an example to a class label is incorrectly predicted. Therefore, this metric considers the incorrect label predictions and the relevant labels not predicted, over the total number of labels. In this case, the computed hamming loss is 0,17 %, which is a significantly low value and indicates a good performance of the learning algorithm.

Recall metric quantifies the number of predicted correct labels made out of all the positive predictions that could have been made. Unlike precision that only comments on the correctly predicted labels out of all the positive predictions, the recall provides an indication of the missed positive predictions. For this model, the calculated recall was 84,74 % (Table 18), which indicates that a large number of the actual labels were predicted.

On the other hand, precision is the ratio of how much of the predicted is correct, i.e., it only considers the positive predicted results. In this case, the precision equals the exact match ratio (83,34 %).

Furthermore, the F1 measure is the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases than the Accuracy Metric. The F1 measure reaches 83,34 %, which is an indication of both good precision and good recall.

Finally, as previously mentioned, the model was tested in different training and validation datasets in order to evaluate the accuracy of the model under different circumstances. After inspecting the results on Table 19, it is possible to observe that the best performance of the model was 94,44% and that the average accuracy of the model is 88,89%. This method achieved reasonable results and the obtained results are a significant improvement in relation to the ML Kit default model performance. Thus, this method shows promise and can be implemented in other worst-performing inhaler models to enhance the text recognition performance.

5.4. Final Remarks

The inhalers models that seemed to perform best across all the datasets were *Diskus* and *Novolizer*, two of the most common inhalers in Portugal during 2016 [13]. Nevertheless, the datasets have different sizes and inhaler distributions meaning that the success of the algorithm may depend on that variation.

Furthermore, the developed self-contained module allows for easy integration into other applications with the same goal: an objective measure of adherence to inhaler medications.

Regarding the trained model, in spite of the good performance (average validation accuracy of 88,89%), the model only works in a narrow cropping of the inhaler dose counter. This aspect of the model is not ideal, since for the model to work the dose counter would have to be cropped with precision, which is unrealistic for real life applications. Thus, improvements on the custom model are necessary in order to be suited for the app functions.

Althought, the frequency of distribution of the characters does not seem to insert bias into the model, a larger dataset with more variety of characters could achieve better results. In addition, the model was trained in images acquired under optimal conditions, which means that the model may not have the performance on photos captured in the less-than-ideal environments (e.g. poor lighting). As such, in future works the model should be trained in a larger dataset with pictures acquired under a wider variety of scenarios.

Moreover, the deep learning model can be easily translated across datasets and could be applied in other worst-performing inhaler models to improve the text recognition performance.

In future work, the TensorFlow Lite Model could be integrated in the inhaler detection model of the app.

CHAPTER 6 - CONCLUSION

6. Conclusion

The purpose of this thesis was to develop a mobile application module that could detect the dose counter digits, in commercially available inhalers with numerical counters. This goal was met by creating a text detection module on Android, equipped with machine learning capabilities, as well as pre and post processing features. The work was taken further by improving the performance of the text recognizer on a worst performing inhaler. This was done by building a machine learning model, trained on a database of inhaler images, compatible with mobile applications. In addition, the new datasets collected for this work could be used in future research.

The method of image recognition used in this implementation proved to be practicable and promising when it came to obtain additional evidence to monitor adherence to inhaled medicines. Moreover, it provides patients with the tools to self-manage the treatment and promote compliance with therapeutic plan.

Although this approach is revealed to be feasible and promising for acquiring additional data that can be easily shared with the health professional remotely, it's still dependent on patient adherence to the application and the recording of inhalers. Nonetheless, this system has the potential to assist in a smoother transition from a health professional supported scenario to a more empowered self-management setting.

To the best of my knowledge, there are not many approaches in the literature that help to reduce the unreliability of patient compliance and self-reporting by making use of mobile devices to record effective dosage in inhaler dose counters. Furthermore, the proposed work explores the potential of mobile devices without external devices or expensive electronic monitoring devices, thus making this work relevant to help mitigating the patient's unreliable, self-reported adherence.

Nevertheless, further improvements are still needed to enhance the detection performance. In future work, an object detector-like algorithm can be implemented to detect the dose counter, thus avoiding the cropping stage in the image pre-processing. Additionally, the trained models can be improved and integrated in the current version of the app.

References

[1] J. E. Fergeson, S. S. Patel, and R. F. Lockey, "Acute asthma, prognosis, and treatment," *J. Allergy Clin. Immunol.*, vol. 139, no. 2, pp. 438–447, 2017, doi: 10.1016/j.jaci.2016.06.054.

[2] J. P. Barbosa, M. Ferreira-Magalhães, A. Sá-Sousa, L. F. Azevedo, and J. A. Fonseca, "Cost of asthma in Portuguese adults: A population-based, cost-of-illness study," *Rev. Port. Pneumol. (English Ed.)*, vol. 23, no. 6, pp. 323–330, Nov. 2017, doi: 10.1016/j.rppnen.2017.07.003.

[3] R. Jeminiwa, L. Hohmann, J. Qian, K. Garza, R. Hansen, and B. I. Fox, "Impact of eHealth on medication adherence among patients with asthma: A systematic review and meta-analysis," *Respir. Med.*, vol. 149, no. February, pp. 59–68, 2019, doi: 10.1016/j.rmed.2019.02.011.

[4] B. G. Bender, "Technology Interventions for Nonadherence: New Approaches to an Old Problem," *J. Allergy Clin. Immunol. Pract.*, vol. 6, no. 3, pp. 794–800, 2018, doi: 10.1016/j.jaip.2017.10.029.

[5] D. A. Fedele *et al.*, "Applying Interactive Mobile health to Asthma Care in Teens (AIM2ACT): Development and design of a randomized controlled trial," *Contemp. Clin. Trials*, vol. 64, no. March, pp. 230–237, 2018, doi: 10.1016/j.cct.2017.09.007.

[6] A. H. Y. Chan, H. K. Reddel, A. Apter, M. Eakin, K. Riekert, and J. M. Foster, "Adherence Monitoring and E-Health: How Clinicians and Researchers Can Use Technology to Promote Inhaler Adherence for Asthma," *J. Allergy Clin. Immunol. Pract.*, vol. 1, no. 5, pp. 446–454, 2013, doi: 10.1016/j.jaip.2013.06.015.

[7] C. Jácome *et al.*, "Patient-physician discordance in assessment of adherence to inhaled controller medication: a cross-sectional analysis of two cohorts," *BMJ Open*, vol. 9, no. 11, p. e031732, Nov. 2019, doi: 10.1136/BMJOPEN-2019-031732.

[8] E. Finnegan, M. Villarroel, C. Velardo, and L. Tarassenko, "Automated method for detecting and reading seven-segment digits from images of blood glucose metres and blood pressure monitors," *J. Med. Eng. Technol.*, vol. 43, no. 6, pp. 341–355, 2019, doi: 10.1080/03091902.2019.1673844.

[9] V. N. Shenoy and O. O. Aalami, "Utilizing smartphone-based machine learning in medical monitor data collection: Seven segment digit recognition," *arXiv*, pp. 1564–1570, 2018.

[10] H. Jiang, T. Gonnot, W. J. Yi, and J. Saniie, "Computer vision and text recognition for assisting visually impaired people using Android smartphone," *IEEE Int. Conf. Electro Inf. Technol.*, pp. 350–353, 2017, doi: 10.1109/EIT.2017.8053384.

[11] C. Jácome *et al.*, "Inspirers: An app to measure and improve adherence to inhaled treatment," *Proc. Int. Conf. E-Health, EH 2017 - Part Multi Conf. Comput. Sci. Inf. Syst. 2017*, pp. 135–139, 2017.

[12] Cristina *et al.*, "Feasibility and Acceptability of an Asthma App to Monitor Medication Adherence: Mixed Methods Study," *JMIR Mhealth Uhealth*

2021;9(5)e26442 https//mhealth.jmir.org/2021/5/e26442, vol. 9, no. 5, p. e26442, May 2021, doi: 10.2196/26442.

[13] P. Vieira-Marques *et al.*, "InspirerMundi—Remote Monitoring of Inhaled Medication Adherence through Objective Verification Based on Combined Image Processing Techniques," *Methods Inf. Med.*, vol. 60, no. S 01, pp. e9–e19, Apr. 2021, doi: 10.1055/S-0041-1726277.

[14] M. Zhang, A. Joshi, R. Kadmawala, K. Dantu, S. Poduri, and G. S. Sukhatme, "OCRdroid: A framework to digitize text using mobile phones," *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng.*, vol. 35 LNICST, pp. 273–292, 2010, doi: 10.1007/978-3-642-12607-9_18.

[15] A. Chaudhuri, K. Mandaviya, P. Badelia, and S. K. Ghosh, *Optical character recognition systems*, vol. 352. 2017.

[16] K. Kanagarathinam and K. Sekar, "Text detection and recognition in raw image dataset of seven segment digital energy meter display," *Energy Reports*, vol. 5, pp. 842–852, 2019, doi: 10.1016/j.egyr.2019.07.004.

[17] S. Ramiah, T. Y. Liong, and M. Jayabalan, "Detecting text based image with optical character recognition for English translation and speech using Android," *2015 IEEE Student Conf. Res. Dev. SCOReD 2015*, pp. 272–277, 2015, doi: 10.1109/SCORED.2015.7449339.

[18] K. Hamad and M. Kaya, "A Detailed Analysis of Optical Character Recognition Technology," *Int. J. Appl. Math. Electron. Comput.*, vol. 4, no. Special Issue-1, pp. 244–244, 2016, doi: 10.18100/ijamec.270374.

[19] H. El Bahi and A. Zatni, "Text recognition in document images obtained by a smartphone based on deep convolutional and recurrent neural network," *Multimed. Tools Appl.*, vol. 78, no. 18, pp. 26453–26481, 2019, doi: 10.1007/s11042-019-07855-z.

[20] N. Reddy Soora and P. S. Deshpande, "Review of Feature Extraction Techniques for Character Recognition," *IETE J. Res.*, vol. 64, no. 2, pp. 280–295, 2018, doi: 10.1080/03772063.2017.1351323.

[21] N. Islam, Z. Islam, and N. Noor, "A survey on optical character recognition system," *arXiv*. 2017.

[22] D. González Verdugo, "OCR on Android, optical character recognition: Tesseract," Feb. 26, 2017. https://solidgeargroup.com/en/ocr-on-android/ (accessed Nov. 29, 2020).

[23] Microsoft, "Optical Character Recognition (OCR) - Computer Vision - Azure Cognitive Services | Microsoft Docs," Nov. 08, 2020. https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-recognizing-text (accessed Nov. 29, 2020).

[24] ABBYY, "ABBYY Mobile Document Capture and Real-Time Recognition SDK," 2020. https://www.abbyy.com/mobile-capture-sdk/ (accessed Nov. 29, 2020).

[25] Dynamsoft, "Document scanning API for Mobile | Dynamsoft Camera SDK," 2020. https://www.dynamsoft.com/Products/dynamsoft-webcam-sdk.aspx (accessed Nov. 29, 2020).

[26] Anyline, “Anyline Mobile Scanning Technology for Mobile Data Capture | Anyline.” https://anyline.com/technology/ (accessed Nov. 29, 2020).

[27] “ML Kit | Google Developers.” https://developers.google.com/ml-kit (accessed Nov. 29, 2020).

[28] V. Kuprenko, “ML Kit for Firebase: features, capabilities, pros and cons | by Vitaly Kuprenko | Towards Data Science,” Aug. 21, 2019. https://towardsdatascience.com/ml-kit-for-firebase-features-capabilities-pros-and-cons-a182b4299cc (accessed Nov. 29, 2020).

[29] A. Abdulkader and M. R. Casey, “Low cost correction of OCR errors using learning in a multi-engine environment,” *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, pp. 576–580, 2009, doi: 10.1109/ICDAR.2009.242.

[30] C. S. Yang and Y. H. Yang, “Improved local binary pattern for real scene optical character recognition,” *Pattern Recognit. Lett.*, vol. 100, pp. 14–21, 2017, doi: 10.1016/j.patrec.2017.08.005.

[31] D. Tsiktsiris, K. Kechagias, M. Dasygenis, and P. Angelidis, “Accelerated Seven Segment Optical Character Recognition Algorithm,” *5th Panhellenic Conf. Electron. Telecommun. PACET 2019*, pp. 1–5, 2019, doi: 10.1109/PACET48583.2019.8956283.

[32] N. Sourvanos and G. Tsatiris, “Challenges in input preprocessing for mobile OCR applications: A realistic testing scenario,” *2018 9th Int. Conf. Information, Intell. Syst. Appl. IISA 2018*, 2019, doi: 10.1109/IISA.2018.8633688.

[33] X. Peng, H. Cao, S. Setlur, V. Govindaraju, and P. Natarajan, “Multilingual OCR research and applications: An overview,” *ACM Int. Conf. Proceeding Ser.*, no. ii, 2013, doi: 10.1145/2505377.2509977.

[34] N. Arica and F. T. Yarman-Vural, “An overview of character recognition focused on off-line handwriting,” *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 31, no. 2. pp. 216–233, May 2001, doi: 10.1109/5326.941845.

[35] A. Chaudhuri, K. Mandaviya, P. Badelia, and S. K Ghosh, *Optical Character Recognition Systems for Different Languages with Soft Computing*. 2017.

[36] I. Hesso, S. Nabhani Gebara, G. Greene, R. W. Co stello, and R. Kayyali, “A quantitative evaluation of adherence and inhalation technique among respiratory patients: An observational study using an electronic inhaler assessment device,” *Int. J. Clin. Pract.*, vol. 74, no. 2, Feb. 2020, doi: 10.1111/ijcp.13437.

[37] S. M. O’Dwyer *et al.*, “The effect of providing feedback on inhaler technique and adherence from an electronic audio recording device, INCA®, in a community pharmacy setting: Study protocol for a randomised controlled trial,” *Trials*, vol. 17, no. 1, p. 226, May 2016, doi: 10.1186/s13063-016-1362-9.

[38] S. Kagen and A. Garland, “Asthma and Allergy Mobile Apps in 2018,” *Current Allergy and Asthma Reports*, vol. 19, no. 1. 2019, doi: 10.1007/s11882-019-0840-z.

[39] B. E. Himes, L. Leszinsky, R. Walsh, H. Hepner, and A. C. Wu, “Mobile Health and Inhaler-Based Monitoring Devices for Asthma Management,” *J. Allergy Clin.*

Immunol. Pract., vol. 7, no. 8, pp. 2535–2543, Nov. 2019, doi: 10.1016/j.jaip.2019.08.034.

[40] B. Rusyn, O. Lutsyk, R. Kosarevych, and Y. Varetsky, "Automated Recognition of Numeric Display Based on Deep Learning," in *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019 - Proceedings*, Jul. 2019, pp. 244–247, doi: 10.1109/AIACT.2019.8847868.

[41] P. Tangtisanon, "Healthcare system for elders with automatic drug label detection," in *International Conference on Control, Automation and Systems*, Jan. 2016, vol. 0, pp. 666–670, doi: 10.1109/ICCAS.2016.7832390.

[42] S. Cakic, T. Popovic, S. Sandi, S. Krco, and A. Gazivoda, "The Use of Tesseract OCR Number Recognition for Food Tracking and Tracing," *2020 24th Int. Conf. Inf. Technol. IT 2020*, no. February, 2020, doi: 10.1109/IT48810.2020.9070558.

[43] R. P. Ghugardare, S. P. Narote, P. Mukherji, and P. M. Kulkarni, "Optical character recognition system for seven segment display images of measuring instruments," 2009, doi: 10.1109/TENCON.2009.5395994.

[44] G. Developers, "Lose It! uses ML Kit to extract data from nutrition labels and improve user experience," 2020. https://developers.google.com/ml-kit/case-studies/lose-it (accessed Jan. 03, 2021).

[45] G. Developers, "Text Recognition | ML Kit," 2021. https://developers.google.com/ml-kit/vision/text-recognition (accessed Oct. 01, 2021).

[46] A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," *MM 2019 - Proc. 27th ACM Int. Conf. Multimed.*, pp. 2276–2279, Oct. 2019, doi: 10.1145/3343031.3350535.

[47] G. Colaboratory, "Colaboratory | Frequently Asked Questions," 2021. https://research.google.com/colaboratory/faq.html (accessed Oct. 22, 2021).

[48] TensorFlow, "TensorFlow," 2021. https://www.tensorflow.org/ (accessed Oct. 22, 2021).

[49] H. Scheidl, "An Intuitive Explanation of Connectionist Temporal Classification," *Towards Data Science*, 2018. https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c (accessed Oct. 17, 2021).

[50] M. Yousef, K. F. Hussain, and U. S. Mohammed, "Accurate, data-efficient, unconstrained text recognition with convolutional neural networks," *Pattern Recognit.*, vol. 108, p. 107482, Dec. 2020, doi: 10.1016/J.PATCOG.2020.107482.

[51] T. Core, "Treinamento personalizado: passo a passo," 2021. https://www.tensorflow.org/tutorials/customization/custom_training_walkthrough#define_the_loss_and_gradient_function (accessed Oct. 24, 2021).

[52] PyTorch, "CTC Loss," 2019. https://pytorch.org/docs/stable/generated/torch.nn.CTCLoss.html (accessed Oct. 24, 2021).

[53] J. Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep

Learning," 2017. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (accessed Oct. 24, 2021).

[54] M. Sorower, "A literature survey on algorithms for multi-label learning," *Oregon State Univ. Corvallis*, 2010.

[55] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2004, vol. 3056, pp. 22–30, doi: 10.1007/978-3-540-24775-3_5.

[56] S. Developers, "Metrics and scoring: quantifying the quality of predictions," 2021. https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score (accessed Oct. 24, 2021).

[57] G. Developers, "Custom Models with ML Kit," 2021. https://developers.google.com/ml-kit/custom-models#train_your_own_image_classification_model (accessed Oct. 17, 2021).

[58] Baeldung on Computer Science, "Learning Curves in Machine Learning," 2020. https://www.baeldung.com/cs/learning-curve-ml (accessed Oct. 24, 2021).

[59] P. Vieira-Marques *et al.*, "Combined Image-Based Approach for Monitoring the Adherence to Inhaled Medications," in *IFMBE Proceedings*, Sep. 2020, vol. 76, pp. 1399–1404, doi: 10.1007/978-3-030-31635-8_171.

[60] S. C. Kalichman *et al.*, "A Simple Single Item Rating Scale to Measure Medication Adherence: Further Evidence for Convergent Validity," *J. Int. Assoc. Physicians AIDS Care (Chic).*, vol. 8, no. 6, p. 367, Nov. 2009, doi: 10.1177/1545109709352884.