



## End of studies dissertation

For obtaining the Bachelor's degree in Computer Science

# Automatic information recognition and extraction from medication labels

### *Submitted by :*

MR. BENRACHED Youcef  
MR. BOUAGADA Youcef  
MR. BELLOUTI Redhouane

### Board of jury members :

Mr. Ali Khalfi	President
Mr. Omar Boukadoum	Examiner
Mr. Riadh Meghatria	Supervisor
Dr. Imène Ait Abderrahim	Co-supervisor

Academic Year : 2022/2023

# Dedication

“

*To my parents and my family members,  
To all my dear friends and colleagues,  
To everyone who supported me and believed in me  
Thank you all and have a nice life.*

”

**- Karim**

# Dedication

“

*I dedicate this humble and modest work with great love,  
sincerity, and pride :*

*To my dear parents, a source of tenderness, nobility, and  
affection. May this milestone be a reason for your  
satisfaction,*

*To all the members of my family,*

*To all my colleagues and friends,*

*Thank you for being a part of my journey and for your  
continuous encouragement.*

”

**- Youcef**

# Dedication

“

*To all my family and my close friends especially my  
parents, Thank you very much for believing in me and  
supporting me in my life,*

*I wish you a happy , blessed life .*

”

**- Redhouane**

# Acknowledgment

*First, we thank ALLAH, the Almighty who has illuminated our path and who armed us with courage and patience to accomplish this work in the best conditions.*

*We extend our deepest appreciation to our supervisors, **Mr. Meghatria Riadh** and **Mrs. Ait Abderrahim Imène**, for their unwavering support and availability during the execution of this work. Their immense assistance, high-quality guidance, and valuable advice and information they generously shared with us are truly commendable.*

*We are truly grateful for their unparalleled patience and professionalism, which have greatly contributed to the success of this endeavor.*

*We also thank anyone who participated directly or indirectly in the execution of this modest work.*

*May the members of the jury find, here, the expression of our sincere thanks for the honor they do us by taking the time to read and evaluate this work.*

# *Abstract*

In the field of pharmacology, medication labels play a crucial role in providing essential information about the medicines being sold or distributed, including details such as lot numbers, fabrication and expiration dates, and other pertinent information. However, the absence of a standardized model for medical labels approved by the Minister of Health has resulted in a lack of automatic recognition systems for medical labels in Algeria. In this work, we propose an Artificial Intelligence (AI)-based system capable of automatically identifying medical labels, extracting their content, and classifying them as refundable or non-refundable medications.

The study introduces two approaches: the Tesseract-based approach and the VisionAI-based approach. The Tesseract-based approach involves traditional detection and segmentation techniques using classic image processing methods, such as horizontal and vertical histogram analysis, for the identification and segmentation of lines and sections within the medical labels. Additionally, PyTesseract, an Optical Character Recognition (OCR) tool, is utilized to extract text from the medical labels. On the other hand, the VisionAI-based approach incorporates a custom-trained medical label detection and segmentation model, which demonstrates precise and reliable outcomes. Furthermore, the approach leverages VisionAI, a highly accurate OCR engine, for text extraction from medical labels.

Experimental evaluations demonstrate the superior performance of the VisionAI-based approach, achieving a remarkable accuracy of approximately 95.79% in the detection and extraction of their content. This research presents a promising solution to the lack of automated recognition systems for medical labels in Algeria, providing a foundation for the implementation of efficient and reliable tools for medication management and classification.

**Keywords :** Artificial Intelligence, Medical labels, Text recognition, Detection, Segmentation, Classification, OCR, VisionAI, Tesseract.

# *Resume*

Dans le domaine de la pharmacologie, les étiquettes des médicaments ou aussi appelées vignettes jouent un rôle essentiel en fournissant des informations importantes sur les médicaments qui sont vendus ou distribués, notamment des détails tels que les numéros de lot, les dates de fabrication et d'expiration, ainsi que d'autres informations pertinentes sur les médicaments. Cependant, l'absence d'un modèle normalisé pour les étiquettes médicales approuvé par le Ministère de la Santé a conduit à l'absence de systèmes de reconnaissance automatique des étiquettes médicales en Algérie. Dans ce travail nous proposons un système basé sur l'intelligence artificielle capable d'identifier automatiquement les étiquettes médicales, d'extraire leur contenu et de les classer comme médicaments remboursables ou non remboursables.

Cette étude présente deux approches : l'approche basée sur Tesseract et l'approche basée sur VisionAI. L'approche basée sur Tesseract utilise des techniques traditionnelles de détection et de segmentation à l'aide de méthodes classiques de traitement d'images, telles que l'analyse des histogrammes horizontaux et verticaux, pour identifier et segmenter les lignes et les sections des étiquettes médicales. De plus, PyTesseract, un outil de reconnaissance optique de caractères (OCR), est utilisé pour extraire le texte des étiquettes médicales. D'autre part, l'approche basée sur VisionAI intègre un modèle de détection et de segmentation personnalisé pour les étiquettes médicales, qui présente des résultats précis et fiables. De plus, cette approche utilise VisionAI, un moteur OCR très précis, pour l'extraction du texte des étiquettes médicales.

Les évaluations expérimentales démontrent les performances supérieures de l'approche basée sur VisionAI, atteignant une précision remarquable d'environ 95,79% dans la détection des étiquettes médicales et l'extraction de leur contenu. Cette recherche présente une solution prometteuse au manque de systèmes de reconnaissance automatique des étiquettes médicales en Algérie, fournissant ainsi une base pour la mise en œuvre d'outils efficaces et fiables de gestion et de classification des médicaments. Des recherches supplémentaires sont nécessaires pour améliorer les capacités du système et élargir son applicabilité dans le domaine de la santé.

**Mots clé :** Intelligence Artificielle, Vignette de médicament, Reconnaissance de text, OCR, Tesseract, VisionAI, Détection, Segmentation, Classification.

## ملخص

في مجال الصيدلة، تلعب الملصقات الطبية دورًا حاسمًا في توفير المعلومات الأساسية حول الأدوية المباعة أو الموزعة، بما في ذلك تفاصيل مثل أرقام الدفعة وتواريخ الصنع والانتهاء، وغيرها من المعلومات المهمة حول الأدوية. ومع ذلك، فإن غياب نموذج قياسي للملصقات الطبية المعتمد والموافق عليه من قبل وزارة الصحة أدى إلى عدم وجود نظام تعرف آلي للملصقات الطبية في الجزائر. يقترح هذا المشروع نظامًا قائمًا على الذكاء الاصطناعي قادرًا على التعرف الآلي على الوصفات الطبية واستخلاص محتواها، وكذلك تصنيف هذه الأدوية إلى قابلة لإسترداد مبلغ الأدوية أم لا. يتم تقديم نهجين: نهج يعتمد على Tesseract ونهج يعتمد على VisionAI. ينطوي النهج الأول على بعض تقنيات الكشف والتجزئة التقليدية باستخدام تقنيات معالجة الصور التقليدية مثل التحليل الأفقي والعمودي للهستوغرام المستخدم للكشف والتجزئة للأسطر وأجزاء الملصقة الطبية، بالإضافة إلى استخدام أداة تعرف الحروف البصرية PyTesseract لتحديد واستخراج النص من الملصقات الطبية. بالنسبة للنهج الثاني، قمنا بإنشاء نموذج كشف مخصص للملصقات الطبية يعرض نتائج دقيقة وموثوقة، بالإضافة إلى استخدام أحد أكثر محركات التعرف البصرية الموثوقة التي تسمى VisionAI. أظهرت التجارب أن النهج الثاني كان أداءه جيدًا جدًا في اكتشاف الوصفات الطبية واستخراج محتواها بدقة تصل إلى حوالي 95.79%.

### الكلمات المفتاحية :

الذكاء الاصطناعي ، الملصقات الطبية ، الرسم البياني ، التعرف الضوئي على الحروف ، Tesseract ، VisionAI ، الكشف ، التجزئة، التصنيف



# Contents

<b>Dedication</b> . . . . .	<b>2</b>
<b>Dedication</b> . . . . .	<b>3</b>
<b>Dedication</b> . . . . .	<b>4</b>
<b>Acknowledgment</b> . . . . .	<b>I</b>
<b><i>Abstract</i></b> . . . . .	<b>II</b>
<b><i>Resume</i></b> . . . . .	<b>III</b>
<b>General Introduction</b> . . . . .	<b>1</b>
<b>1 Medical Label Information Processing</b> . . . . .	<b>3</b>
1.1 Introduction . . . . .	4
1.2 Information recognition from medical labels (vignettes) . . . . .	4
1.2.1 What is a medical label? . . . . .	4
1.2.2 Medical label use in Algerian Pharmacies . . . . .	5
1.2.3 Types of medical labels . . . . .	5
1.2.4 Medical label information content . . . . .	5
1.2.5 Utilization of medical label information . . . . .	6
1.3 Medical label information processing in Algerian pharmacies . . . . .	6
1.4 Automatic processing of medical label information . . . . .	6
1.5 How to process information from an image . . . . .	7
1.6 Recognition . . . . .	7
1.7 Types of text recognition . . . . .	8
1.7.1 Printed Text Recognition . . . . .	8
1.7.2 Handwriting Recognition . . . . .	8
1.8 Text recognition from Images . . . . .	8
1.8.1 Image recognition . . . . .	9
1.8.2 Image pre-processing . . . . .	9
1.8.3 Enhancement techniques . . . . .	11
1.8.4 Segmentation . . . . .	12
1.8.5 Segmentation Methodologies . . . . .	12
1.8.6 Segmentation Types . . . . .	13
1.9 Postprocessecing . . . . .	15
1.10 Conclusion . . . . .	15

<b>2</b>	<b>Existing Text Recognition Methods</b>	<b>16</b>
2.1	Introduction	17
2.2	Text recognition methods	17
2.3	OCR-Based Methods	17
2.3.1	Optical Character Recognition (OCR)	17
2.3.2	types of OCRs	17
2.4	AI-based methods	19
2.4.1	Machine Learning (ML) Methods	19
2.4.2	Deep Learning (DL) Methods	20
2.5	Related work	21
2.5.1	Language-free methods	21
2.5.2	Language-aware methods	21
2.5.3	Self-supervised Implicit Glyph Attention	22
2.6	Existing applications of Medical labels in Algeria	22
2.7	Contribution	22
2.8	Conclusion	22
<b>3</b>	<b>Conception and Implementation</b>	<b>23</b>
3.1	Introduction	24
3.2	Environment Setup	24
3.2.1	Python	24
3.2.2	Visual Studio Code	24
3.2.3	Python IDLE	24
3.2.4	Google Colab	24
3.3	Tools Used	25
3.3.1	OpenCV	25
3.3.2	PyTesseract	25
3.3.3	Ultralytics	25
3.3.4	YOLOv8	25
3.3.5	RoboFlow	25
3.3.6	Google Cloud Vision AI	26
3.4	Implementation	26
3.4.1	Dataset Collection	26
3.4.2	Tesseract-based Approach	28
3.4.3	VisionAI-based Approach	34
3.4.4	Graphical User Interface	47
3.5	Experimentation and Results	51
3.5.1	Segmentation results	51
3.5.2	Medical Label detection results	52
3.5.3	Medical label character recognition results	53
3.5.4	Differences and Comparison	55
3.6	Conclusion	56
	<b>Conclusion and perspectives</b>	<b>57</b>

# List of Figures

1.1	Example of a medical label(sticker)	4
1.2	Medication labels types	5
1.3	visual representation of the different types of OCR systems	9
1.4	Architecture for text recognition steps	10
1.5	(a) Input image (b) Filtered image using median filter showing only the center pixel	11
1.6	Original Text Based Image and its Histogram output	13
2.1	Architecture of Tesseract OCR [21]	18
3.1	Content of a medical label with 2 parts	27
3.2	Content of a medical label with 3 parts	28
3.3	Tesseract-based approach FlowChart	29
3.4	Medical label detection process	30
3.5	Gray-scaled Medical label	31
3.6	Segmentation process	32
3.7	Tesseract OCR results	32
3.8	VisionAI-based approach FlowChart	34
3.9	Dataset structure	35
3.10	Training process	35
3.11	Creation of a project in Roboflow	36
3.12	Image uploading	37
3.13	Image annotations	37
3.14	Augmentation of the Dataset	38
3.15	Dataset Exportation	38
3.16	Content of the YAML file	39
3.17	YOLO command	39
3.18	Training and Validation results	40
3.19	Metrics results	41
3.21	Service account key export	43
3.22	Authentication and verification of the service account	43
3.23	Model and Image loading	43
3.24	Results of the YOLO detection	44
3.25	The final results of the medical label detection	44
3.26	VisionAI-OCR results	45
3.27	The home page of the GUI	47
3.28	Upload Page	48
3.29	Results Page	49

3.30	Export Page . . . . .	50
3.31	Example of a good segmentation . . . . .	52
3.32	Example of a bad segmentation . . . . .	52
3.33	High Confidence score detection . . . . .	53
3.34	Example of a bad rotated medical label . . . . .	53
3.35	Comparison study of Accuracy . . . . .	54
3.36	OCR experiment results' comparison . . . . .	54

# Tables

3.1	OCR accuracy results in both approaches . . . . .	54
-----	---	----

# List of abbreviations and acronyms

<b>API</b>	<i>Application Programming Interface</i>
<b>AI</b>	<i>Artificial Intelligence</i>
<b>CNN</b>	<i>Convolutional Neural Network</i>
<b>CRNN</b>	<i>Convolutional Recurrent Neural Network</i>
<b>DL</b>	<i>Deep Learning</i>
<b>JSON</b>	<i>JavaScript Object Notation</i>
<b>LSTM</b>	<i>Long Short-Term Memory</i>
<b>ML</b>	<i>Machine Learning</i>
<b>OCR</b>	<i>Optical Character Recognition</i>
<b>RGB</b>	<i>Red Green Blue</i>
<b>YAML</b>	<i>YAML Ain't Markup Language</i>
<b>YOLO</b>	<i>You Only Look Once</i>
<b>HTR</b>	<i>Handwritten Text Recognition</i>
<b>QR</b>	<i>Quick Response</i>
<b>CNAS</b>	<i>Caisse Nationale Des Assurances Sociales</i>
<b>DIA</b>	<i>Document Image Analysis</i>
<b>NLP</b>	<i>Natural language processing</i>
<b>RNN</b>	<i>Recurrent Neural Network</i>
<b>HMMs</b>	<i>Hidden Markov Models</i>
<b>VGG</b>	<i>Visual Geometry Group</i>
<b>STR</b>	<i>Scene Text Recognition</i>
<b>GUI</b>	<i>Graphical User Interface</i>

# General Introduction

## Context

In the domain of pharmacology, medication labels play a vital role in conveying crucial information about the medicine being sold or distributed, including the lot number, fabrication and expiration date, name of the medical company, medicine name, price, and refundability. Pharmacies primarily use these labels to determine the medication's name and refundability, while insurance companies and CNAS rely on them to verify and store medication information in their databases.

However, in Algeria, the absence of an approved and universally accepted standard model for medical labels, endorsed by the Minister of Health, has resulted in the lack of an automatic recognition system for these labels. This challenge stems from the wide range of label variations, such as different fonts, letter sizes, and information placement and order.

Furthermore, although numerous object recognition applications and models are available, none of them has been tested to effectively detect and identify medical labels. This limitation arises due to the underutilization of medical labels in these systems.

## Problematic

A primary issue with medical labels lies in their underutilization for their intended purpose. They are predominantly employed to merely confirm the medication's name and refundability, while the remaining information is accessed by scanning the barcode on the medication, which also facilitates buying and selling. Moreover, entities such as CNAS rely on the manual processing of the text on medical labels to authenticate the information in their database, resulting in a time-consuming process.

## Motivation

One of the issues arising from the aforementioned shortcomings is the underutilization of medical labels in Algeria, which necessitates the use of the Barcode method. To overcome this problem, there is an opportunity to leverage medical labels to automate the entire process for pharmacy workers by enabling automatic detection of their content.

# Objectives

In our project, we aimed to address the existing issues with medical labels by developing an automated system for the detection, segmentation, and extraction of medical label information. Our objective was to provide effective solutions for the challenges associated with medical labels.

# Dissertation organisation

This dissertation is organized into 3 chapters:

The first chapter will talk about the *Medical Label Information Processing*.

The second chapter will consist of the *Existing Text Recognition Methods*.

The third chapter represents our *Contribution and proposed approach for medication labels recognition and extraction problem*.

Finally, we conclude our work contribution and give a few perspectives to this work.



# Chapter 1

## Medical Label Information Processing

### 1.1 Introduction

The field of pharmacology (along with its industrial partners and those in public health represented by the CNAS uses medical labels (see Figure 1.1) for the commercialization and reimbursement of medicaments sold or distributed in pharmacies and hospitals, in Algeria. Medical labels (also called stickers) have a format that differs from the labels used for other commercial products, such as the barcode that can be scanned using a barcode reader and the QR code, which is widely used by smartphone applications and can be captured using a camera, these medicals labels that we can see are currently used in many pharmacies, however, there is currently no standard sticker model that is recognized by the Ministry of health and its partners in the pharmaceutical industry nor any systems for reading and recognizing stickers and their upstream processing (detecting whether refundable or not, reading the PPA price, TR, expiry date, etc.) and one of the problems that occurs in the mentioned lacks is that we are not using medical labels to its full potential in Algeria, which leads to the use of the Barcode method too while having a solution with the medical labels itself and also to exploit medical label in the goal of automating the whole process of workers using it in pharmacies by making the detection of it's content automatic, in our work, the idea is to propose a system able of reading and recognizing the medical labels information automatically. In this chapter, we will present what is a medical label and the different information that can be found in its content, then we will talk about its use and after that, we will give an explanation of how text information can be recognized and processed out of an image.

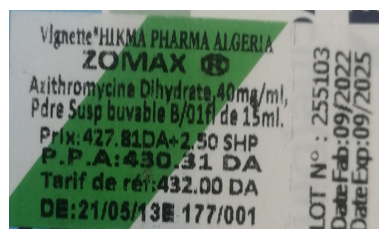


Figure 1.1: Example of a medical label(sticker)

### 1.2 Information recognition from medical labels (vignettes)

#### 1.2.1 What is a medical label?

A medication label serves as an essential resource in the healthcare industry that is providing crucial information about pharmaceutical products. It consists of a typically small piece of printed material with text content, usually affixed to the packaging of medications. The purpose of a medical label is to convey vital details about the medication, enabling healthcare professionals, pharmacists, and patients to make informed decisions regarding its use.

### 1.2.2 Medical label use in Algerian Pharmacies

In Algerian pharmacies, medical labels serve as a vital means of communication between pharmacists and patients. They are affixed to medication packages to ensure proper usage and to avoid any potential risks associated with incorrect administration.

### 1.2.3 Types of medical labels

In Algeria, most of the medications are generally labeled with one of three types of stickers, a red or a green or a white one.

- **The green sticker** : represents Refundable medications.
- **The red sticker** : represents Non-Refundable medications.
- **The white sticker** : represents non-refundable supplements or vitamins.



Figure 1.2: Medication labels types

### 1.2.4 Medical label information content

The medical label can actually contain many important information that are very useful for both pharmacists and patients, the information are like the following:

1. Name of the medication
2. Dosing
3. Dosage format
4. **PPA:** Algerian public price
5. **TR:** the price in the case of purchasing using a "Shiffa" card
6. **SHP:** Pharmacist Honorary Supplement
7. **Fab:** Date of manufacture
8. **Exp:** expiration date
9. **Lot:** the batch number

### 1.2.5 Utilization of medical label information

Information from medical labels is important in Algerian pharmacies and is used for more than just delivering information about pharmaceuticals. Information from medical labels is used in both the commercial and administrative facets of pharmacy operations.

Here are the key ways in which medical label information is utilized:

#### Commercial reasons

Pharmacists rely on medical label information to determine the price of medications, track inventory, and a way to communicate between pharmacists and patients to manage sales records.

#### Administrative reasons

Information on medical labels is essential for administrative tasks like complying with regulations, ensuring adequate storage conditions, and maintaining accurate records of medication dispensed.

## 1.3 Medical label information processing in Algerian pharmacies

Currently, we live in a world where most of data are digital and processed automatically, however, Algerian pharmacies are not fully utilizing the potential of medication labels in a digital sense. They merely use them to identify the medication names and rely on barcodes as a management system for selling medicines. The process of submitting medication information to the CNAS (The National Insurance Fund) involves using medication stickers, which are attached to the back of the prescriptions sheet. Then, a worker manually verifies the information on the medication labels by comparing it with the content in the delivery note (le bordereau).

This brings us to the question of how can we automate this whole process? from the detection of medication label content to the extraction of the final data to be compared with the delivery note. Therefore, achieving this would solve various problems and transform mostly the entire workflow of (pharmacies and the CNAS, the sale of medications, ...,ect) into an automated system.

## 1.4 Automatic processing of medical label information

In our technologically advanced society, information is stored, processed, organized, and retrieved by computer systems, reflecting the digital nature of our world.

The field of optical character recognition (OCR) has experienced significant advancements, despite these achievements, medication label recognition has not been solved. Due to the lack of technological tools for reading medication labels(stickers), our primary objective in this project was to develop an optical character recognition (OCR) system specifically designed to address the question from the previous section.

Thus, it is possible to automate this entire process by utilizing the medication stickers more effectively. Although we could automate this process by using the medicament sticker by exploiting the sticker in a better way, where the detection of its content is done automatically, then by segmenting the information contained within the sticker image, the content of the medication labels can be extracted and stored in a ".txt" file. This information can then be compared automatically with other files, including the delivery note (le bordereau). By automatizing these steps, we can streamline the process between pharmacies and the CNAS.

Our major contribution then consists in building an automatized system for the detection and recognition of medication labels.

### 1.5 How to process information from an image

Processing information from an image involves several essential steps to extract correct and meaningful text. Firstly, pre-processing techniques are applied to enhance image quality, reducing noise and adjusting contrast. Then, image segmentation is employed to isolate individual characters, lines, or paragraphs from the image. Next, character recognition algorithms analyze the segmented regions to identify and recognize the characters present. Subsequently, the recognized characters are grouped to form words and sentences through word and text recognition processes. Finally, post-processing techniques are applied to refine the recognized text, improving accuracy and correcting any errors. These steps collectively enable the conversion of text within an image into an editable and searchable format, unlocking the information contained within the image for further analysis and use [11]. In the following, we will be expanding in more detail on each step of this process.

### 1.6 Recognition

In the world of technology, recognition technologies use complex algorithms and techniques to recognize and understand environmental sensory information that is visual, aural, or other. These technologies frequently use computer vision, deep learning, and machine learning approaches to process and evaluate data and come to reliable recognition conclusions [11].

### 1.7 Types of text recognition

In recent times, there has been significant progress in the field of optical character recognition (OCR) irrespective of machines and computers.

OCR systems typically receive input in two main types of text: **handwritten text** or **machine-printed text** for recognition. In the past, OCR results were relatively predictable due to the consistent character styles and positions on the document page. However, modern OCR faces challenges with varying writing patterns and character sets in different languages, making the problem more complex.[8]

OCR systems can be classified into two categories: online and offline. Online OCR systems operate in real-time as users write characters, and they can also record writing speed. Therefore, they are generally less complex. In contrast, offline OCR systems process pre-existing data, which makes pattern recognition more complex. Online systems are more in demand due to their ability to provide more accurate results, ease of development, and compatibility with devices such as tablets and Figure 1 provides a visual representation of the different types of OCR systems [8].

#### 1.7.1 Printed Text Recognition

The most popular form of OCR writing recognition, known as printed text recognition, involves extracting printed text from scanned documents or images. In order to identify the text in the printed text, OCR typically involves examining the shapes and patterns of individual characters and words. In many different industries, including finance, health-care, and logistics, this kind of OCR is commonly utilized for document digitization, text extraction from invoices, receipts, and forms, and automated data entry [11].

#### 1.7.2 Handwriting Recognition

Handwriting recognition is a difficult aspect of OCR writing recognition since it requires evaluating the diverse and variable characters of handwritten text. Handwriting recognition OCR systems must account for diverse writing styles, shapes, and sizes of characters, making it more sophisticated than printed text recognition. Handwriting recognition OCR has applications in activities such as recognizing handwritten forms, converting handwritten notes to digital text, and signature verification in the financial and legal businesses.[8]

### 1.8 Text recognition from Images

The goal of text recognition in image research is to create a computer system that can automatically extract text from images. The demand to digitize information that is currently only available in paper documents and save it in computer systems for simple search-based retrieval is expanding in the contemporary digital age. An option that is frequently used is to scan paper documents and store the generated images as a way of

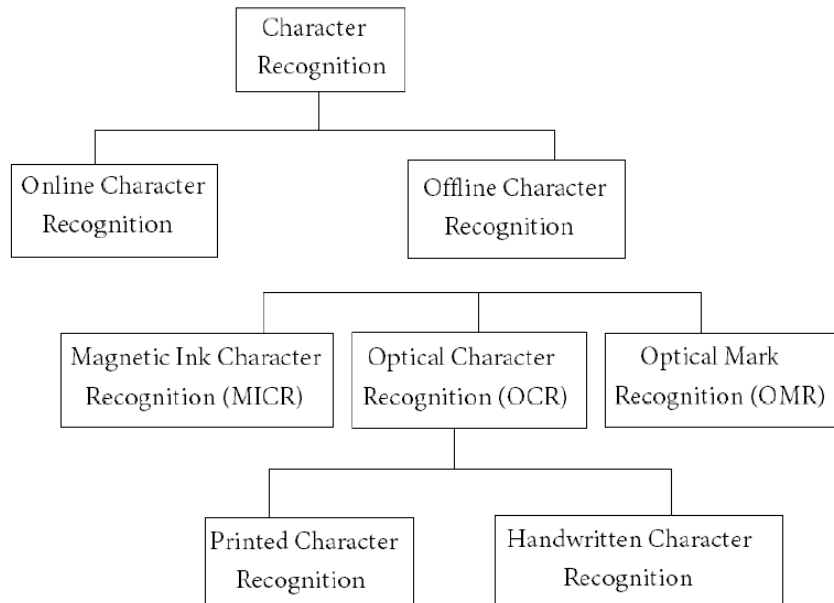


Figure 1.3: visual representation of the different types of OCR systems

storing data. However, because of variables like font styles and image quality, which might result in inaccurate character identification by the computer, it can be difficult to extract and search the contents of these documents line-by-line and word-by-word for information reuse. to make Document Image Analysis (DIA) easier. When including transforming paper documents into electronic/digital representations, character recognition algorithms are essential. With the ultimate goal of attaining accurate text recognition from images, this study suggests many methods for extracting text from photographs utilizing a certain order of processing modules [18].

### 1.8.1 Image recognition

Image recognition, commonly referred to as computer vision, is the complex process of carefully examining and decoding visual data taken from images or videos using specialized algorithms and approaches. Several impressive applications, such as item identification, facial recognition, and image-based search, are made possible by leveraging the power of image recognition systems. The amazing ability of these cutting-edge technologies to identify different entities, including objects, landscapes, patterns, and even specific features inside a given image, is on evident.[11].

### 1.8.2 Image pre-processing

There is a variety of factors that influence on the accuracy of a text detected by OCR. These elements include scanner quality, scan resolution, and the type of the document, and here comes the importance of the pre-processing stage which is a crucial step in text recognition, where the input image undergoes various transformations and enhancements to improve the accuracy and reliability of the subsequent text extraction process. It is

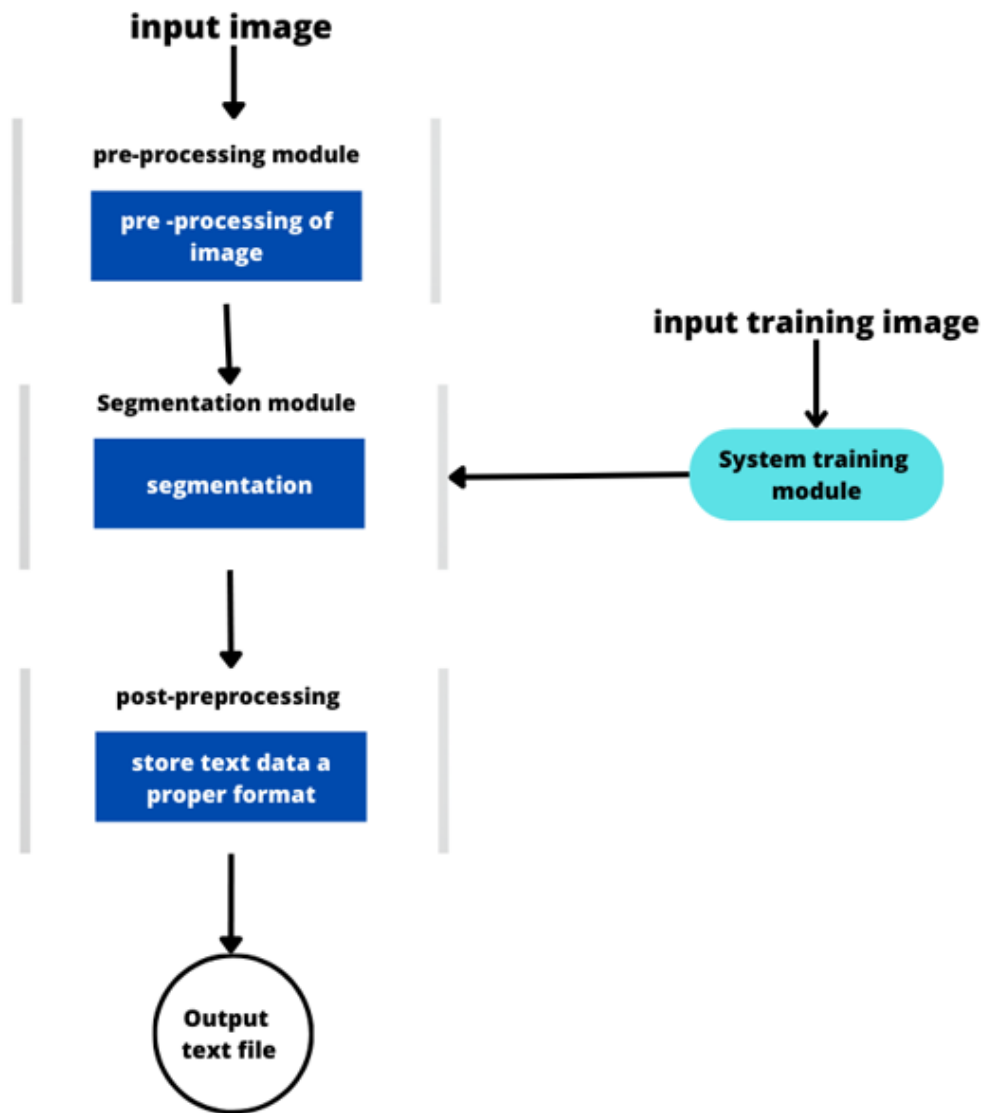


Figure 1.4: Architecture for text recognition steps

important to have an effective pre-processing stage; thus, utilizing effective pre-processing techniques makes the OCR system more resilient, mostly through image normalization, image normalization, noise removal, image enhancement, histogram equalization.[2]

### 1. Image normalization

Image normalization is a commonly employed technique in image pre-processing, aimed at standardizing the inherent characteristics of an image. Its primary objective is to make adjustments to the image, such as resizing it to a specific resolution, converting it to grayscale or binary format, or applying geometric transformations to correct any skew or distortion. Converting images to grayscale or binary format simplifies the subsequent processing tasks, as it reduces the complexity of the data representation and eliminates color-dependent variations. Geometric transformations, on the other hand, rectify any distortions or misalignments in the image, ensuring that subsequent algorithms operate on accurately aligned data. The pro-



cess of normalization ensures that all input images are presented in a consistent and uniform manner, consequently enhancing the efficiency and reliability of subsequent processing stages [2].

### 2. Noise removal

In images, noise can cause undesirable artifacts and make text extraction more difficult. A variety of noise reduction techniques, including median filtering, Gaussian filtering, and adaptive filtering, can be used to reduce these difficulties. These techniques examine the neighborhood of pixels inside the image and alter only a few of them, efficiently suppressing noise while retaining the text's key characteristics [2].

### 3. Median filter

The median filter, which works as a specialized low-pass filter, is a widely used non-linear operator. The median filter works by taking a certain area of an image (for example, 3x3, 5x5, 7x7) and sorting all of the pixel values within that area. The median value from the sorted list is then used to replace the center pixel. The median filter, unlike convolution-based filters, does not use convolution processes. When there are an even number of pixels in the neighborhood, the average of the two middle pixel values is used. The computation technique for the median filter is displayed in Figure 3. This filter is good at reducing impulse noise, such as "salt and pepper noise," which appears as random black-and-white pixels occurrences.

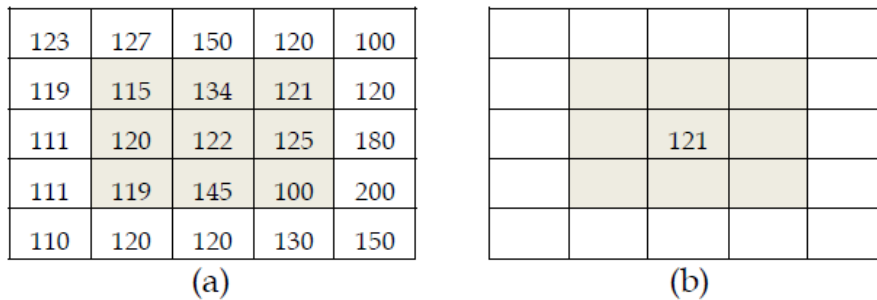


Figure 1.5: (a) Input image (b) Filtered image using median filter showing only the center pixel

The pixel values within the shaded area are sorted as follows: 100, 115, 119, 120, 121, 122, 125, 134, and 145. Consequently, the median value of 121 is obtained and applied as the output image.[2]

### 1.8.3 Enhancement techniques

Enhancement techniques in image preprocessing refer to a set of methods and algorithms aimed at improving the visual quality, clarity, and overall interpretability of images. These techniques are utilized to enhance specific image attributes, such as contrast, brightness, sharpness, and color, in order to reveal finer details and facilitate more accurate analysis or interpretation. one of the most used techniques in this step is the "histogram equalization".

Histogram equalization is a commonly used technique for enhancing image contrast on a global scale. It aims to stretch the histogram of an image across the entire range of pixel

values, typically from 0 to 255. By redistributing the pixel intensities, it increases the distinction between different brightness levels. This method is beneficial for normalizing illumination variations in image analysis tasks. This process is quite simple and for each brightness level  $j$  in the original image, the new pixel level value ( $k$ ) is calculated as given in the equation:

$$k = \sum_{i=0}^j N_i / T$$

where the sum counts the total number of pixels in the image with brightness equal to or less than  $j$  (as determined by integrating the histogram) and  $T$  is the total number of pixels [2].

**Note:** It is important to note that the order and selection of pre-processing approaches may differ based on the individual text recognition system, the nature of the input images, and the desired outcome. Numerous pre-processing algorithms and approaches have been developed by researchers and practitioners, each with its own set of benefits and drawbacks.

### 1.8.4 Segmentation

Text segmentation is a fundamental step in text recognition, dividing an image or document into individual characters, words, or lines. It sets the stage for subsequent analysis. Character recognition, a core component of text recognition, encompasses several key phases. It begins with pre-processing, where the input data undergoes various operations to enhance its quality and preparatory aspects. Then, the segmentation phase comes into play, meticulously dividing the text into its constituent elements, be it characters, words, or lines. In this section, we will present a few text segmentation methods and types.

### 1.8.5 Segmentation Methodologies

In this section, we explore different methodologies to segment a text document from an image. Numerous techniques exist for segmentation, and the following methods are frequently employed:

#### 1. Pixel Counting Approach

The Pixel Counting Approach is a segmentation methodology used to extract information from images by tallying the number of pixels that meet specific criteria, such as color, intensity, or texture properties. This methodology sets thresholds based on these criteria and calculates the number of pixels that surpass or fall within these thresholds. This method can be implemented through threshold-based counting, region-growing techniques, or texture analysis. It is known for its computational efficiency, ease of implementation, and intuitive results. However, it may struggle with complex scenes, relies on subjective thresholding, and may not provide precise shape information[10]

According to [20], the line separation technique consists of scanning the image row by row. The row on the previous line reflects the pixel row rather than the address line, indicating that the entire image is scanned from left to right and top to bottom. The intensity of the pixel is then tested for 0 or 1 (in this case, we're looking at a binarized image). A binarized image is one in which 0 represents black and 1 represents white. The algorithm would change according to the image under consideration.

### 2. Histogram Approach

The Histogram Approach is a popular segmentation method. It entails examining an image's pixel intensity distribution to identify regions or objects. This approach can be implemented through various techniques such as threshold-based segmentation, multi-threshold segmentation, Otsu's method, or adaptive segmentation. The Histogram Approach offers computational efficiency and facilitates the intuitive interpretation of segmentation outcomes. It may, however, struggle with overlapping intensities and necessitates careful threshold selection. Overall, it is a useful image segmentation technique.[10]

### 3. Histogram Projection

X histogram is used to segment words and characters and a Y histogram (Figure ) is used to segment the text lines, After completing the preprocessing steps of image binarization, noise removal, and normalization, the next step involves obtaining the Y histogram projection of the entire image. The goal is to use a simple and quick way to correctly identify potential line segments inside the text. Each text line corresponds to a peak in the histogram. The added pixels for each y value are represented by the histogram. As a result, the gaps or empty spaces between these peaks correspond to potential areas that may exist between different text lines.[10]

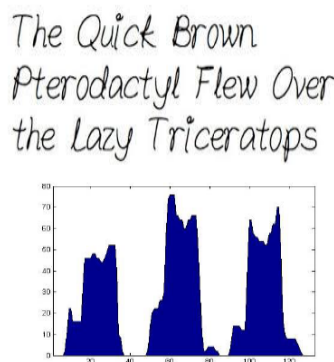


Figure 1.6: Original Text Based Image and its Histogram output

### 1.8.6 Segmentation Types

Segmentation of a text can be done in different ways, in the following we define that.

### 1. Page segmentation

This stage uses segmentation on each page where it rectangle around the page when the histogram detects each area(text .. etc), This step makes it possible to locate in each page the zones of information, By segmenting text at the page level, we can process and analyze documents page by page, enabling efficient indexing and retrieval of information.[5]

### 2. Line segmentation

Line segmentation is the process of splitting text into distinct lines. This level of segmentation is critical, By segmenting text at the line level, we can accurately recognize and interpret handwritten or printed text, and convert it into a machine-readable format, The majority of studies in the field take the method of breaking down the image into connected components. Some studies, on the other hand, rely mostly on approaches that make use of histograms of horizontal or vertical projections.[6]

### 3. Words segmentation

The following level of text segmentation is word segmentation. It entails breaking up text into component words and Word segmentation plays a vital role in various NLP (Natural language processing) tasks, One frequent way for segmenting words is to examine the histogram of vertical projections of text lines. This helps in determining the gaps between words and allows them to be separated. This method, however, may not be useful in cases where words overlap, as is frequently the case with Arabic writing. In such cases, alternate approaches for word segmentation, such as contour tracking or skeletonization, can be used.[3]

### 4. Character segmentation

Character segmentation is the process of segmenting individual characters within a line of text. Character segmentation can be complex due to variations in character shapes, sizes, and styles. Individual characters are identified and isolated using techniques such as connected component analysis, contour analysis, and machine learning algorithms.[7]

Furthermore, advances in deep learning and natural language processing (NLP) techniques have permitted the development of advanced segmentation models. These models efficiently segment text at multiple levels by utilizing various algorithms, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer models. These algorithms use contextual information from the text to segment and evaluate the data effectively.

### 1.9 Postprocessecing

Postprocessing is essential for improving the outcomes of text recognition systems. Following the initial segmentation and recognition stages, postprocessing techniques are used to improve accuracy and address faults generated during the earlier stages. The complexities of ocr should also be reduced so that it takes less time to execute the overall process.

It also includes a spell checker and a dictionary to increase the correctness of the image input.[8]

### 1.10 Conclusion

In this chapter, we have presented an introductory overview of some general concepts of Text recognition and extraction for an effective text recognition system. The description shows that text recognition can be created by using several techniques on different steps, including image preprocessing, segmentation, and postprocessing and how to prepare the image for subsequent processing. Text recognition systems continue to evolve and improve, offering promising applications in areas like artificial intelligence, and robotics. In the next chapter, we will go through a few different methods used for this process.

## Chapter 2

# Existing Text Recognition Methods

### 2.1 Introduction

Optical character recognition (OCR), also known as character and handwriting recognition, has been a prominent field of study in computer science for several decades, with its origins dating back to the late 1950s. The research of OCR writing recognition focuses on the creation of technology that can detect text from photographs or scanned documents automatically. OCR has become an important technique for digitizing printed text and handwritten documents, making it possible to extract and analyze data effectively. With improvements in computer vision, machine learning, and artificial intelligence, OCR writing recognition has developed over time, leading to increased accuracy and performance. In this chapter, we will be describing two types of methods used for text recognition, OCR-based methods and Artificial Intelligence-based methods.

### 2.2 Text recognition methods

Text recognition is a crucial field that enables the conversion of printed or handwritten text into a machine-readable format. Over the years, two prominent categories of methods have emerged for text recognition: OCR-based methods and artificial intelligence (AI) based methods. In this section, we will give describe both categories:

### 2.3 OCR-Based Methods

OCR-based methods are traditional approaches that have gained widespread usage in the field of text recognition, These methods encompass a series of sequential steps involved in image processing and the extraction of textual information.[25]

#### 2.3.1 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) involves the conversion of scanned or printed text images, as well as handwritten text, into editable and machine-readable text for subsequent processing. This technology enables machines to automatically recognize and interpret text, akin to the combined functionality of the human eye and brain. While the eye views the text within images, it is the brain that processes and comprehends the extracted text that the eye perceives. it enables the digitalization and processing of analog text, making it editable, searchable, and usable by computers.[21]

#### 2.3.2 types of OCRs

There are different types of OCR:

##### 1. Conventional OCR:

This type of OCR is based on traditional techniques that involve preprocessing, segmentation, feature extraction, and classification. It works well for standard fonts and clear

text.[21]

An example of conventional OCR is an open source optical character recognition engine is called Tesseract. It was created at HP between 1984 and 1994, and in 1995 it underwent modifications and improvements for higher accuracy. Google now develops and maintains it and It supports a number of languages. [21]

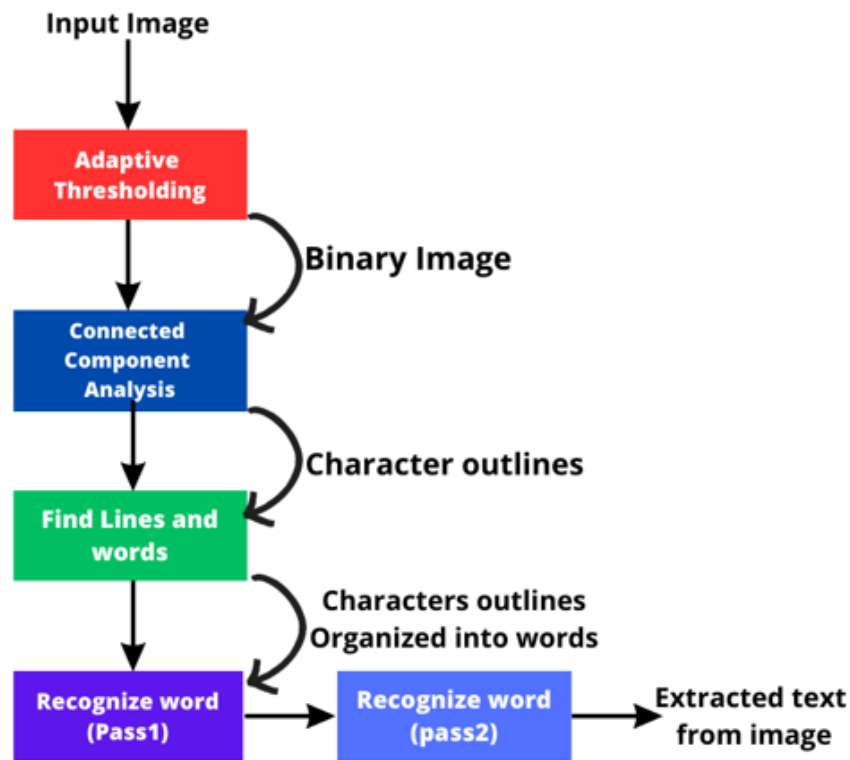


Figure 2.1: Architecture of Tesseract OCR [21]

### 2. Handwritten Text Recognition (HTR):

This type of OCR is specifically designed to recognize handwritten text. It employs techniques like image preprocessing, character segmentation, and sequence recognition. Hidden Markov Models (HMMs) and N-grams are just a couple of the principles and techniques that current HTR technology borrows. A mix of methods, including text line extraction, pre-processing activities, lexical and language modeling, etc., are required to obtain good HTR accuracy. HTR technologies that are currently on the market do not provide error-free solutions. However, the currently acquired results can be helpful for interactive transcription, search, and indexing.[23]

An example of HTR is the IAM Handwriting Database and the associated HTR system.

### 3. Scene Text Recognition:

This OCR type is focused on extracting and recognizing text from natural scene images, which often contain complex backgrounds, various fonts, and uneven lighting conditions. An example is a popular OCR system used for scene text recognition is ABBYY FineReader.



It utilizes advanced OCR techniques to extract and recognize text from complex scene images, including text with varying fonts, sizes, and orientations.[28]

### **4.Document Layout Analysis:**

Document image processing includes layout analysis algorithms, which have a variety of uses in "OCR," information retrieval, machine translation, and the development of automated document "understanding." The purpose of document layout analysis, which is the most crucial stage in the processes of document analysis and recognition, is the extraction of isolated text blocks (layout objects), such as titles, paragraphs, headers, and footers.[13]

as an example To produce layout information from a document, a bottom-up approach has been primarily used. Ishitani generates the layout based on the emergent computation model. A linked list is used by Mitchell P.E. et al. to keep all the patterns that were extracted from an image. Patterns are defined by merging adjacent rectangles, which are thought of as a rectangular region of loosely connected black pixels of a character [13].

## **2.4 AI-based methods**

With the development of deep learning and machine learning, AI-based solutions for text recognition have become more popular.

### **2.4.1 Machine Learning (ML) Methods**

Here below we describe two machine learning methods used in text recognition.

#### **1. Template Matching**

Template Matching is a machine learning method used for finding small template patterns within larger images. In text recognition, Template Matching can be applied to detect specific characters or words in an image.[22]

#### **For example:**

let's say you have a template image of the letter "A". To recognize instances of "A" in a larger image, you can use Template Matching. The algorithm compares the template image with different regions of the larger image by sliding the template over the image and calculating the similarity score between the template and each region. The region with the highest similarity score is identified as a match for the template. In text recognition, this process can be repeated for different characters or words to identify and locate them within an image.[22]

#### **2. Support Vector Machine (SVM)**

This method uses a type of machine learning algorithm that is designed for classification tasks. SVM can be used for text recognition by training the algorithm on a large dataset of images and text, and using it to recognize text in new images.[24]

### **example:**

Suppose you have a dataset consisting of images of handwritten or printed characters labeled with their corresponding classes (e.g., letters A-Z). You can extract features from these images, such as pixel intensities or shape descriptors. Using these features as input, you can train an SVM model to learn the patterns and characteristics of each class. Once trained, the SVM model can classify new unseen images of characters based on the learned patterns, enabling text recognition.[24]

### **2.4.2 Deep Learning (DL) Methods**

In the Deep Learning methods, we define the following methods:

#### **1. Convolutional Neural Network (CNN)**

This method uses a deep neural network to recognize text from an image. The network is trained on a large dataset of images and text, and it can learn to identify different font styles and sizes, as well as deal with variations in text orientation and lighting conditions [14].

for example by training the CNN on a large dataset of images, The CNN learns to automatically extract relevant features, such as edges, corners, or other visual patterns, from the input images. These learned features help the CNN recognize and classify characters or words [15].

#### **2. Recurrent Neural Network (RNN)**

The Recurrent Neural Network (RNN) is a specialized type of neural network designed to handle sequential data, such as sequences of characters in a word or sentence. Its purpose is to predict the label or output at the current time step by considering the contextual information from past time steps. RNNs are known for their effectiveness as classification models; however, they are not extensively utilized in the literature. One significant reason for this limited usage is the requirement of a lengthy training process, as the error path integral tends to decay exponentially along the sequence [26].

as an example given a sequence of input images representing characters or words, an RNN can be trained to process the sequence step by step, updating its internal memory state based on the previous inputs. This allows the RNN to capture context and dependencies between characters, aiding in accurate recognition [26].

#### **3. Long Short-Term Memory (LSTM)**

This method uses a specific type of RNN that is designed to handle long-term dependencies in data. LSTM networks are commonly used for text recognition because they are able to effectively handle the variable length of text sequences [26].

as an example To address the aforementioned issue of RNNs, the Long Short-Term Memory (LSTM) model was introduced. LSTM incorporates an internal memory structure to replace the nodes found in traditional RNNs. The output activation of the LSTM network at a given time step ( $t$ ) is determined by both the input data at time  $t$  and the internal memory stored in the network at the time ( $t-1$ ). As a result, the learning process in LSTM becomes localized and consistent. Additionally, LSTM employs a forget gate, which determines whether to reset the stored memory. This approach enables the RNN to effectively retain contextual information and mitigate errors during the learning process [26].

#### 4. Convolutional Recurrent Neural Networks (CRNN)

CRNN models combine the strengths of CNNs and RNNs. CNNs are used to extract visual features from images, and RNNs, such as LSTMs, are employed to process sequential information and generate the final text recognition output, to enhance the accuracy of scene text recognition using CRNN, various deeper CNN architectures are explored to obtain more effective feature descriptors. Specifically, VGG and ResNet architectures are introduced to train these deep models and capture informative image encodings for improved text recognition outcomes.[17]

**as an example:** The OpenOCR library is one well-known piece of software tool that uses the Convolutional Recurrent Neural Network (CRNN) architecture for text recognition, OpenOCR is an open-source OCR framework that combines the power of deep learning models, including CRNN, for text recognition tasks and it's highly customizable and can be adapted to different use cases and requirements. It supports various programming languages [16].

## 2.5 Related work

Recently, some of the best approaches have been developed to recognize entire images instead of just characters like many traditional bottom-up approaches[19][29]

These approaches can be divided into 2 methods:

### 2.5.1 Language-free methods

These approaches consider STR as a task of character-level classification, focusing primarily on leveraging visual cues to identify text. Implicit attention methods are designed for STR that is supervised by sequence-level text annotations, whereas supervised attention methods necessitate additional annotations of character-level bounding boxes.

### 2.5.2 Language-aware methods

Taking inspiration from natural language processing techniques, STR methods incorporate a language model to enhance text recognition by leveraging visual outputs. Some

approaches utilize self-attention structures for tasks requiring semantic reasoning. Additionally, certain methods employ a masked language model similar to BERT, where a language model is pre-trained to predict masked characters based on linguistic context and combine them with visual outputs to improve performance. Although these language-aware methods optimize joint character prediction probability with visual models, reducing prediction errors in the presence of linguistic context, they struggle to generalize effectively to various types of texts, particularly contextless texts with specific coding schemes. Therefore, the extraction of distinct visual features from characters remains a critical aspect of text recognition.

### 2.5.3 Self-supervised Implicit Glyph Attention

In this study, the implicit attention method was followed as the baseline structure. The glyph structures of text images were delineated as the supervision of the attention network by proposing a novel online glyph pseudo-label construction module. The learned glyph attention encouraged the text recognition network to focus on the structural regions of glyphs, resulting in improved attention correctness [12].

## 2.6 Existing applications of Medical labels in Algeria

Currently, in Algeria, there is no approved application available to handle Medical Labels. However, there is a system called SLVAP, created by two Master's students majoring in AI from the University of Mohammed Seddik Benyahia - Jijel, which is capable of detecting text within medical labels. The system they developed demonstrates an accuracy rate of 88% with a manual detection of the medical label.

## 2.7 Contribution

Our main contribution was the creation of a detection and recognition system specifically designed for medication labels. We introduced a new method for detecting medication labels within an image as distinct objects, utilizing the best available technology in the market, which is YOLOv8. Additionally, we developed a user interface that allows direct interaction with the system. Our primary focus is to guarantee the system's optimal functionality, and we are committed to delivering a reliable and robust solution that effectively addresses the challenges associated with reading medication labels.

## 2.8 Conclusion

In this chapter, we extensively explored various types of text recognition methods available today, including AI-based approaches and OCR-based approaches. We delved into the distinctions between these two methods and highlighted recent research and applications that leverage the latest advancements in text recognition techniques.

# Chapter 3

## Conception and Implementation

### 3.1 Introduction

In this chapter, we will present our contribution which consists of two proposed approaches using artificial intelligence methods. The first approach is an OCR-based approach and the second one is a VisionAI- based approach. In the following, we will describe the implementation and the experiment results of the two proposed approaches, then finish by a comparison study between the two of them.

### 3.2 Environment Setup

In this part, we will show all the environments we worked on in this project, such as the programming language and the code editors, and more.

#### 3.2.1 Python

Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. It's known for its clean and easy-to-understand structure which makes it one of the most popular languages for beginners as well as experienced programmers [1].

#### 3.2.2 Visual Studio Code

Visual Studio Code (Also referred to as VS code) is a free open-source source code that was developed by Microsoft. It is widely used and highly regarded by developers for its extensive features, flexibility, and ease of use.

#### 3.2.3 Python IDLE

Python IDLE is the default code editor that comes with installing Python, it is fairly simple and is considered to be old-fashioned compared to other modern code editors such as VS Code or Sublime Text but its simplicity and direct approach helps beginners to quickly pick up and learn the language.

#### 3.2.4 Google Colab

Google Colab, short for Collaboratory is an online cloud-based platform that provides a Jupyter notebook environment for writing and running Python code. It is a popular tool among researchers, data scientists, and developers for its convenience and collaborative features.

### 3.3 Tools Used

In this section, we will be showing all the tools and software we used in order to make our project work.

#### 3.3.1 OpenCV

OpenCV, short for Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. It provides a comprehensive set of functions and algorithms for image and video processing, object detection and tracking, facial recognition, and more.

#### 3.3.2 PyTesseract

Pytesseract is a Python wrapper for Tesseract, an open-source Optical Character Recognition (OCR) engine. OCR technology allows computers to recognize and extract text from images or scanned documents. Pytesseract makes it easy to use Tesseract's OCR capabilities in Python applications including document processing, text extraction from images, automated data entry, and more.

#### 3.3.3 Ultralytics

Ultralytics is a computer vision and deep learning research company that focuses on developing state-of-the-art models, tools, and software for object detection, tracking, and segmentation tasks. One of their biggest and most notable contributions is a software library called "YOLOv8".

#### 3.3.4 YOLOv8

YOLOv8, which stands for "You Only Look Once v8," is an advanced algorithm for object detection and tracking. It is designed to deliver real-time performance while maintaining high accuracy. Released on January 10th, 2023, this version of the YOLO framework continues to undergo regular updates and enhancements. It represents the most recent and cutting-edge advancement in object detection and machine learning models. Remarkably, YOLOv8 has the capability to detect a wide range of 80 different objects without the need for any pre-training or additional improvements.

#### 3.3.5 RoboFlow

Roboflow is a platform that offers a range of tools and services for creating custom object detection models. With its user-friendly interface, Roboflow simplifies the process of creating a custom trained YOLO model by providing step-by-step instructions. Additionally, the platform includes tools that assist in creating and organizing datasets necessary for training the model.

### 3.3.6 Google Cloud Vision AI

Google Cloud Vision AI is a suite of pre-trained machine learning models and APIs (Application Programming Interface) developed by Google. It offers a range of advanced computer vision capabilities that allow developers to integrate powerful image and video analysis features into their applications.

## 3.4 Implementation

After seeing the lack of intelligent recognition of medical labels and the text within them, we have decided to use 2 methods each with their own different approaches to explore the different OCR engines and to compare the results of each method and determine which method is more efficient than the other and to understand why that is the case.

### 3.4.1 Dataset Collection

Before we began our implementation, a dataset that contains medical labels and their content was required for doing a comparison study and for creating a medical label detection model. But after a continuous search, we have not found any existing database which lead us to creating one of our own.

We first requested permission from a local pharmacy named "Dis Mebarek" in (khemis miliana-Ain Defla) to take pictures of approximately 300 medications (About 1020 pictures) then we began the process of collecting the pictures. Each member of the group took about 300 pictures using the following phone specs:

- **Redmi Note 10:**
  - Dimensions: 160.5 x 74.5 x 8.3 mm
  - Resolution: 1080 x 2400 pixels
  - CPU: Octa-core (2x2.2 GHz Kryo 460 Gold & 6x1.7 GHz Kryo 460 Silver)
  - Internal Memory: 64GB 6GB RAM
  - Quad Camera: 48 MP, f/1.8, 26mm (wide), 1/2.0", 0.8µm, PDAF  
8 MP, f/2.2, 118 ° (ultrawide), 1/4.0", 1.12µm  
2 MP, f/2.4, (macro)  
2 MP, f/2.4, (depth)
- **Xiaomi Poco X3 NFC:**
  - Dimensions: 165.3 x 76.8 x 9.4 mm
  - Resolution: 1080 x 2400 pixels
  - CPU: Octa-core (2x2.3 GHz Kryo 470 Gold & 6x1.8 GHz Kryo 470 Silver)
  - Internal Memory: 64GB 6GB RAM



- Quad Camera: 64 MP, f/1.9, (wide), 1/1.73", 0.8µm, PDAF  
13 MP, f/2.2, 119 ° (ultrawide), 1.0µm  
2 MP, f/2.4, (macro)  
2 MP, f/2.4, (depth)
- **Honor 8X:**
  - Dimensions: 160.4 x 76.6 x 7.8 mm
  - Resolution: 1080 x 2340 pixels
  - CPU: Octa-core (4x2.2 GHz Cortex-A73 & 4x1.7 GHz Cortex-A53)
  - Internal Memory: 64GB 6GB RAM
  - Dual: 20 MP, f/1.8, 27 mm (wide), PDAF  
2 MP, (depth)

After we finished photographing the medications, our database had about 1024 pictures that we used for the upcoming training, in addition to another 73 pictures that were given to us by the Master thesis [4] members that we used for the evaluation and the comparison study.

After photographing the medical labels, we began documenting the information on each label in an Excel file to compare it with the results of both methods and calculate how accurate they are. The information on the label is documented in the following format:

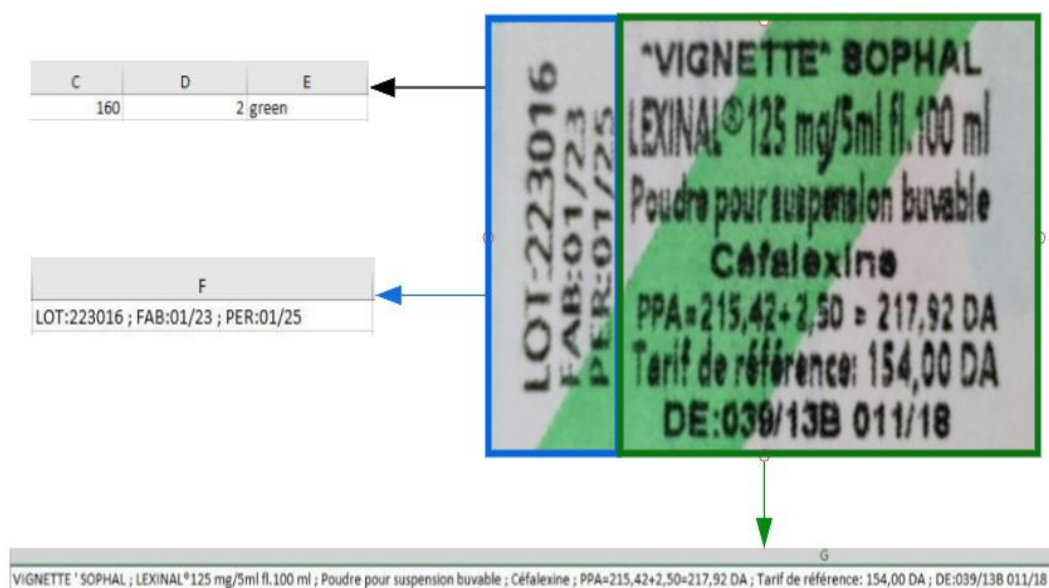


Figure 3.1: Content of a medical label with 2 parts

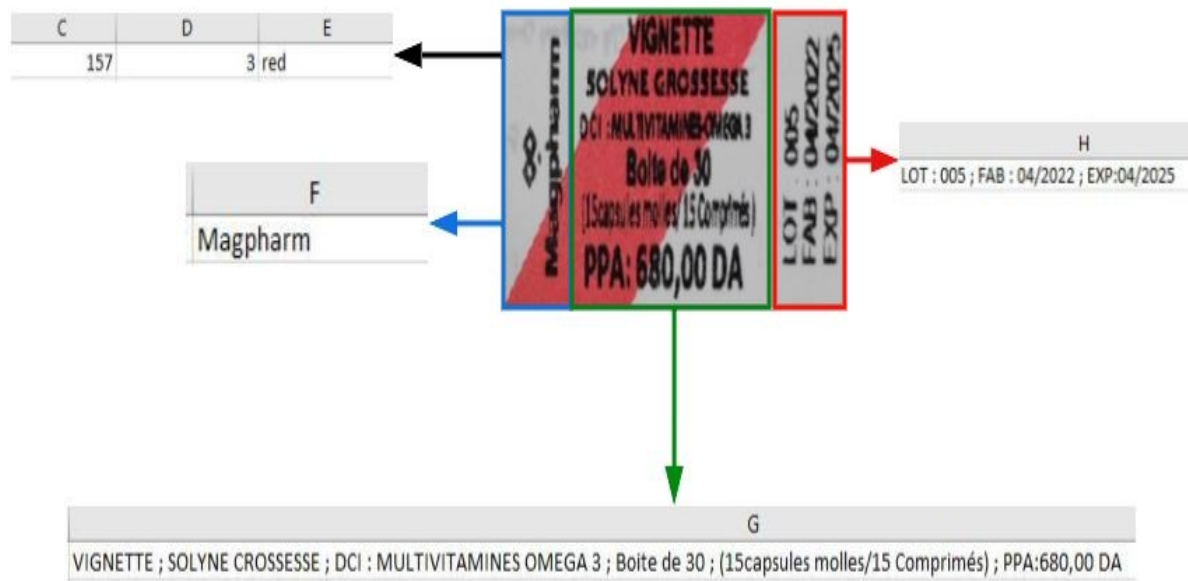


Figure 3.2: Content of a medical label with 3 parts

Here's what each Cell in the Excel file represents:

- Cell C: Number of the medical label in the Excel file.
- Cell D: Number of the parts (The vertical and horizontal parts) in the medical label
- Cell E: The color of the strap (Green for Refundable and Red for Nonrefundable)
- Cell F: Content of the Left part of the medical label
- Cell G: Content of the Middle part of the medical label
- Cell H: Content of the Right part of the medical label

### 3.4.2 Tesseract-based Approach

The Tesseract-based approach uses a classic semi-intelligent approach, from a manual selection of the 4 corners of the medical label, applying a histogram to detect each vertical and horizontal written part and the lines of texts within these parts, all the way to character recognition. This method uses traditional detection and segmentation methods such as image segmentation using histogram thresholding [27].

### 3.4.2.1 FlowChart of the Tesseract-based approach

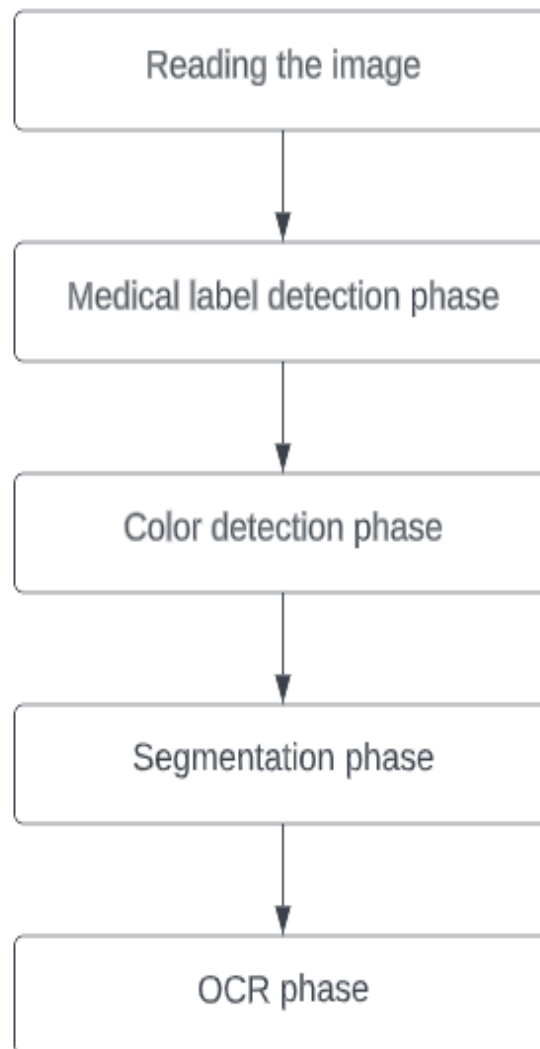


Figure 3.3: Tesseract-based approach FlowChart

### 3.4.2.2 Medical Label Detection phase

In the initial stage of the Tesseract-based approach, the medical label detection process involves the manual selection of the four corners of the label by clicking with a mouse. Once the selection is completed, the system performs a corrective rotation to align the label in the correct position. Finally, the result is presented and displayed.



Figure 3.4: Medical label detection process

### 3.4.2.3 Color Detection phase

In the second phase of the Tesseract-based approach, the color detection of the medical label is performed to determine if the medicine is refundable or not. The green color represents refundable medicines, while the red color indicates non-refundable medicines.

To accomplish this, a function is defined that establishes specific ranges for what is considered as red and green pixels. The function then iterates through each pixel of the image, comparing its RGB value with the defined red and green ranges. It keeps track of the count of red and green pixels encountered. Once all pixels have been examined, the function compares the count of red and green pixels and prints "Remboursable" if the count of green pixels exceeds the count of red pixels, or "Irremboursable" if the count of red pixels surpasses the count of green pixels, and if neither surpass the other meaning that there was no count for red or green, then the medical label is a white label and it will print "Irremboursable".

### 3.4.2.4 Segmentation phase:

The first step of text segmentation is detecting the vertical and horizontal lines within the medical label. In general, medical labels consist of a main part in the middle written horizontally that usually contains the company name, the drug's name, the dosage and the price. The second part is the fabrication, expiration date, the lot number, which is usually written vertically and it can be found on either sides of the medical label.

In order to detect each vertical and horizontal part of the medical label, A gray-scaling is required for the resulting image from the Medical Label Detection phase. Gray-scaling is the process of converting a colored image into a black-and-white image by combining the RGB values of each pixel that represents its brightness. This step is necessary for image processing since it simplifies the image and makes it easier to analyze and process.

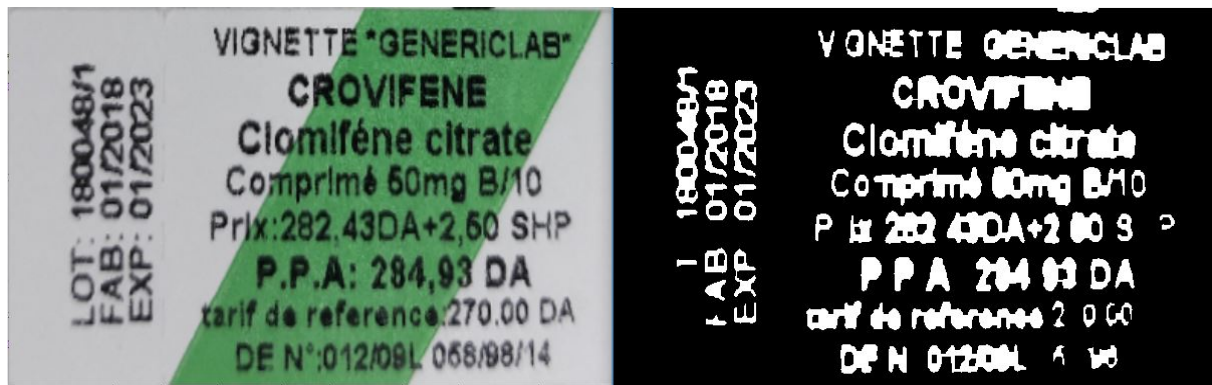


Figure 3.5: Gray-scaled Medical label

The next step is separating the vertical and horizontal written parts of the medical label. This can be achieved by using the image's Histogram.

An image histogram is a graphical representation that shows the distribution of pixel intensities in an image. It provides information about the frequency of occurrence of different intensity levels or colors in the image.

After the resulting image has been gray-scaled, the Histogram is applied to calculate the average intensity of pixels for each column of the image's pixels, and if there are five or more columns with average pixels of 0 it means that there is a gap between the vertical part and the horizontal part and not a gap between the letters, then the image will be cut there and the two parts will be separated. And the process is repeated throughout the entire image in case the image had three or more different parts. Then the positions of the detected parts on the gray-scaled image are saved and applied on the original image.

Upon separating the vertical and horizontal written parts, the vertical portions are identified by comparing the dimensions of the image. If the height exceeds the width, a clockwise rotation is performed to align these segments, though it is not always correct since some vertical parts are written from top to bottom so a clockwise rotation will flip the parts to the wrong direction.

Subsequently, each line of text within the identified vertical parts is segmented. This is achieved by applying a horizontal histogram to a gray-scaled version of the relevant segment. When a line of pixels exhibits an average pixel value of zero, indicating a lack of text, the corresponding line is highlighted. This process is repeated until the entire image has been processed. Once the histogram analysis is complete, the highlighted lines will be deleted and the positions of the non-highlighted segments are recorded and applied to the original colored image, which leads to the final step of segmenting and cutting the image using the positions given. This process is repeated with every detected part.

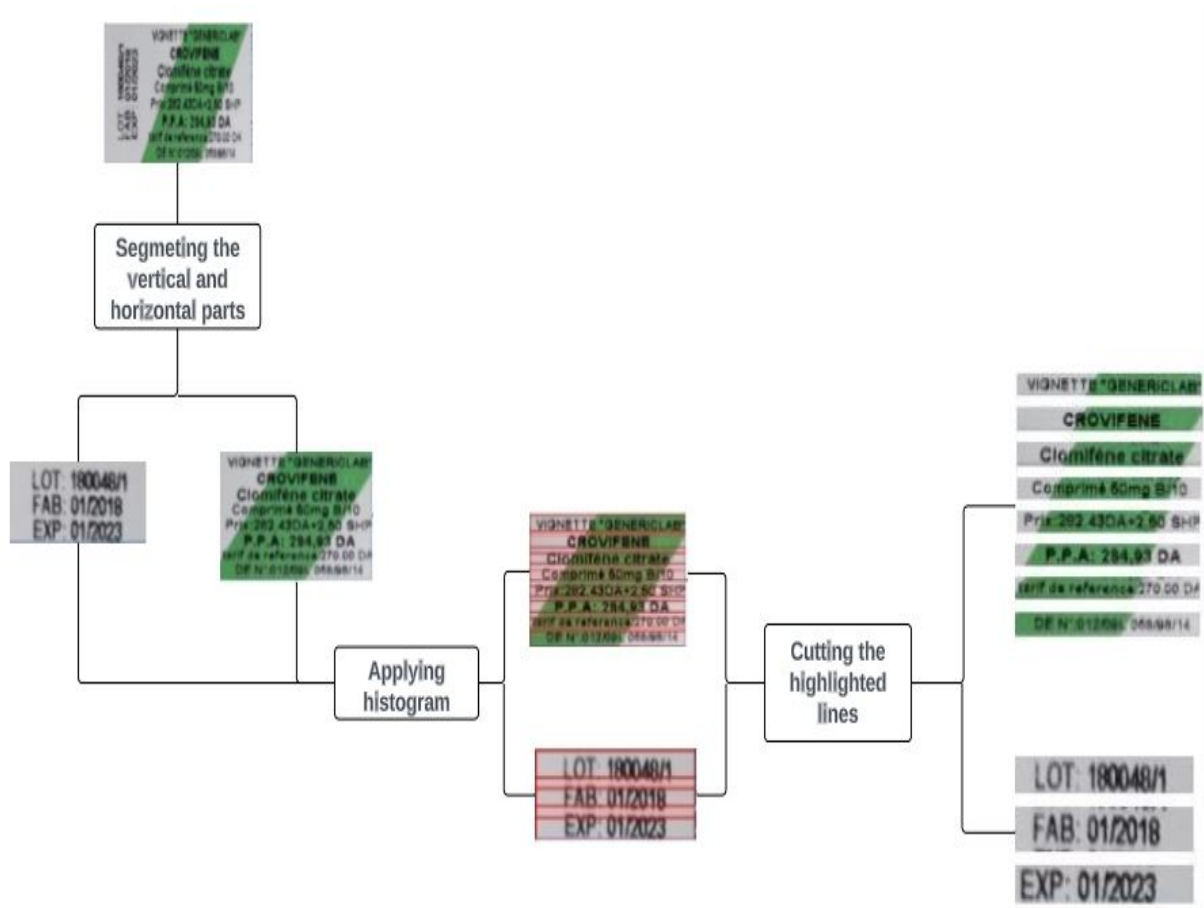


Figure 3.6: Segmentation process

### 3.4.2.5 Character Recognition phase:

Once the lines of text have been segmented, an immediate OCR is applied individually on each line of text using the PyTesseract OCR Model.

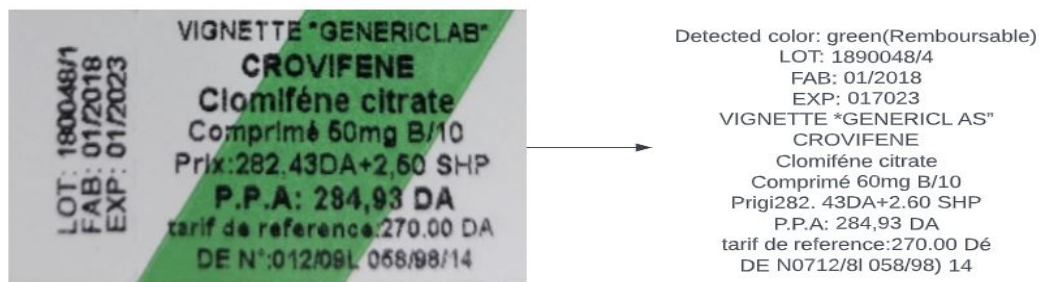


Figure 3.7: Tesseract OCR results

The results in this method are not always accurate, this is caused by the unrecognized letters in the medical labels mainly because the letters are not well printed on the label so ultimately the result will not be accurate. Another factor that is worth mentioning is the bad quality of the picture since the quality is reduced when using an image that is taken by a phone on a computer.

### 3.4.2.6 Algorithm of the Tesseract-based Approach

**BEGIN**

```
/*Beginning of Medical Label detection process*/
<Acquiring the image>;      /*Read image*/
<Medical label detection>;   /*Selection of the four corners*/
<Rotating the image to correct position>;
<Resizing the image>;
<Color Detection>;          /*Refundable or non-Refundable*/
/*End of Medical Label detection process*/
/*Beginning of Segmentation process*/
<Gray-scaling the resulted image>;
<Vertical histogram on the gray-scaled image>;
<Segmenting the detected part>;
<Applying the positions on the original image>;
<Rotation of the Vertical-written parts>;
For Part in Parts Do
    <Gray-scaling the detected part>;
    <Horizontal histogram on the gray-scaled part>;
    <highlighting the lines with no text>;
    <Deletion of the part highlighted lines>;
    <Save the position of the the non-highlighted part>;
    <Apply the positions saved to the original colored image>;
/*Beginning of OCR process*/
    For Line in Lines Do
        Text ← PyTesseract(Segmented_Line)
        Print(Text);
    EndFor
/*End of OCR process*/
EndFor
/*End of Segmentation process*/
```

**END**



### 3.4.3 VisionAI-based Approach

This second approach is based entirely on Artificial Intelligence, implemented with the latest and most up-to-date models and APIs as of today. It uses a YOLOv8 Model for detecting the medical label and Vision-AI for the OCR and Color detection.

#### 3.4.3.1 FlowChart of VisionAI based approach

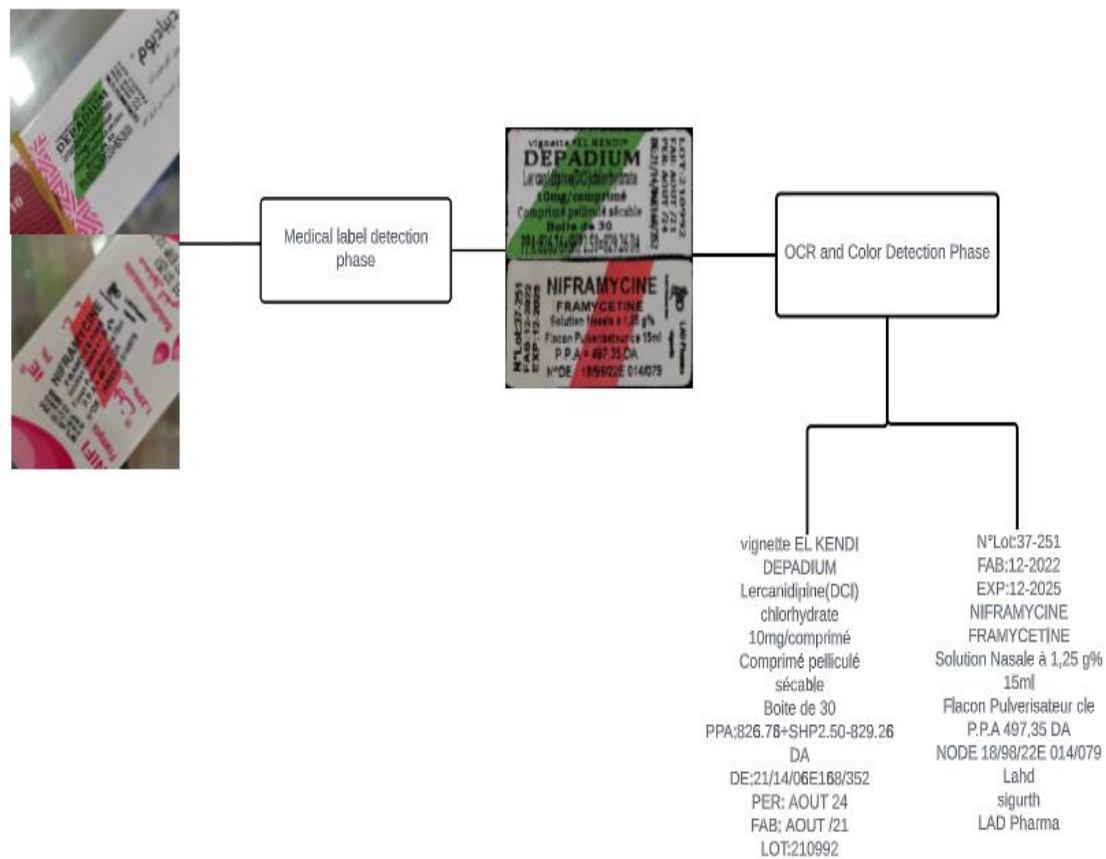


Figure 3.8: VisionAI-based approach FlowChart

#### 3.4.3.2 YOLOv8 Training phase

In order to begin training our model, a dataset is required containing files for Training, Validating, and Testing. Each file should contain the images and their labels so we used the database that we collected (1024 pictures).

Then we uploaded all the pictures into the RoboFlow platform which helps with the annotation and expanding the dataset for more accurate and precise results. And began manually annotating all the 1024 pictures.



### Yolov8 Dataset preparation and pre-processing

As described in the The YOLOv8 training phase, The model requires the dataset to be in a specific format, it requires it to be split into 3 folders Train, Valid and Test. And each folder must have 2 folders inside, an Images folder containing the images and a Labels folder that contains text files that has the coordinates of the annotations for each image in the Images folder. Meaning that each image must have a corresponding text file that has the positions of the object that is being detected.

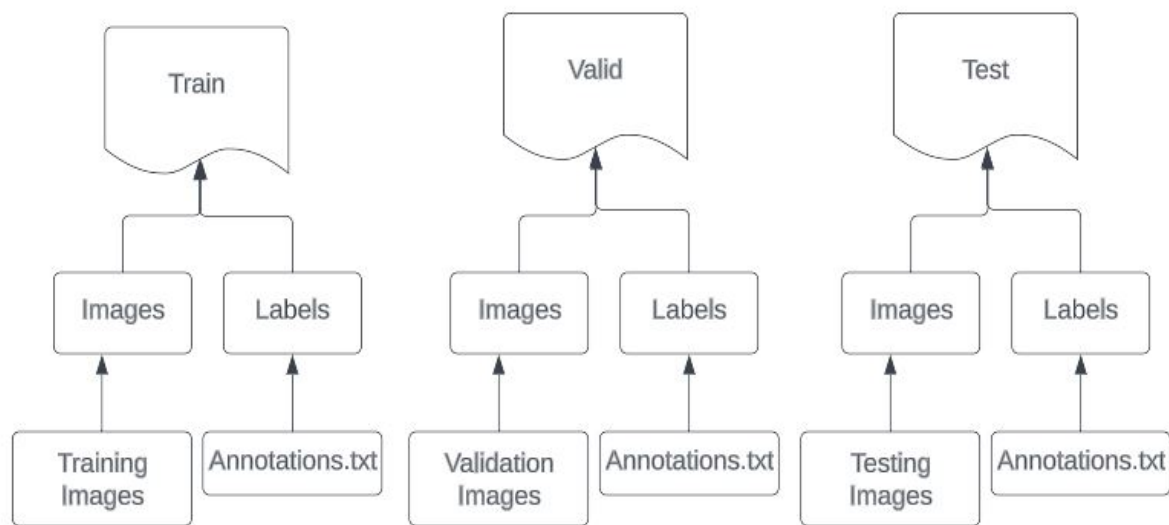


Figure 3.9: Dataset structure

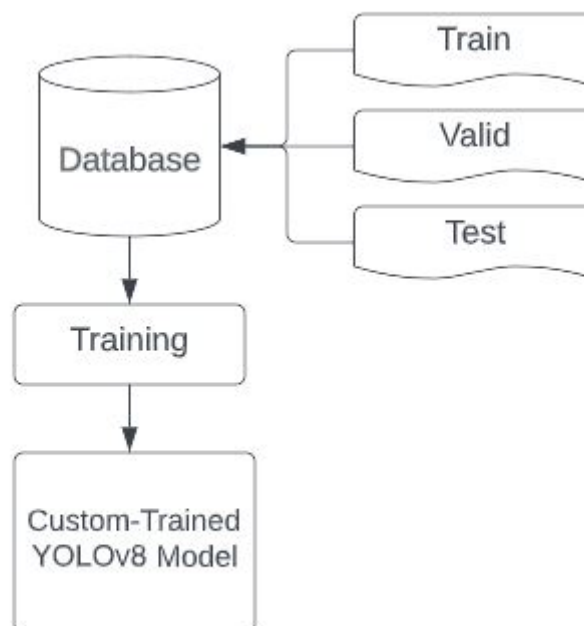


Figure 3.10: Training process

### Preparing the Dataset

The RoboFlow platform offers the tools to prepare the dataset and export it in the format that is required by the YOLOv8 Model.

We first create an account and a project then we name the object we are trying to detect.

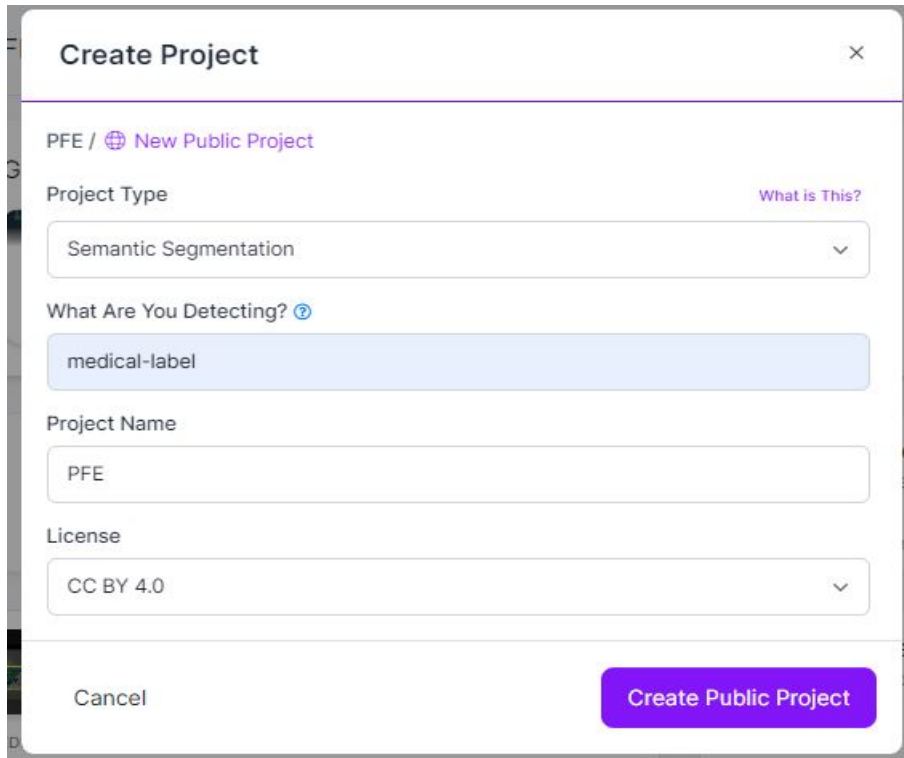
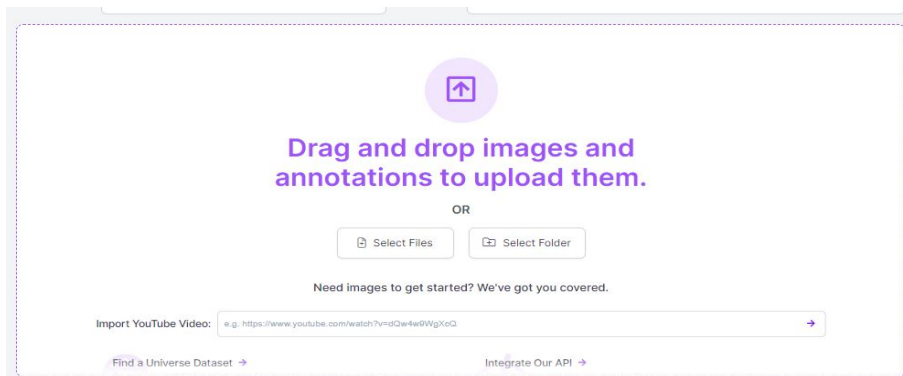
The image shows a 'Create Project' modal window from the RoboFlow platform. At the top, it says 'PFE / New Public Project'. Below this, there's a 'Project Type' dropdown menu set to 'Semantic Segmentation'. A link 'What is This?' is next to it. Then, 'What Are You Detecting?' is followed by a text input field containing 'medical-label'. Below that is the 'Project Name' field with 'PFE' entered. The 'License' dropdown is set to 'CC BY 4.0'. At the bottom, there are two buttons: 'Cancel' and 'Create Public Project'.

Figure 3.11: Creation of a project in Roboflow

Next we upload the images taken of the medicines (Over 1000 Pictures)

The image shows the main upload interface of the RoboFlow platform. It features a large dashed box with a purple circle containing an upload icon at the top center. Below the icon, the text 'Drag and drop images and annotations to upload them.' is displayed. Underneath, the word 'OR' is centered, followed by two buttons: 'Select Files' and 'Select Folder'. A message 'Need images to get started? We've got you covered.' is shown. At the bottom, there's an 'Import YouTube Video:' field with a placeholder URL and a right-pointing arrow. Below this, there are two links: 'Find a Universe Dataset' and 'Integrate Our API'.

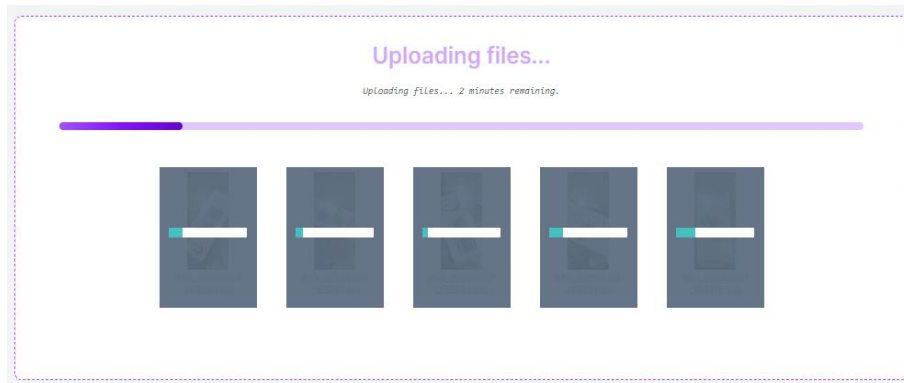


Figure 3.12: Image uploading

The next step is the most time-consuming step, which is the manual annotation of each image by selecting the 4 corners of the Medical label.

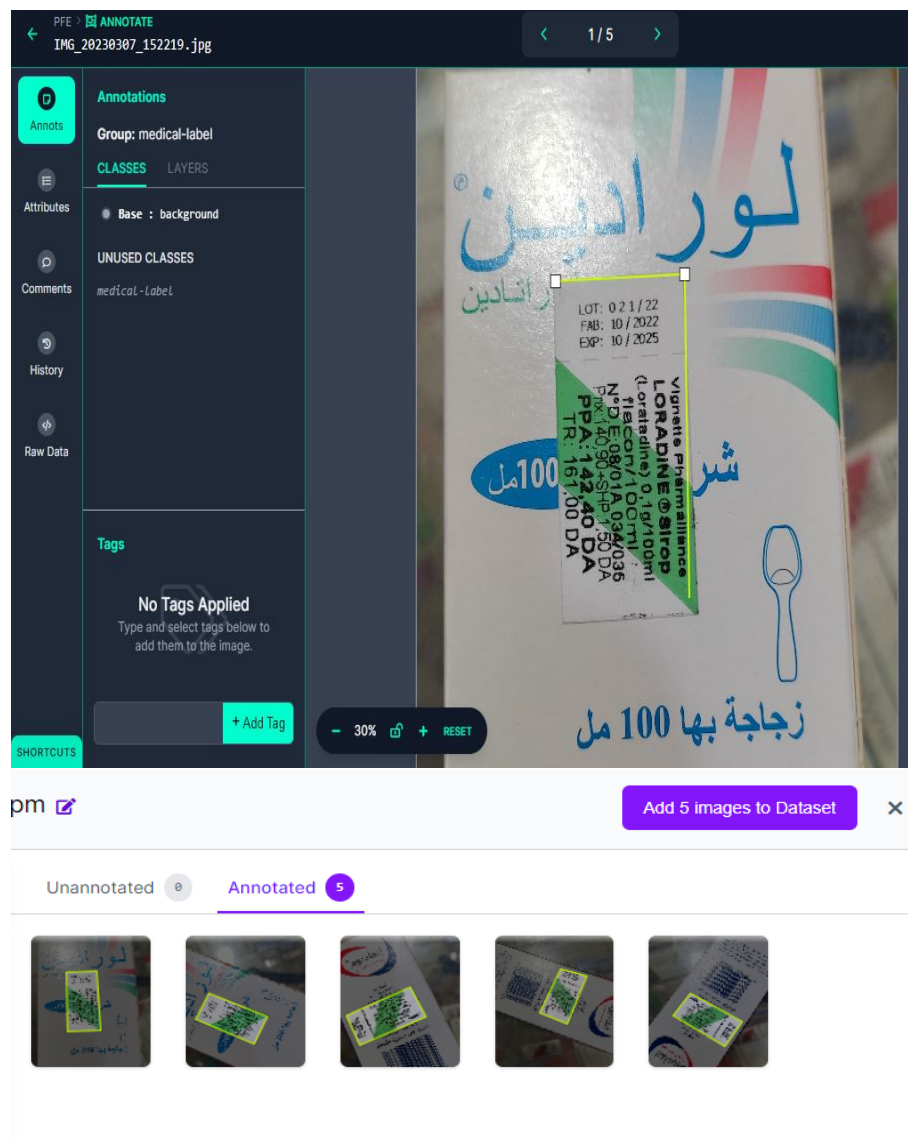


Figure 3.13: Image annotations

Once the annotations are done and all the images has been annotated, the next step is generating rotated and inverted versions of each image in our training set which ultimately helps our model be more accurate.

The screenshot shows a three-step process for dataset augmentation. Step 3, 'Preprocessing', is the active step, indicated by a purple circle with the number 3. It includes a help icon and the text 'What can preprocessing do?' followed by an explanation: 'Decrease training time and increase performance by applying image transformations to all images in this dataset.' Below this is a list of preprocessing steps: 'Auto-Orient' and 'Resize' (with the detail 'Stretch to 640x640'). Each step has an 'Edit' link and a close 'x' button. At the bottom of the list is a '+ Add Preprocessing Step' button. A green 'Continue' button is located below the list. Steps 4, 'Augmentation', and 5, 'Generate', are listed below step 3 but are not active.

Step	Step Name
3	Preprocessing
4	Augmentation
5	Generate

Figure 3.14: Augmentation of the Dataset

The final step is exporting the dataset that is now ready and in the correct format.

The screenshot shows a dialog box titled 'Export' with a close 'x' button in the top right corner. It features a 'Format' dropdown menu currently set to 'YOLOv8'. Below the dropdown is a note: 'TXT annotations and YAML config used with YOLOv8.' There are two radio buttons: 'download zip to computer' (which is selected) and 'show download code'. At the bottom of the dialog are two buttons: 'Cancel' and 'Continue'.

Figure 3.15: Dataset Exportation

### Creating the YAML file:

The YOLOv8 model also requires a YAML file which is a configuration file that has the number of classes (Objects) that are being detected, the path to the database and the path to Train, Valid and Test files. The model will use this file to know what it is training for and where to find the training set.

```
path: ../drive/MyDrive/Pfe/Pfe.v2i.yolov8
train: train/images # train images (relative to 'path') 128 images
val: valid/images # val images (relative to 'path') 128 images
test: test/images # test images (optional)

# Classes
names:
  0: 'medical label'
```

Figure 3.16: Content of the YAML file

### Training process:

Once the dataset and YAML file have been prepared, the training process can now be started.

For our training, we used the Google Colab environment for faster and more accurate results. The first step in Google Colab is importing our Drive content which contains the dataset and YAML file. Then we install the Ultralytics library using the 'pip install Ultralytics' command since the library contains the Train and Predict classes and methods needed.

The final step is starting the training using the command below:

```
[ ] !yolo task=segment mode=train model=yolov8x-seg.pt imgsz=640
    data=/content/drive/MyDrive/Pfe/vign_dataset.yaml
    epochs=15 batch=10 name=Vignette_detect
```

Figure 3.17: YOLO command

Here is a brief explanation of the arguments in the command:

- Task: Whether we want to detect, segment, or classify on the dataset of our choice.
- Mode: Mode can either be train, val, or predict. As we are running training, it should be train.
- Model: The model that we want to use. Here, we use the YOLOv8 x model since it is the best one out of 5 models.

- `imgsz`: The image size. The default resolution is 640.
- `Data`: Path to the dataset YAML file.
- `Epochs`: Number of epochs we want to train for.
- `Batch`: The batch size for data loader. You may increase or decrease it according to your GPU memory availability.
- `name`: Name of the results directory for runs/Segment.

Once we run the command, the yolov8x-seg model will be downloaded then the dataset path will be verified then the training process begins. It could take anywhere from 1 Hour to 3 Hours depending on the dataset size, number of epochs and the type of model used.

Finally, once the training is done, a folder will be created containing the results of the training process including our Custom-Trained model with the name 'best.pt'.

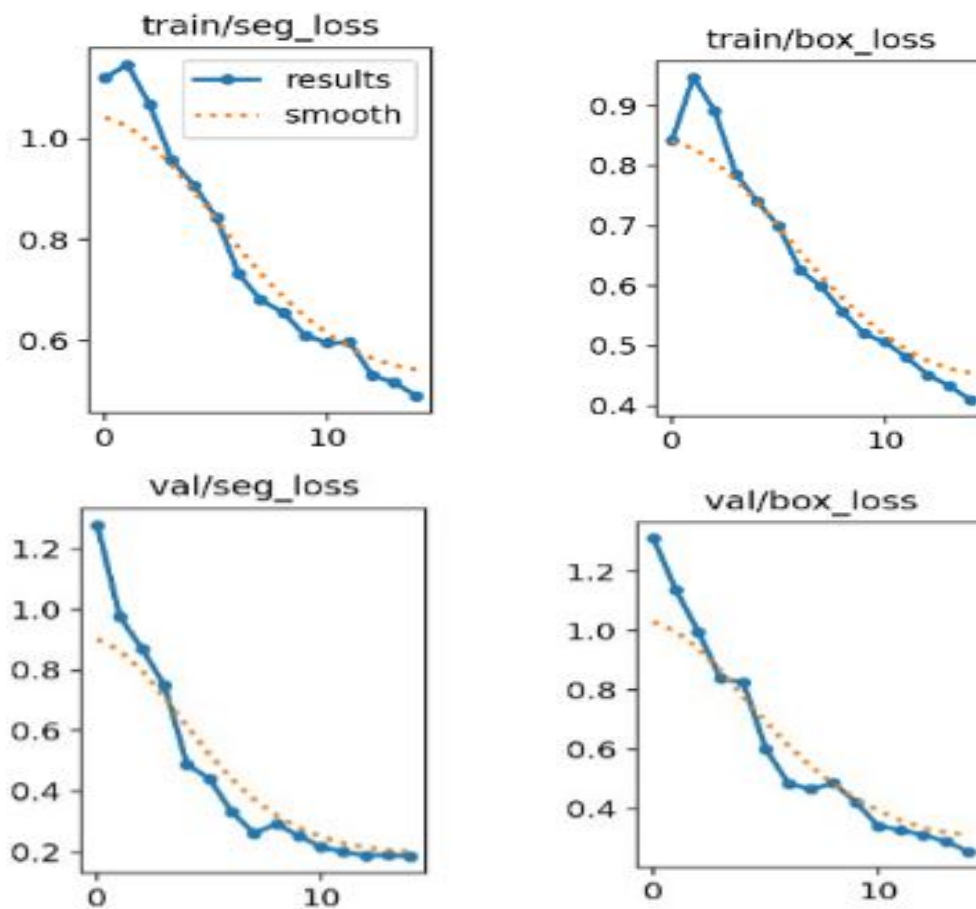


Figure 3.18: Training and Validation results

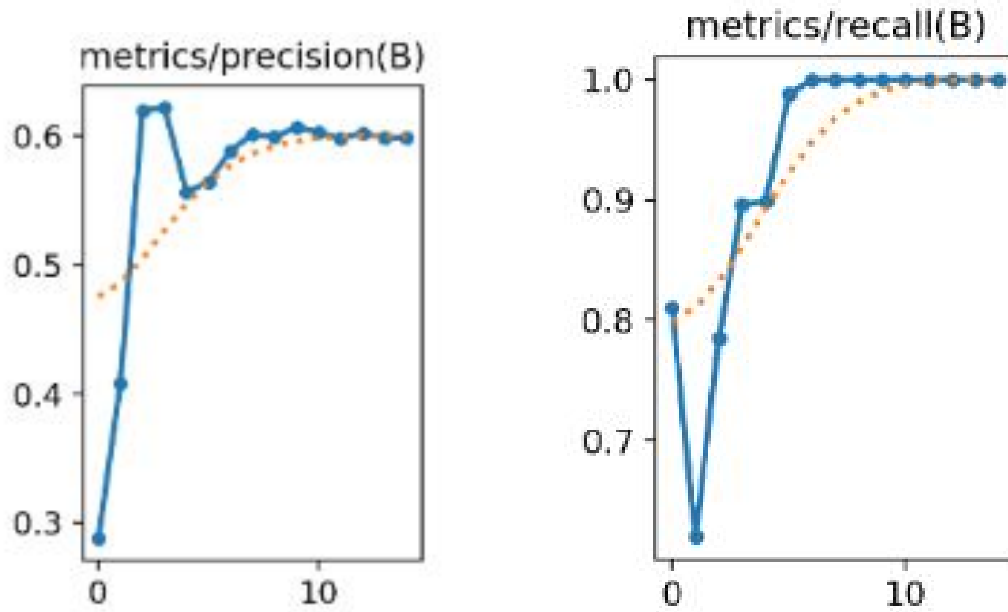
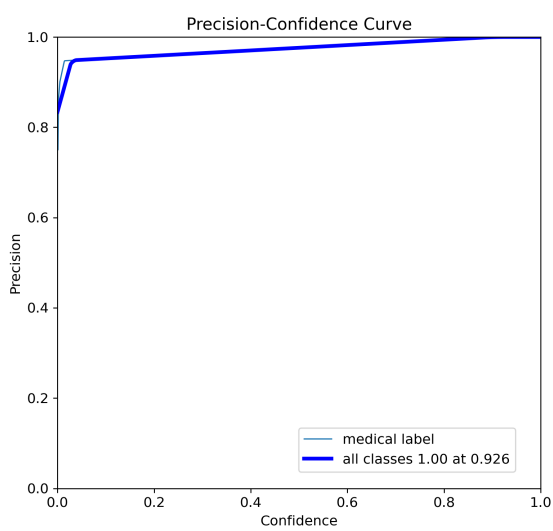
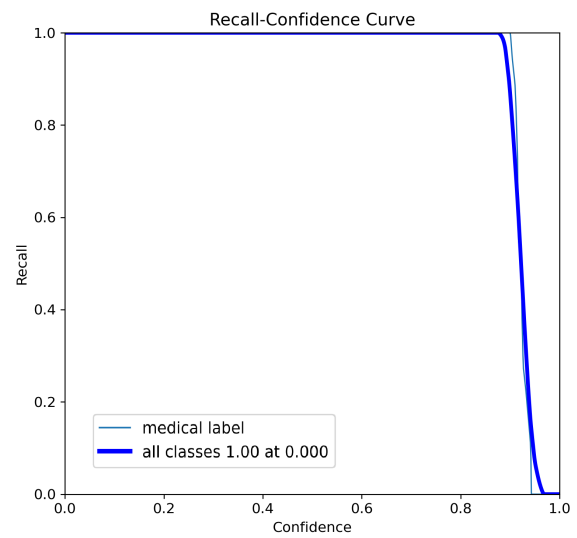


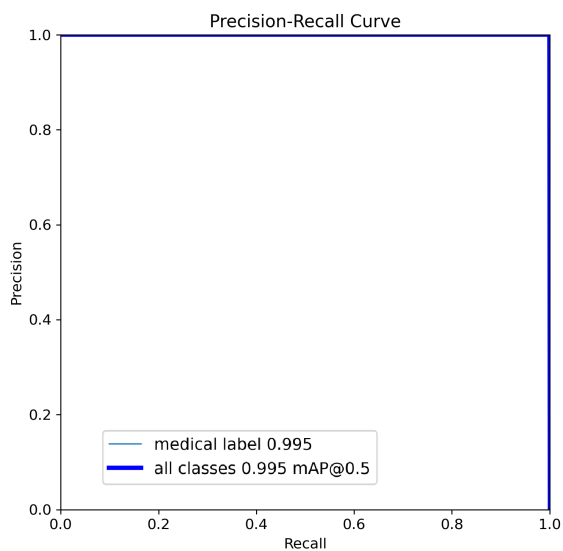
Figure 3.19: Metrics results



(a) Precision Testing results



(b) Recall Testing results



(c) Precision-Recall Testing results

Here's a brief explanation of what the plots represent:

- Train/seg\_loss: The loss that measures how precise the segmentation is on the predicted object during the training phase.
- Train/box\_loss: The loss that measures how close the bounding box is to the predicted object during the training phase.
- Metrics/precision: how many retrieved objects or items are relevant.
- Metrics/Recall: how many relevant objects or items are retrieved.
- Val/box\_loss: The loss that measures how close the bounding box is to the predicted object during the validation phase.
- Val/seg\_loss: The loss that measures how precise the segmentation is on the predicted object during the validation phase.
- Precision-Confidence Curve: a graphical representation that illustrates the relationship between Precision and Confidence thresholds.
- Recall-Confidence Curve: a graphical representation that illustrates the relationship between Recall and Confidence thresholds.
- Precision-Recall Curve: a graphical representation that illustrates the trade-off between precision and recall for different thresholds.

### 3.4.3.3 Google Cloud Vision API set-up

Google Cloud Vision API is considered to be the best detection API since it is developed by Google, one of the largest companies in the world. In addition, it allows us to integrate vision detection features within applications, including image labeling, face and landmark detection, optical character recognition (OCR), and tagging of explicit content.

In order to use the Google Cloud Vision API, we first need to create a google cloud platform account. Then we enable the API and set-up a Service Account that has the permission to use the API. Once the service account is created, we generate a Service Account Key and download it as a JSON file format.



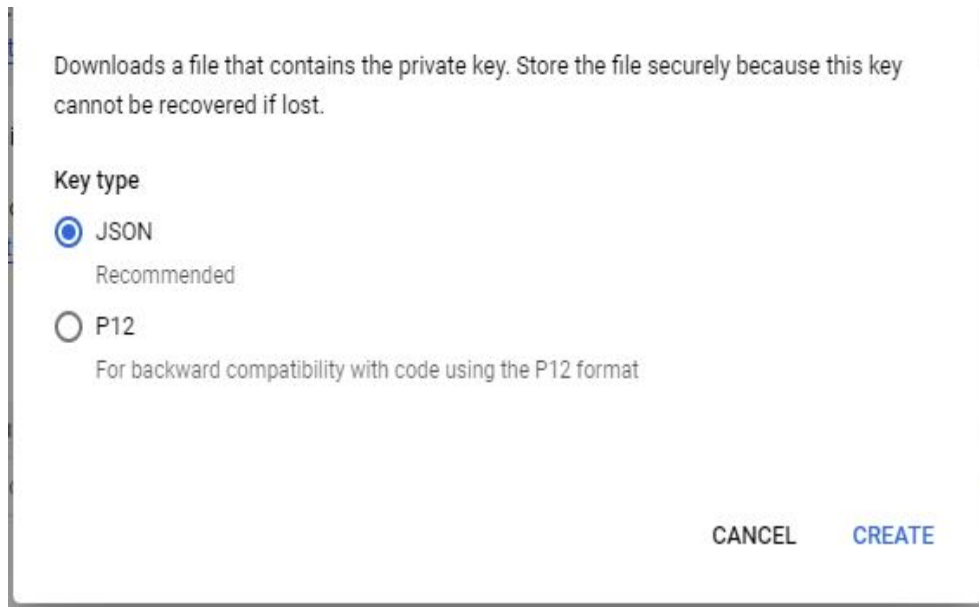


Figure 3.21: Service account key export

Once we acquire the file, an authentication is required in order to use the API, this can be done using the following code:

```
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'account_key.json'

client = vision.ImageAnnotatorClient()
```

Figure 3.22: Authentication and verification of the service account

### 3.4.3.4 Medical Label recognition phase

Once the authentication is done, we load our Custom-Trained model then we give it the image that we are trying to detect as a Source then we set Save as True to save the results in a file and resize the image to 640.

```
model = YOLO("best.pt")
im1 = cv2.imread("test_images/pic2.jpg")
results = model.predict(source=im1, imgsz=640, save=True, save_crop=True)
```

Figure 3.23: Model and Image loading

After running the code, a file will be created containing the results and cropped version of the image.

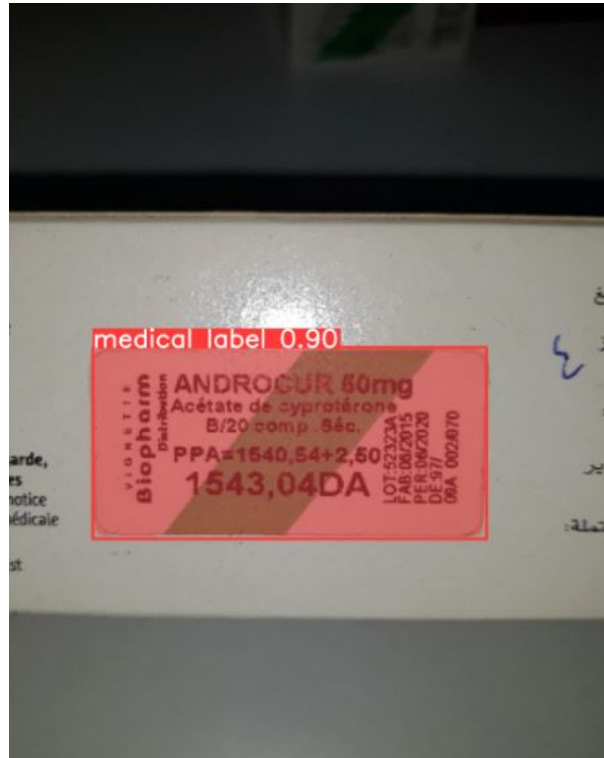


Figure 3.24: Results of the YOLO detection

The obtained results do not solely represent the medical label. To fix this, an additional processing step has been incorporated. This step involves duplicating the masks (highlighted in orange in the results image) and overlaying them onto the original image. By doing so, only the medical label is displayed.

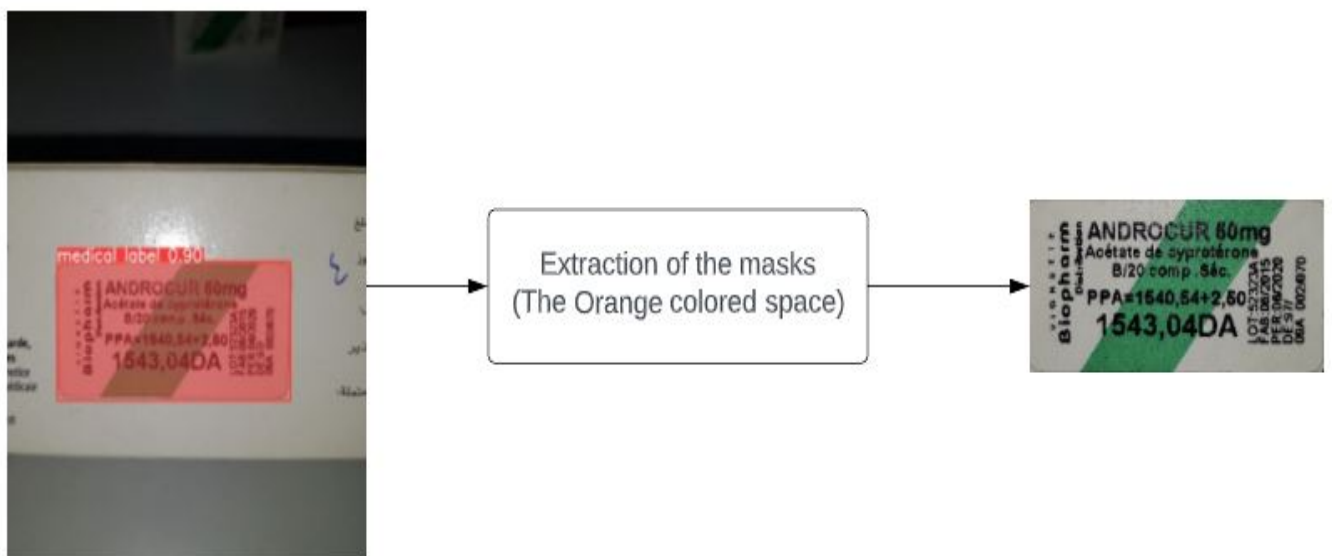


Figure 3.25: The final results of the medical label detection

### 3.4.3.5 VisionAI OCR phase

The final step in the VisionAI-based method is applying OCR on the results, this is done by using the **Detect\_Text** function that can be found in the Cloud Vision Documentation [9]. However, the function as it is in the document will display the results in an unwanted format that contains unnecessary information such as Score, Topicality. So we adjusted the function to make it display only the information we need in a simple and clean format.



Figure 3.26: VisionAI-OCR results

### 3.4.3.6 VisionAI Color detection phase

Each resulted image has its properties that are associated with it once it is detected by VisionAI, such as dominant color, number of Blocks, Pages, Lines etc. We can access these properties by using the pre-defined function **Detect\_properties** but the function will return so unwanted information so we modified it to make it determine whether the green or red colors are more dominant. If the green color is more dominant then it will print "Green(Remboursable)" and if the red color is more dominant it prints "Red(Irremboursable)". And if neither colors are more dominant than the other, it prints out "White(Irremboursable)" instead.

### 3.4.3.7 Algorithm of the VisionAI-based approach

BEGIN

```
<Google Service Account Authentication>;  
/*Beginning of Medical Label detection process*/  
<Acquiring the image>;      /*Read image*/  
Model  $\leftarrow$  Best.pt;      /*Load the Custom-Trained model*/  
Results  $\leftarrow$  Model.predict(Inserted_Image);      /*Load the image in the model*/  
<Applying the masks on the original image>;  
Print(Results);  
/*End of Medical Label detection process*/  
/*Beginning of OCR process*/  
<Acquiring the image>;      /*Reading the image of the detected medical label*/  
Text  $\leftarrow$  VisionAI(Client,Resulted_Image);      /*Load the resulted image and Client*/  
<Calling the detect_properties function>;  
Print(Text);  
Print(Color);      /*Refundable or Non-Refundable*/  
/*End of OCR process*/
```

END

### 3.4.4 Graphical User Interface

Python provides a wide range of options for creating graphical user interfaces (GUI), with several popular tools available, including:

- Tkinter: The standard GUI toolkit included with Python, providing a simple and easy-to-use interface for creating basic GUI applications.
- PyQt: A powerful and feature-rich library for creating GUI applications using the Qt framework, offering extensive capabilities and cross-platform compatibility.
- Pygame: A library for creating games and multimedia applications, offering functionality for graphics, sound, and user input handling.
- wxPython: A Python binding for wxWidgets, allowing developers to create native-looking GUI applications with a wide range of widgets and functionality.
- CustomTkinter: CustomTkinter is a python UI-library based on Tkinter, which provides new, modern and fully customizable widgets. They are created and used like normal Tkinter widgets and can also be used in combination with normal Tkinter elements.

In our project, we created a Graphical User Interface that has both approaches using Tkinter and CustomTkinter because of their compatibility and the clean, simple widgets they offer.

#### Home Page:

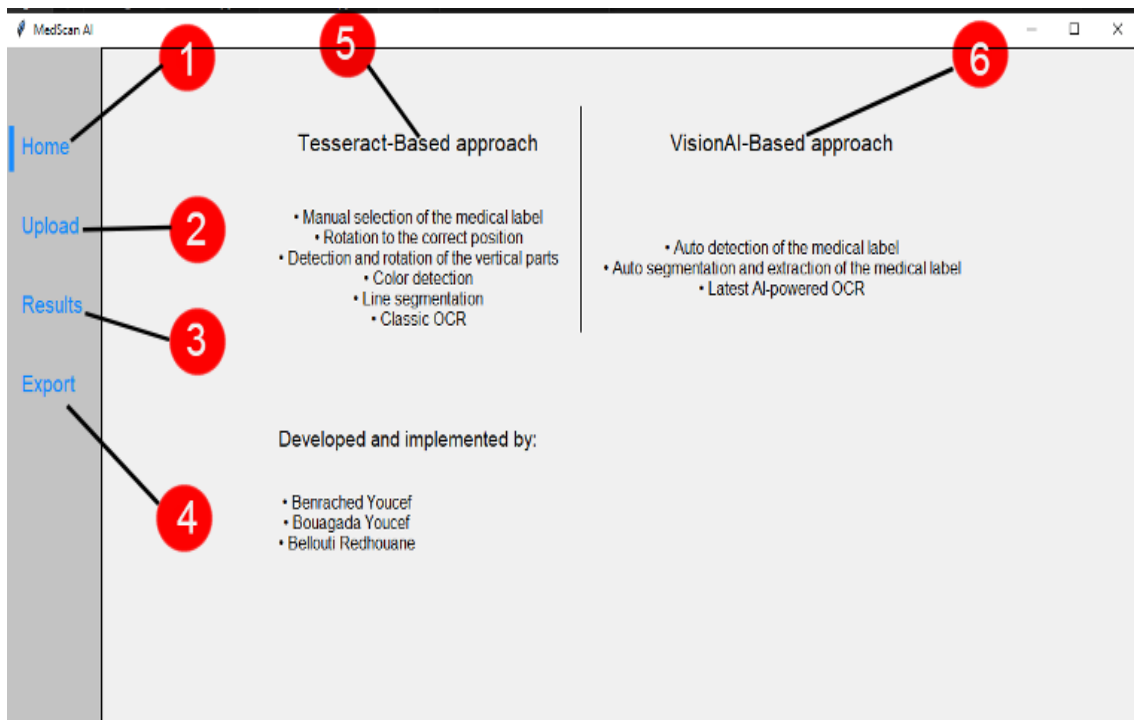


Figure 3.27: The home page of the GUI

The home page consists of the following features:

1. Home Button: for the main window that has some information about the 2 approaches.
2. Upload Button: for the upload window that allows the user to read and detect an image.
3. Results Button: for the window that contains the results of each method.
4. Export Button: for the export and save window that allows the user which information they want to save as a TXT or PDF format.
5. General information about the Tesseract-Based approach
6. General information about the VisionAI-Based approach

### Upload Page:

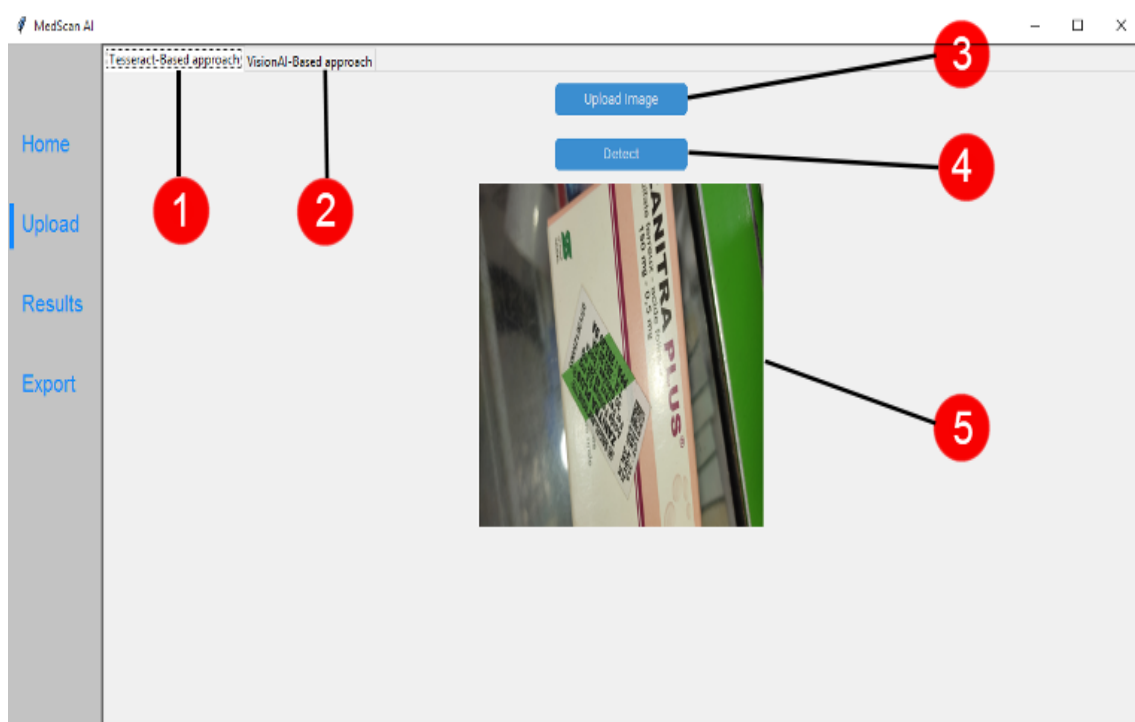


Figure 3.28: Upload Page

The Upload page consists of the following features:

1. Tesseract-Based approach Tab: the specified window for the tesseract-based approach.
2. VisionAI-Based approach Tab: the specified window for the VisionAI-Based approach.

3. Upload Image Button: the button to read the image that is being detected.
4. Detect Button: The button to start the detection process for each method.
5. The display of the image that is being detected.

### Results Page:

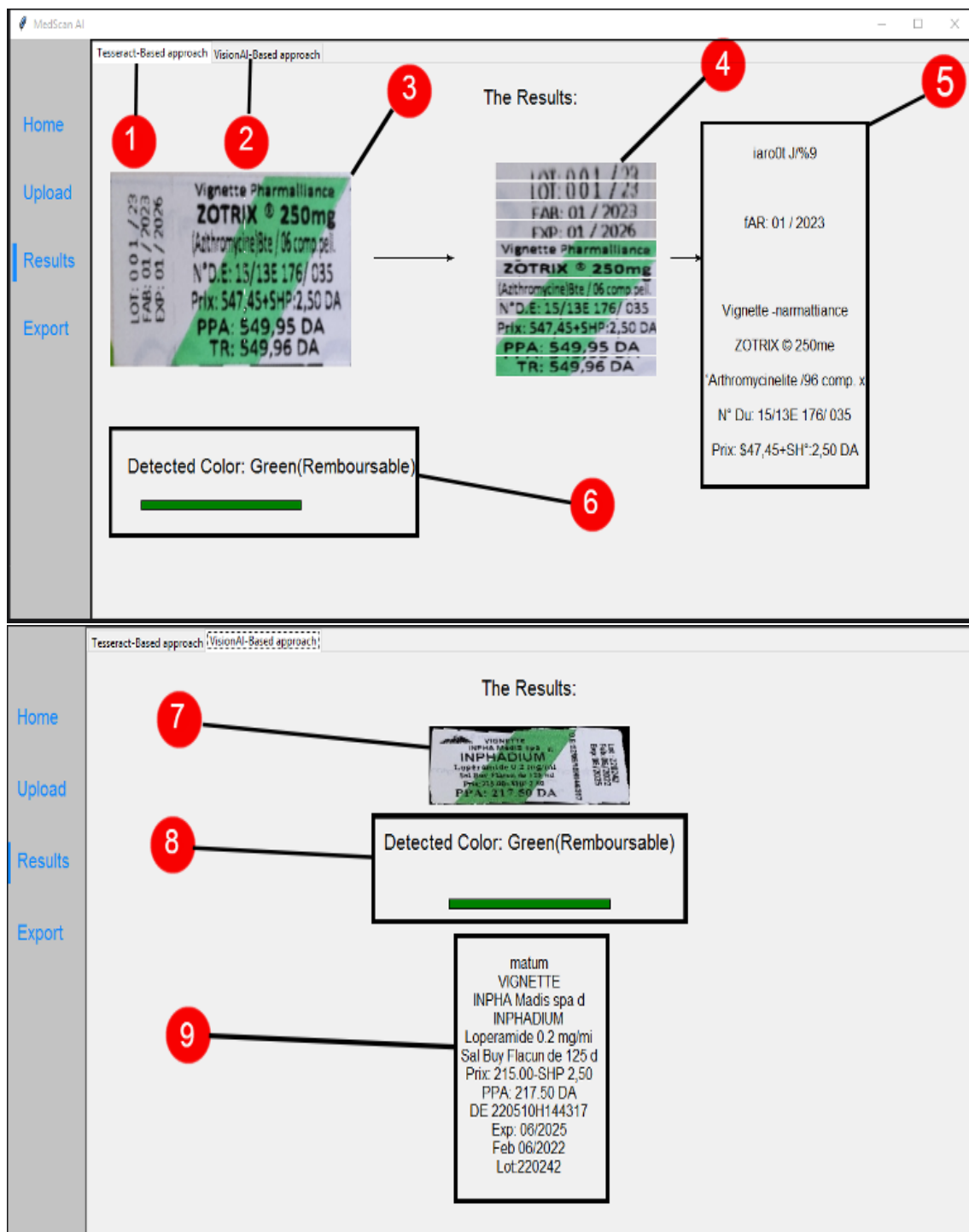


Figure 3.29: Results Page

The results page consists of the following features:

1. Tesseract-Based approach Tab: the specified window for the tesseract-based approach.
2. VisionAI-Based approach Tab: the specified window for the VisionAI-Based approach.
3. Result image from the Manual selection of the 4 corners.
4. Results of segmentation.
5. PyTesseract OCR results.
6. Detected color with the corresponding bar color.
7. Detected medical label from the custom-trained YOLOv8 model.
8. The detected color with the corresponding bar color.
9. VisionAI OCR results.

### Export Page:

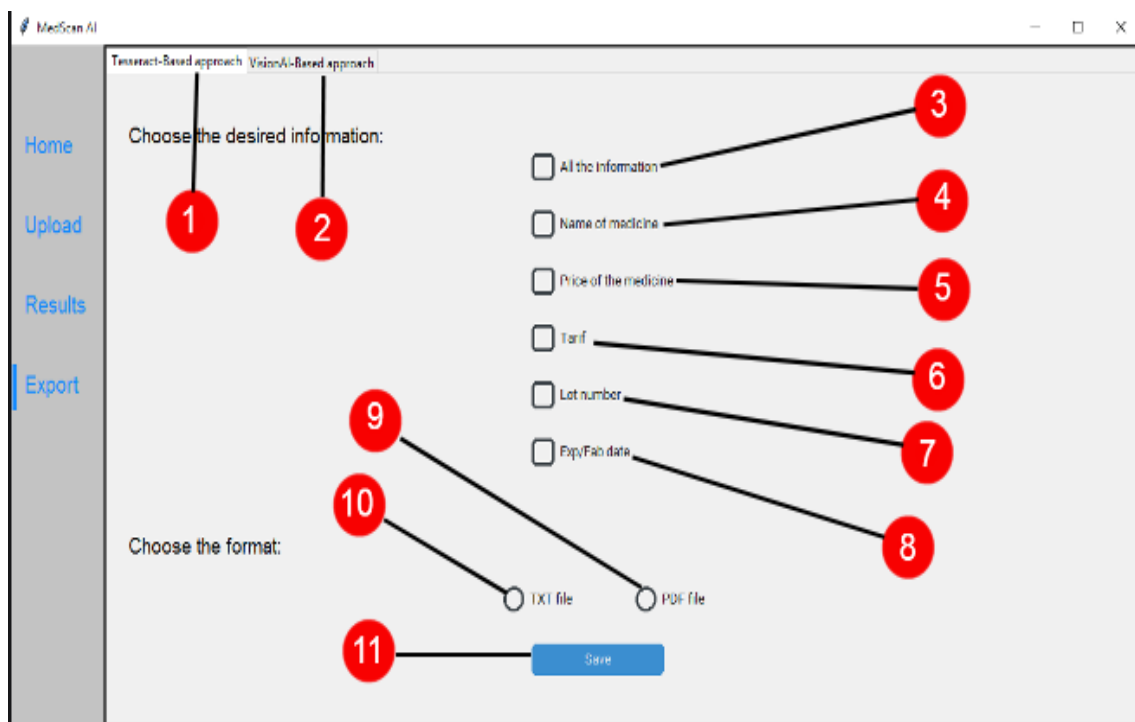


Figure 3.30: Export Page



The export page consists of the following features:

1. Tesseract-Based approach Tab: the specified window for the tesseract-based approach.
2. VisionAI-Based approach Tab: the specified window for the VisionAI-Based approach.
3. To save all the information detected of the medical label in the save file.
4. To add the name of the medicine in the save file.
5. To add the price of the medicine in the save file.
6. To add the reference interest rates (Tarif de références) in the save file.
7. To add the lot number in the save file.
8. To add the fabrication and expiration date in the save file.
9. Save the file as a .TXT file.
10. save the file as a .PDF file.
11. Save the file with the chosen information.

## 3.5 Experimentation and Results

### 3.5.1 Segmentation results

For the Tesseract-Based approach, we calculated the precision of the segmentation using the following formulation:

- $D$ : Number of lines detected correctly.
- $T$ : Number of Total lines in a medical label.

$$SegmentACC\% = \frac{D}{T} \times 100$$

After testing the segmentation on 40 different medical labels, we acquired an accuracy of **70.78%** and this is a result of the variation that the medical labels come in, such as the font, text positions and the way the labels are printed. So it is very difficult to cover all the different cases.



Figure 3.31: Example of a good segmentation

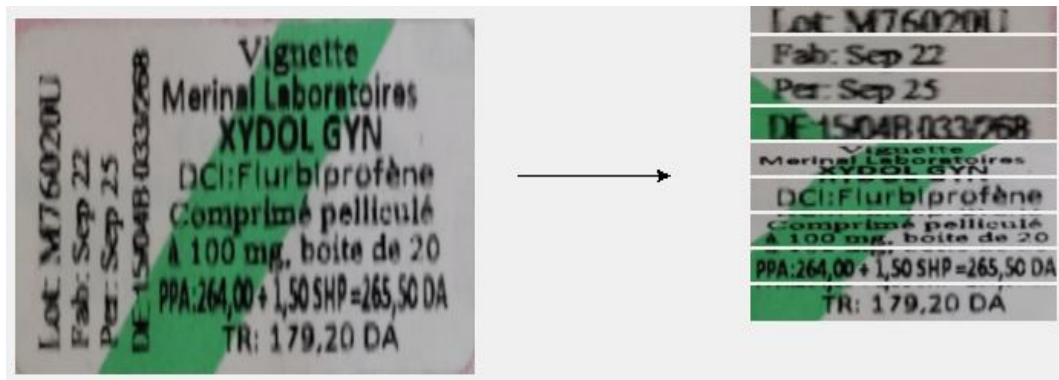


Figure 3.32: Example of a bad segmentation

As mentioned in the Segmentation Phase in the Tesseract-Based approach, the clockwise rotation of the vertical written parts is not always correct since some labels have texts that are written top to bottom and a clockwise rotation would rotate them in the wrong direction.

### 3.5.2 Medical Label detection results

As for the VisionAI-based approach, the YOLO model has a Confidence Score that indicates how sure the model is that the bounding box contains the object that is being predicted, the higher the score is- the greater the confidence in the answer. The Confidence Score is automatically calculated by the Model using the following formula:

$$C = \text{Pr}(\text{object}) * \text{IoU}$$

**Pr(Project):** the probability that an object is present in the cell.

**IoU:** Intersection over Union between the predicted box and the ground truth.

We predicted over 30 different medical labels and the average Confidence Score was **0.93** which means that the model is **93%** confident that its prediction is correct.



Figure 3.33: High Confidence score detection

The only issue that the Detection phase has is that it does not always rotate the image to the correct position. This is common for the flipped images, but the VisionAI OCR still works normally.

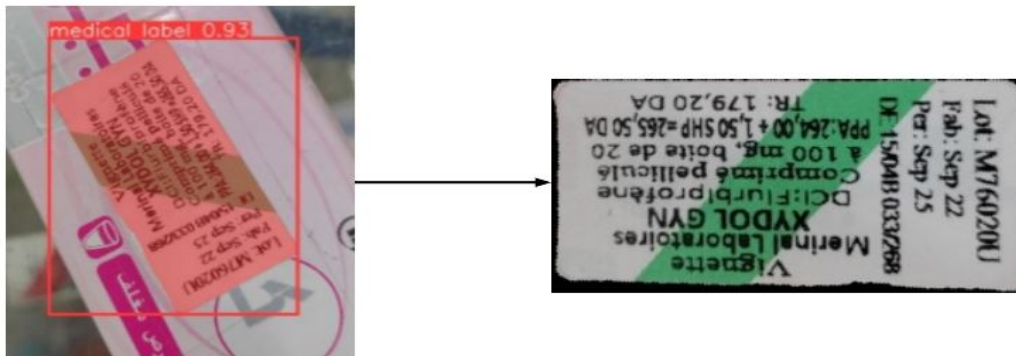


Figure 3.34: Example of a bad rotated medical label

### 3.5.3 Medical label character recognition results

In this part, we will present the character recognition results from each method then we will compare them to see which is more accurate and reliable in addition to understanding why that is the case.

We evaluated the 2 methods in terms of accuracy using the following formula:

$$Accuracy\% = \frac{\text{Number of characters recognised correctly} \times 100}{\text{Total number of characters in a medical label}}$$

And upon using this formula on 50 different labels, we acquired the following results:

	Tesseract-Based	VisionAI Based
OCR Accuaracy	52.68%	95.79%

Table 3.1: OCR accuracy results in both approaches

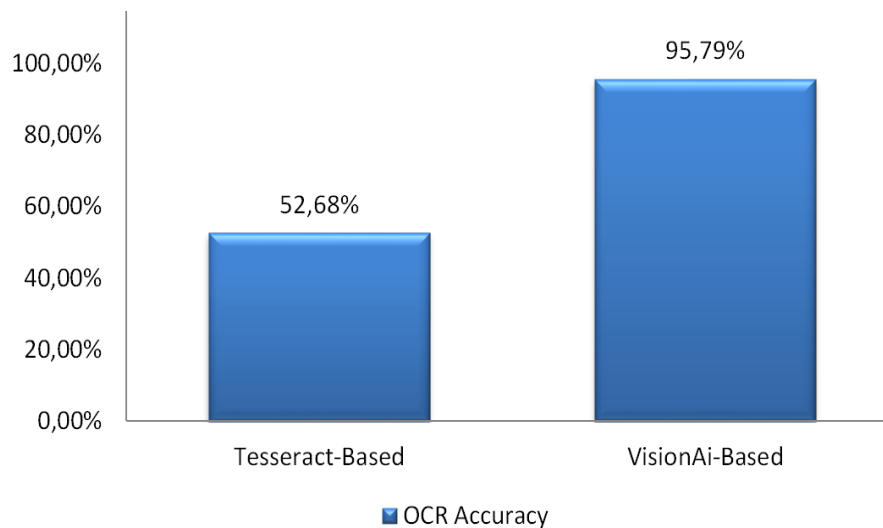
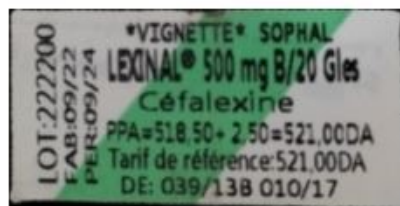


Figure 3.35: Comparison study of Accuracy

**VisionAI:**



LOT:222200  
FAB:09/22  
PER:09/24  
\*VIGNETTE SOPHAL  
LEXINAL 500 mg B/20  
Gles  
Cefalexine  
PPA=518,50+2,50=521,00DA  
Tarif de référence:521,00DA  
DE: 039/138 010/17

**Tesseract:**



—107:222700  
EFAB:09/22  
PER:09/24  
\*VIGNETTE\*® SOPHAL  
Cefalexine.  
PPA=518 50+ 2.50=521,00DA  
Tarif de référence:521,00DA  
DE: 0397138 010/17 —

Figure 3.36: OCR experiment results' comparison

### 3.5.4 Differences and Comparison

As noticed, there is a big difference between the 2 approaches in terms of the methods used and the results.

One of the biggest differences is the OCR results where the VisionAI-based approach has shown significant better results than the Tesseract-based approach.

And the reason for this difference is that Tesseract relies solely on the LSTM model to enhance the recognition which may not be enough for the different varieties the medical labels come in (Different fonts, Bad printing, etc). Whereas Vision-AI has more advanced and refined models that it uses to handle different varieties of complex images. Another difference between these OCR engines is that Tesseract is relatively a small project compared to Vision-AI which is the API of the app Google Lens. So even though both Vision-AI and Tesseract use AI-powered OCR engines, the results is not the same because of the advanced models Vision-AI has.

Another big difference between the 2 approaches is that the VisionAI-based approach uses an Intelligent detection of the medical label whereas the Tesseract-based approach uses a manual selection of the corners of the medical label. But as it was mentioned in the Medical Label Detection Results, the only issue is that the rotation in Vision-AI may not always be correct but it does not effect the OCR results since Vision-AI has advanced models that can understand that the image is rotated.

The following is a summary of the **Pros** and **Cons** of each approach:

#### **Tesseract-Based approach:**

- **Pros:**

- Correct rotation of the medical label.
- Fast execution.

- **Cons:**

- OCR accuracy rate above 50% but not very high.
- Heavily relies on the medical label font, quality, pixel intensity.
- May not always give correct segmentation.

#### **VisionAI-Based approach:**

- **Pros:**

- Precise and automatic detection of the medical label.
- Accurate OCR results.
- Uses an OCR engine powered by advanced and refined models.

- Uses Latest Object Detection model.
- **Cons:**
  - Occasional inaccurate rotation.
  - Slower execution.
  - Needs Internet access to function.

### 3.6 Conclusion

In this chapter, we introduced the approaches we implemented in our project and explained the methods used to achieve these results. We used a Tesseract-Based approach that gave acceptable OCR results in comparison with the VisionAI-based approach that which was more performant and reliable in terms of accuracy results because of its use of advanced and refined AI models that are designed for very complex images whereas Tesseract relies mainly on LSTM model for enhancing the results.

After evaluating the different approaches, we concluded that the VisionAI-based approach outperforms the Tesseract-Based approach. However, there are still certain areas where the Tesseract-Based approach excels and the VisionAI-based approach falls short.

# General Conclusion and perspectives

### General conclusion

In this project, we propose and develop an automated system utilizing Artificial Intelligence (AI) to detect, segment, and extract content from medical labels. Our work introduces two distinct approaches to address this task. The first approach incorporates traditional detection and segmentation techniques, employing classic image processing tools such as histograms. Additionally, we leverage PyTesseract, an OCR tool, to recognize and extract text from medical labels.

In the second approach, we present a custom-trained medical label detection and segmentation model that delivers accurate and reliable results. Furthermore, we employ VisionAI, one of the leading AI-powered OCR engines, which provides a comprehensive suite of tools beyond text recognition.

The experimental results and comparative analysis demonstrate the superior performance of the VisionAI-based approach. This is attributed to the utilization of YOLOv8 and VisionAI, two state-of-the-art AI engines trained on extensive datasets. The combination of these advanced technologies yields promising outcomes, further underscoring the efficacy of our proposed system for medical label detection and content extraction.

The contributions realised in our project can be summarized in the following points:

- Creating an accurate and performant Medical Label detection model using artificial intelligence.
- Recognising and extracting the information within the medical labels automatically.
- Implementing the proposed model and testing it.

### Perspectives

Finally, as for perspectives, We believe that our project can be drastically improved since it is only the starting point of a much bigger project that can automate the process between pharmacies and the CNAS and though we were able to detect medical labels using AI, we think that some points can be improved:

- Finding a better AI engine : With the rapid growth of Artificial intelligence, it is no surprise that the AI engines we used will be outdated and replaced with better AIs.
- Improving and optimizing the algorithm: The algorithm we developed can always be optimized and improved, thus we hope that the future work on the project will help optimise the algorithm.
- Develop a better looking GUI: We mainly used a basic library for the GUI (Tkinter) with some slight usage of a modern package (Custom Tkinter). We made this GUI to demonstrate how the methods work and we focused mostly on the back-end so the interface can be drastically improved.



# Bibliography

- [1] Python programming language.
- [2] Yasser Alginahi et al. Preprocessing techniques in character recognition. *Character recognition*, 1:1–19, 2010.
- [3] Jawad H AlKhateeb, Jianmin Jiang, Jinchang Ren, and S Ipson. Component-based segmentation of words from handwritten arabic text. *International Journal of Computer Systems Science and Engineering*, 5(1), 2009.
- [4] Boubkar Bakha, Houssam Kaiba, and Mokhtar Taffar. *Un Système OCR basé Apprentissage Profond pour la Reconnaissance des Vignettes de Médicaments*. Master thesis, Université jijel, 2019.
- [5] Abdel Belaïd. ‘analyse du document: de l’image à la représentation par les normes de codage’. *Document numérique*, 1(1):21–38, 1997.
- [6] A Bennasri, A Zahour, and B Taconet. Extraction des lignes d’un texte manuscrit arabe. In *Vision interface*, volume 99, pages 42–48, 1999.
- [7] Richard G Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 18(7):690–706, 1996.
- [8] Sushant Chandra, Saurav Sisodia, and Preeti Gupta. Optical character recognition-a review. *International Research Journal of Engineering and Technology (IRJET)*, 7(4):3037–3041, 2020.
- [9] Google Cloud, 2023.
- [10] Namrata Dave. Segmentation methods for hand written character recognition. *International journal of signal processing, image processing and pattern recognition*, 8(4):155–164, 2015.
- [11] David Doermann, Karl Tombre, et al. *Handbook of document image processing and recognition*, volume 1. Springer, 2014.
- [12] Tongkun Guan, Chaochen Gu, Jingzheng Tu, Xue Yang, Qi Feng, Yudi Zhao, and Wei Shen. Self-supervised implicit glyph attention for text recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15285–15294, 2023.

- [13] Gaurav Gupta, Shobhit Niranjan, Ankit Shrivastava, and R Mahesh K Sinha. Document layout analysis and classification and its application in ocr. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*, pages 58–58. IEEE, 2006.
- [14] Tong He, Weilin Huang, Yu Qiao, and Jian Yao. Text-attentional convolutional neural network for scene text detection. *IEEE transactions on image processing*, 25(6):2529–2541, 2016.
- [15] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International journal of computer vision*, 116:1–20, 2016.
- [16] YI Kazylina. Text recognition. *Modern machinery and technology: collection of reports of the XX International*, page 293, 2014.
- [17] Zhengchao Lei, Sanyuan Zhao, Hongmei Song, and Jianbing Shen. Scene text recognition using residual convolutional recurrent neural network. *Machine Vision and Applications*, 29:861–871, 2018.
- [18] Pratik Madhukar Manwatkar and Shashank H Yadav. Text recognition from images. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–6. IEEE, 2015.
- [19] Lukáš Neumann and Jiří Matas. Real-time scene text localization and recognition. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3538–3545. IEEE, 2012.
- [20] Erhan Öztop, Adem Yasar Mülayim, Volkan Atalay, and Fatos Yarman-Vural. Repulsive attractive network for baseline extraction on document images. *Signal Processing*, 75(1):1–10, 1999.
- [21] Chirag Patel, Atul Patel, and Dharmendra Patel. Optical character recognition by open source ocr tool tesseract: A case study. *International Journal of Computer Applications*, 55(10):50–56, 2012.
- [22] Jayashree R Prasad, Uday V Kulkarni, and Rajesh S Prasad. Template matching algorithm for gujrati character recognition. In *2009 Second International Conference on Emerging Trends in Engineering & Technology*, pages 263–268. IEEE, 2009.
- [23] Joan Andreu Sánchez, Vicent Bosch, Verónica Romero, Katrien Depuydt, and Jesse De Does. Handwritten text recognition for historical documents in the transcriptorium project. In *Proceedings of the first international conference on digital access to textual cultural heritage*, pages 111–117, 2014.
- [24] CS Shin, KI Kim, MH Park, and Hang Joon Kim. Support vector machine-based text detection in digital video. In *Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No. 00TH8501)*, volume 2, pages 634–641. IEEE, 2000.

- [25] Archana A Shinde and DG Chougule. Text pre-processing and text segmentation for ocr. *International Journal of Computer Science Engineering and Technology*, 2(1):810–812, 2012.
- [26] Bolan Su and Shijian Lu. Accurate scene text recognition based on recurrent neural network. In *Computer Vision–ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part I 12*, pages 35–48. Springer, 2015.
- [27] O.J. Tobias and R. Seara. Image segmentation by histogram thresholding using fuzzy sets. *IEEE Transactions on Image Processing*, 11(12):1457–1465, 2002.
- [28] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *2011 International conference on computer vision*, pages 1457–1464. IEEE, 2011.
- [29] Kai Wang and Serge Belongie. Word spotting in the wild. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part I 11*, pages 591–604. Springer, 2010.