République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique


École nationale supérieure d'informatique
08 Mai 1945 - Sidi Bel Abbès -

**Interdisciplinary project of the second year of the second cycle.**

**Option: IASD**

---

# Detection and Extraction of Textual Information from Pharmaceutical Labels

---

*Project undertaken by :*

Mohamed Amine Lasheb
Nazim Rachid Benbetka
Walid Abdallaoui
Meriem Khedir
Rabah Aouar

*Supervised by :*

Monsieur Chaib Souleiman

## Summary

The current Algerian pharmaceutical strategy is based on manually entering the characteristics of medications mentioned on labels, followed by visual verification of electronic invoices/receipts using these printed labels. This process increases the execution time of these tasks and the risk of errors.

To enhance management rigor in pharmacies we propose the implementation of an artificial intelligence system. This system is designed for the recognition and automatic extraction of information from pharmaceutical labels based on captured or scanned images.

We will utilize deep learning techniques to detect labels on medication boxes and extract the text from these labels. We will incorporate the fuzzy search as a verification step to ensure the accuracy of the extracted information.

Our system is a desktop application designed to efficiently manage pharmacy stock. It offers the capability to utilize a webcam for scanning medication boxes, extracting text from the labels, and automatically adding the medication to the inventory. Additionally, our system incorporates various other functionalities to enhance the overall management process.

# Contents

# Chapter 1

# Introduction

In this chapter, we will delve into a detailed analysis of the problem faced by the Algerian pharmaceutical strategy, highlighting the limitations and challenges. We will also discuss the motivation behind our proposed solution, emphasizing the need for an automated system that can streamline the process and improve management rigor in pharmacies. Finally, we will present our proposed solution, outlining the key components and techniques that will be further detailed in subsequent sections.

## 1.1 Problem Statement and Challenges

The current approach of manually entering medication characteristics from labels and visually verifying electronic invoices/receipts presents several challenges.
Firstly, the process consumes a significant amount of time, resulting in delays and inefficiencies in pharmacy operations.
Secondly, the risk of errors during manual entry and visual verification introduces potential discrepancies and inaccuracies in the recorded data.
Additionally, the dependence on human involvement for these tasks makes the overall process susceptible to human error and inconsistency.
Another challenge lies in the management of pharmaceutical stock. Without an automated system to extract information from labels, inventory management becomes more cumbersome, making it difficult to track medication quantities accurately and monitor expiration dates. This can lead to potential wastage, stock-outs, and compromised patient safety.

## 1.2 Motivation

The limitations of the current approach, along with the potential risks and inefficiencies it introduces, motivate the need for an alternative solution
By implementing an artificial intelligence system for automatic label recognition and information extraction, we aim to streamline and improve pharmaceutical operations in Algeria.
The proposed system will significantly reduce the time required for data entry, minimize the

risk of errors, and enhance overall management rigor in pharmacies.

## 1.3 Proposed Solution

Our proposed solution revolves around developing an artificial intelligence system in the form of a desktop application. This application is designed to effectively recognize and extract information from pharmaceutical labels.
By leveraging deep learning techniques, such as image recognition and optical character recognition (OCR), the system will automatically capture and extract the relevant information from captured or scanned label images. This automated process eliminates the need for manual data entry, reduces errors, and improves the efficiency of information retrieval.
Additionally, to ensure the accuracy of the extracted information, we will incorporate techniques of verification like fuzzy search, user ability to edit the extracted information ...ext. This will help cross-reference the extracted data with a predefined database of medication characteristics, further enhancing the reliability of the system.

## 1.4 Project Objectives

The primary aim of our project is to develop an efficient and accurate system for text extraction from pharmaceutical labels in Algerian pharmacies. To achieve this, we have defined the following specific objectives:

1. **Automate Text Extraction:** Our first objective is to automate the process of extracting information from pharmaceutical labels. By utilizing advanced computer vision techniques, such as object detection and optical character recognition (OCR), we aim to develop an automated system that can accurately locate and extract relevant text data from label images.

2. **Ensure Accuracy and Reliability**: We strive to ensure the accuracy and reliability of the extracted information. Our objective is to minimize errors and discrepancies by implementing robust algorithms and verification mechanisms. Techniques such as fuzzy matching and cross-referencing with existing databases will be employed to enhance the accuracy of the extracted text.

3. **Improve Efficiency and Productivity**: Our goal is to significantly improve the efficiency and productivity of pharmacy operations. By automating the text extraction process, we aim to reduce the time and effort required for manual data entry and verification. This will enable pharmacy staff to allocate more time to critical tasks, such as patient care and medication management.

4. **Enhance User-Friendliness**: Another objective is to create a user-friendly system that can be easily adopted by pharmacy staff. We will focus on developing an intuitive user

interface, providing clear instructions for capturing label images, and ensuring seamless integration with existing pharmacy management systems. Usability testing and user feedback will guide our efforts to enhance the overall user experience.

5. **Promote Wide Adoption**: We aim to develop a system that can be widely adopted in Algerian pharmacies. Our objective is to create a scalable and cost-effective solution that can be easily implemented in various pharmacy settings, ranging from small local pharmacies to larger healthcare facilities. By addressing the specific needs and requirements of the Algerian pharmaceutical industry, we aim to promote the widespread use of our system.

## 1.5 Summary and Conclusion

In conclusion, this chapter provided an in-overview of the problem we aim to address, highlighting the challenges faced in the domain of pharmaceutical label recognition and extraction. We presented our motivation for developing an automated system and introduced our proposed solution. The chapter also outlined the specific objectives of our project.
By understanding the problem landscape and setting clear goals, we are well-equipped to proceed with the implementation and evaluation of our solution.

# Chapter 2

# Background

## 2.1 Introduction

This chapter sets the foundation for our project by exploring key definitions in the field of Artificial Intelligence (AI) and delving into specific concepts such as Deep Learning, Convolutional Neural Networks (CNNs), Transfer Learning, as well as the remarkable techniques of YOLO (You Only Look Once) and OCR (Optical Character Recognition).

We aim to provide a comprehensive understanding of these fundamental terms and methodologies, serving as a starting point for our subsequent discussions and analyses. By elucidating these concepts, we establish a solid framework for the subsequent chapters, where we will delve into the practical application and implementation of these cutting-edge technologies.

## 2.2 Artificial Intelligence

Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building intelligent machines capable of performing tasks that typically require human intelligence. While AI is an interdisciplinary science with multiple approaches, advancements in machine learning and deep learning, in particular, are creating a paradigm shift in virtually every sector of the tech industry.

Artificial intelligent systems can perform tasks commonly associated with human cognitive functions — such as interpreting speech, playing games, and identifying patterns. They typically learn how to do so by processing massive amounts of data and looking for designs to model in their own decision-making. In many cases, humans will supervise an AI's learning process, reinforcing good decisions and discouraging bad ones. But some AI systems are designed to learn without supervision — for instance, by playing a video game repeatedly until they eventually figure out the rules and how to win.

In our project, we will leverage artificial intelligence technologies to construct a system that automates the tasks traditionally performed by pharmacy workers.

## 2.3 Deep Learning

Deep learning is a sub-field of machine learning that focuses on the development and application of **artificial neural networks** with multiple layers, enabling the model to learn hierarchical representations of data. It is inspired by the structure and function of the human brain, where neurons are interconnected to process information.

In deep learning, these neural networks consist of numerous layers, allowing them to automatically learn and extract complex features from raw input data. The networks learn to recognize patterns, make predictions, and perform tasks by adjusting the weights and biases of the interconnected neurons through a process called training.

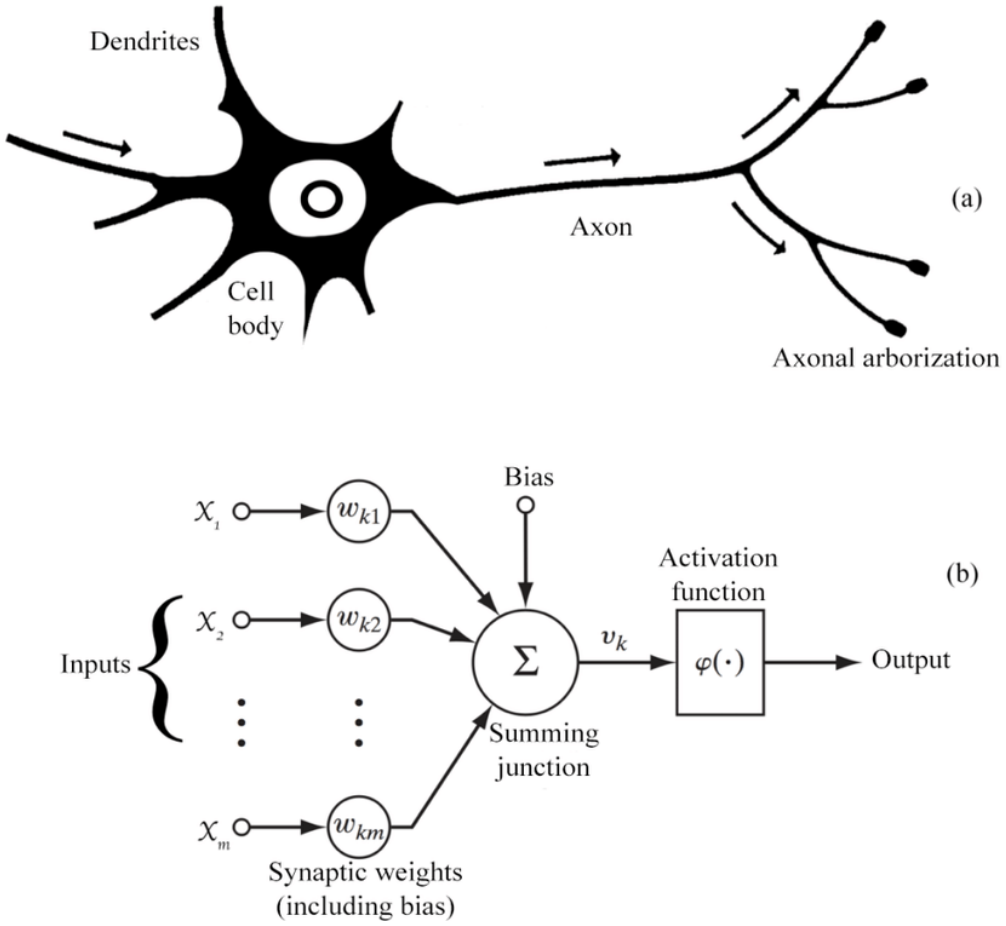


Figure 2.1: Similarity between biological and artificial neural networks [7]

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention, aligning with the goals of our project, which involves label segmentation and text extraction automatically without any human involvement.

## 2.4 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a type of artificial neural network that is commonly used in computer vision tasks such as image classification, object detection, and image segmentation.

The key difference between CNNs and other types of neural networks is that CNNs have convolutional layers, which apply filters to the input image to extract features that are relevant to the task at hand.

The first layer in a CNN is typically a convolutional layer that applies a set of learnable filters to the input image to create a set of feature maps. These feature maps capture different aspects of the input image, such as edges, textures, and patterns.

After the convolutional layers, the output is typically passed through a series of pooling layers, which downsample the feature maps to reduce the dimensionality of the data and make it easier to process. Finally, the output of the last layer is typically passed through one or more fully connected layers, which use the learned features to make a prediction about the input data.

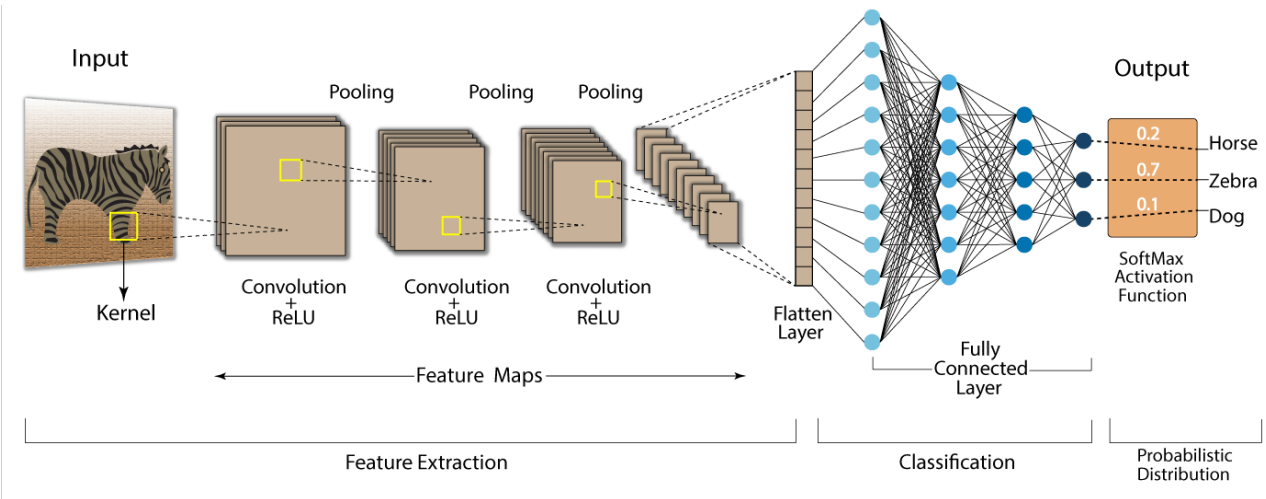


Figure 2.2: Convolutional neural networks basic architecture [2]

### 2.4.1 CNN Architecture [1]

#### 2.4.1.1 LeNet

LeNet is the first CNN architecture. It was developed in 1998 by Yann LeCun, Corinna Cortes, and Christopher Burges for **handwritten digit recognition problems**.
It is one of the earliest and most widely-used CNN architectures and has been successfully applied to tasks such as handwritten digit recognition.
The LeNet architecture consists of multiple convolutional and pooling layers, followed by a fully-connected layer. The model has five convolution layers followed by two fully connected layers.

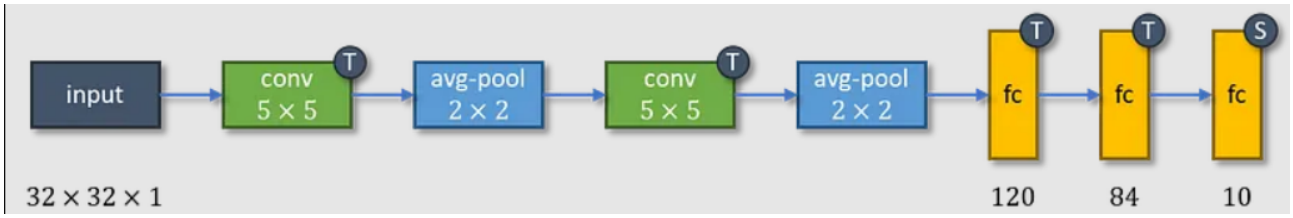


Figure 2.3: LeNet architecture

#### 2.4.1.2 AlexNet

AlexNet is the architecture that popularized CNN. It was developed by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. AlexNet had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other.
AlexNet is composed of 5 convolutional layers with a combination of max-pooling layers, 3 fully connected layers, and 2 dropout layers. The activation function used in all layers is Relu. The activation function used in the output layer is Softmax. The total number of parameters in this architecture is around 60 million.

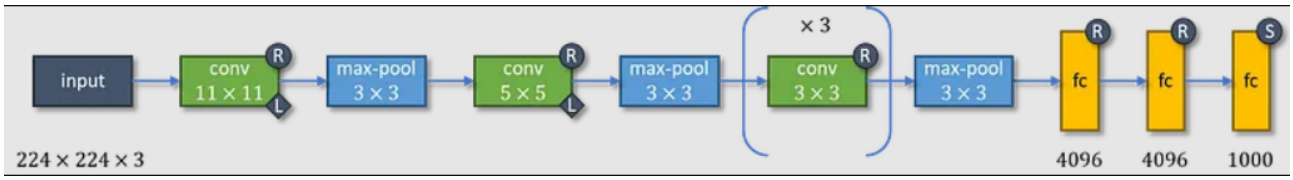


Figure 2.4: AlexNet architecture

#### 2.4.1.3 VGGNet

VGG is one of the biggest networks that has 138 million parameters. it was developed by Karen Simonyan, Andrew Zisserman et al. at Oxford University.
Just like AlexNet, the last layer is equipped with a softmax activation function and all others

are equipped with ReLU. The 2nd, 4th, 7th, 10th, and 13th convolution layers are followed by a 2×2 max-pooling.
Default VGG-16 accepts colored images with dimensions 224×224 and outputs one of the 1000 classes.

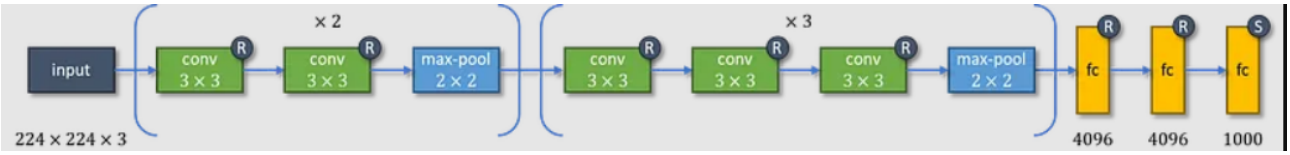


Figure 2.5: VGG-16 architecture

#### 2.4.1.4 GoogleNet

GoogLeNet is the CNN architecture used by Google to win ILSVRC 2014 classification task. It was developed by Jeff Dean, Christian Szegedy, Alexandro Szegedy, et al..
It has been shown to have a notably reduced error rate in comparison with previous winners AlexNet (Ilsvrc 2012 winner) and ZF-Net (Ilsvrc 2013 winner). In terms of error rate, the error is significantly lesser than VGG (2014 runner-up).
It achieves deeper architecture by employing a number of distinct techniques, including 1×1 convolution and global average pooling. GoogleNet CNN architecture is computationally expensive. To reduce the parameters that must be learned, it uses heavy unpooling layers on top of CNNs to remove spatial redundancy during training and also features shortcut connections between the first two convolutional layers before adding new filters in later CNN layers.

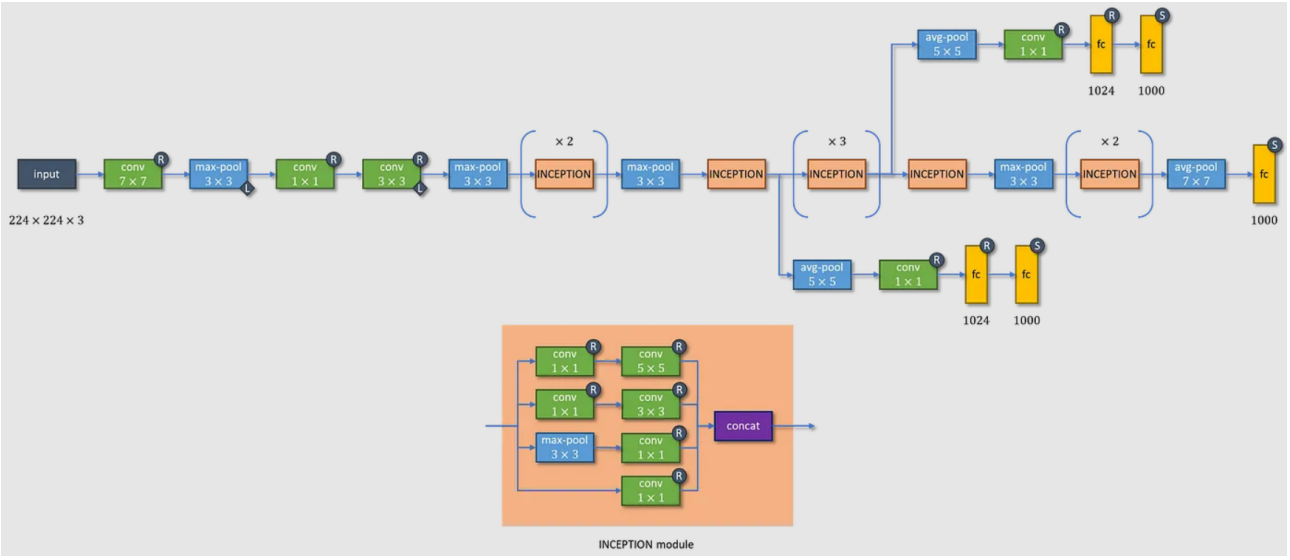


Figure 2.6: GoogleNet architecture

#### 2.4.1.5 ResNet

ResNet is the CNN architecture that was developed by Kaiming He et al. to win the ILSVRC 2015 classification task.
they are characterized by :

- **Identity block:** consists of 3 convolution layers with 1×1, 3×3, and 1×1 kernel sizes, all of which are equipped with BN. The ReLU activation function is applied to the first two layers, while the input of the identity block is added to the last layer before applying ReLU.
- **Convolution block:** same as identity block, but the input of the convolution block is first passed through a convolution layer with 1×1 kernel size and BN before being added to the last convolution layer of the main series.

The network has 152 layers and over one million parameters, which is considered deep even for CNNs because it would have taken more than 40 days on 32 GPUs to train the network on the ILSVRC 2015 dataset.

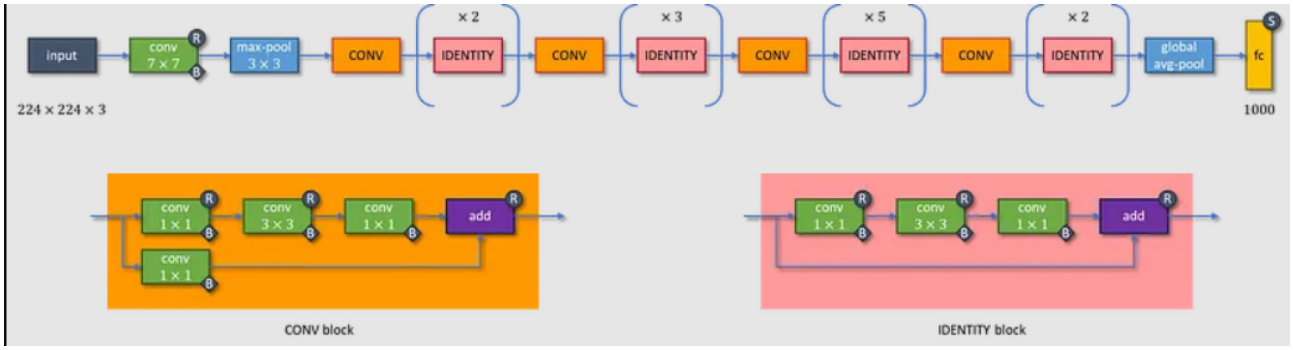


Figure 2.7: ResNet architecture

## 2.5 Image Understanding

### 2.5.1 Definition

An image refers to a visual representation of an object, scene, or concept captured through various imaging devices, such as cameras or sensors.
It consists of a collection of pixels arranged in a grid-like structure, where each pixel represents a specific color or intensity value. For example, in gray-scale images, each pixel is represented by a single value ranging from 0 (black) to 255 (white). In color images, pixels are represented as a combination of three color channels (Red, Green, and Blue), and each channel value typically ranges from 0 to 255.
Properties of Images:

- **Resolution:** Refers to the number of pixels in an image, determining the level of detail and clarity. Higher-resolution images have more pixels and thus offer finer details.
- **Dimensions:** Images have width and height dimensions, representing the number of pixels horizontally and vertically, respectively.
- **Color Space:** Describes the representation of colors in an image. Common color spaces include RGB (Red, Green, Blue), CMYK (Cyan, Magenta, Yellow, Black), and grayscale.
- **Bit Depth:** Represents the number of bits used to store each pixel value. It determines the range of colors or grayscale levels that can be represented. For example, 8-bit depth allows 256 shades of gray or colors.
- **Aspect Ratio:** The ratio of an image's width to its height. It describes the proportional relationship between the image's dimensions.
- **Compression:** Images can be compressed to reduce their file size while preserving visual quality. Compression algorithms remove redundant or irrelevant information, resulting in lossy or lossless compression.
- **Metadata:** Images can contain additional information like date, time, camera settings, and geolocation, stored as metadata.

### 2.5.2 Image Processing Techniques

In artificial intelligence (AI), the treatment of images is a critical component, particularly in computer vision applications. Filtering and morphological operations are two fundamental techniques used for image processing and enhancement, enabling noise reduction, feature extraction, and overall image improvement.

#### 2.5.2.1 Image filtering

Filtering involves the application of mathematical operations to images, allowing for targeted modifications or enhancements.
This technique plays a crucial role in **noise reduction, feature extraction,** and general image enhancement. Commonly used filters in AI include **the Gaussian filter** for noise reduction,**the median filter** for noise removal through pixel replacement,**the Sobel filter** for edge detection, and **the Laplacian filter** for highlighting fine details.

1. **Gaussian Filter:[10]**

   The Gaussian filter is a smoothing filter that reduces noise and blurs the image while preserving important details. It achieves this by convolving the image with a Gaussian kernel, which assigns more weight to the central pixels and gradually decreases the weight for the surrounding pixels. This results in a smoother image with reduced high-frequency noise.

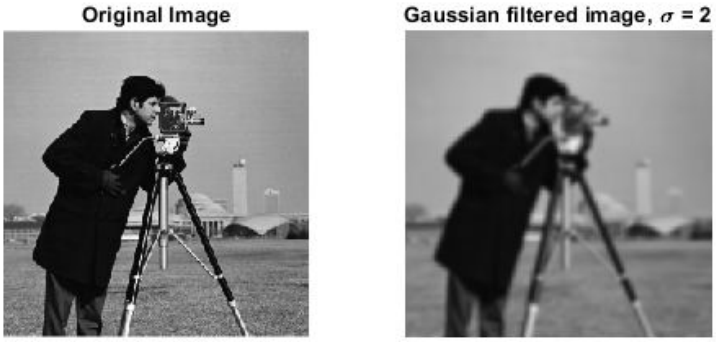


Figure 2.8: Gaussian Filtering

2. **Median Filter [11]:**

   The median filter is a non-linear filter used for removing noise in an image. It replaces each pixel value with the median value of its neighboring pixels. By considering the median instead of the mean, the median filter effectively reduces impulse or salt-and-pepper noise, where individual pixels have extreme values that differ significantly from their neighbors. It preserves edges and fine details better than linear filters.

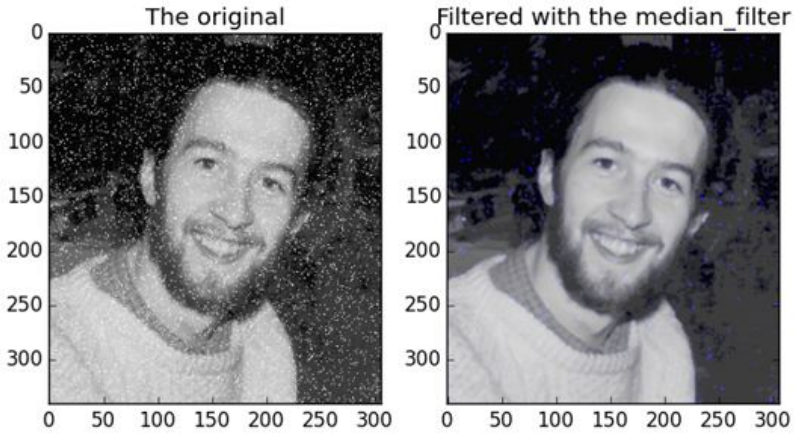


Figure 2.9: Median Filtering

3. **Sobel Filter[15]:**

   The Sobel filter is an edge detection filter that highlights intensity gradients in an image. It uses a pair of convolution kernels (one for horizontal and one for vertical gradients) to approximate the derivative of the image intensity. By computing the gradient magnitude and orientation at each pixel, the Sobel filter can detect edges and emphasize their presence in the image. This is particularly useful for tasks like edge detection, contour extraction, and feature detection.

Figure 2.10: Sobel Filtering

4. **Laplacian Filter[3]:**

   The Laplacian filter is a sharpening filter used to enhance edges and detect fine details in an image. It calculates the second derivative of the image intensity, emphasizing areas of rapid intensity change. The Laplacian filter enhances edges by highlighting regions where the intensity changes abruptly, resulting in enhanced edge sharpness and improved visibility of fine details. It is commonly used in image processing for tasks like edge enhancement and feature extraction.

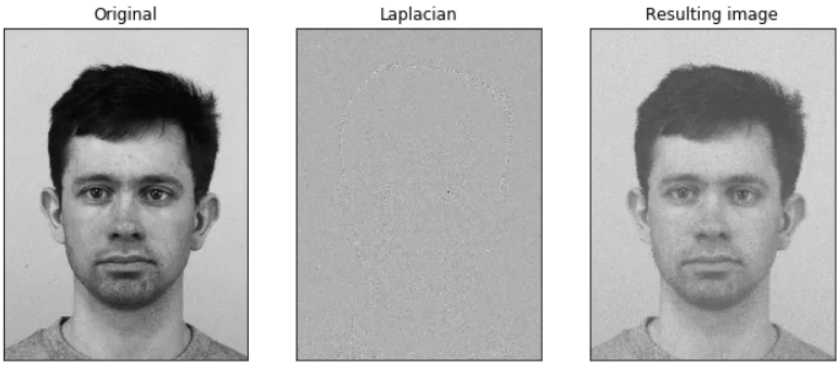


Figure 2.11: The result of adding the Laplacian of an image to the original image

#### 2.5.2.2 Morphological Operations

Morphological operations[4] are a set of image-processing techniques used to analyze and manipulate the shape and structure of objects within an image. These operations are based on mathematical morphology, a theory developed for the analysis of spatial structures.
The primary goal of morphological operations is to **extract meaningful information, refine image structures, remove noise, segment objects**, and perform various other tasks related to image analysis and computer vision. Let's explore the two fundamental morphological operations:**dilation** and **erosion**.

1. **Dilation:**

   Dilation is a morphological operation that expands the boundaries of objects in an image. It works by sliding a structuring element (also known as a kernel or a structuring function) over the image and replacing each pixel with the maximum value within the corresponding neighborhood defined by the structuring element. This causes objects to grow or expand.

   Dilation is useful for tasks such as:

   - **Enlarging objects**: By applying dilation, you can make objects larger, fill gaps, and connect broken parts.
   - **Filling holes**: Dilation can be used to fill small holes within objects, making them more solid.
   - **Merging adjacent objects**: When adjacent objects have narrow separations, dilation can help merge them into a single object.

2. **Erosion:**

   Erosion is the opposite of dilation and is used to shrink the boundaries of objects in an image. It involves sliding a structuring element over the image and replacing each pixel with the minimum value within the corresponding neighborhood. Erosion can remove

small details, thin out objects, and separate connected components.

Erosion is useful for tasks such as:

- **Removing noise:**: Small noisy regions can be eliminated by eroding them away.
- **Isolating objects**: Erosion can separate connected objects or break them apart when they are too close to each other.
- **Extracting boundaries**: By subtracting the eroded image from the original, you can obtain the boundary or contour of objects.

In addition to dilation and erosion, morphological operations also include advanced techniques like opening and closing:

3. **Opening:**

   An opening is an erosion operation followed by a dilation. It is useful for removing noise and small objects while preserving the overall shape and structure of larger objects.

4. **Closing:**

   Closing is a dilation operation followed by erosion. It helps fill small gaps in objects and smoothens object boundaries.

## 2.6 Transfer learning

### 2.6.1 Definition

Transfer learning is a machine learning technique that leverages knowledge learned from one task or domain to improve performance on a different but related task or domain. In transfer learning, a pre-trained model, which has been trained on a large dataset for a specific task, is used as a starting point for a new task.

The idea behind transfer learning is that the knowledge and representations learned by a model on one task can be valuable in solving a different task. Instead of training a model from scratch on a new task, transfer learning allows us to utilize the knowledge and features learned from previous tasks, saving computational resources and time.

The process typically involves taking a pre-trained model, removing or freezing some of its layers, and adding new layers specific to the target task. These new layers are then trained on a smaller dataset specific to the new task. By starting with pre-trained weights, the model can capture general features and patterns that are transferable across tasks, thus improving performance and convergence speed on the new task.

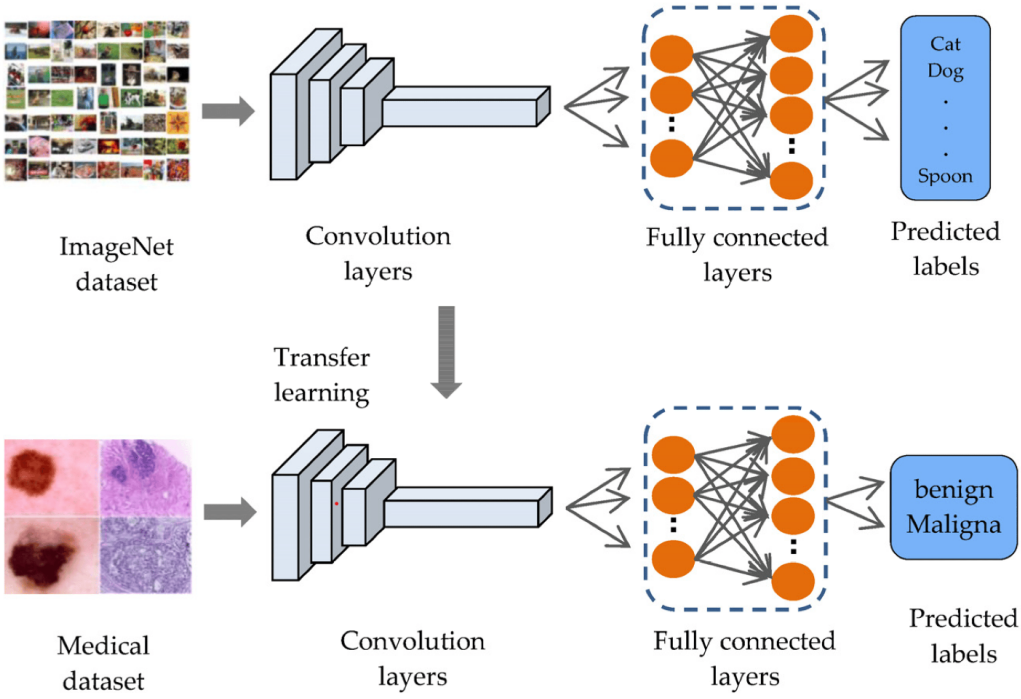


Figure 2.12: Transfer Learning[13]

### 2.6.2 Transfer Learning: Steps for Adapting Pretrained Models to New Domains

This section provides an overview of the transfer learning process, outlining the sequential steps involved in adapting pre-trained models to new domains.

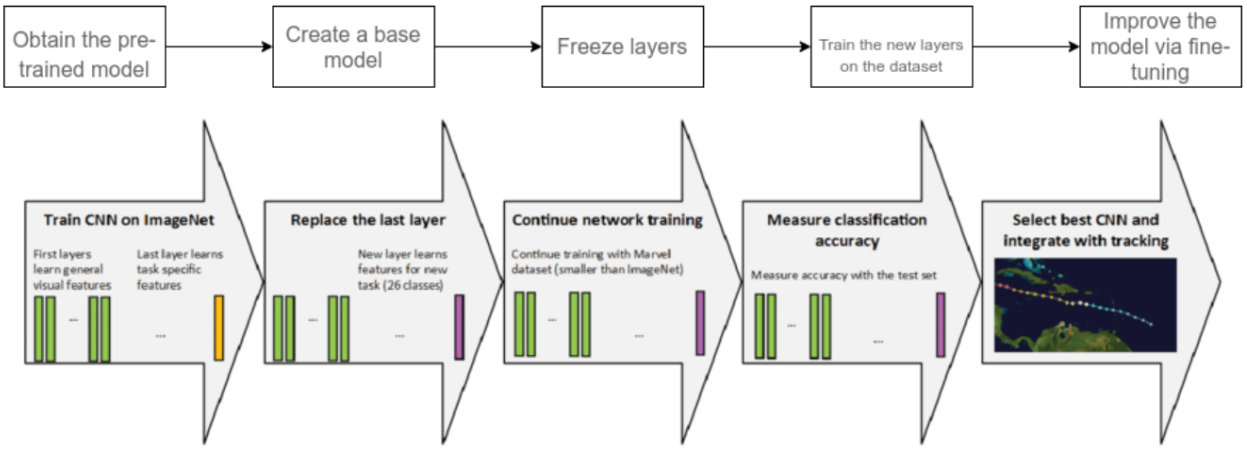


Figure 2.13: Transfer Learning steps[16]

1. **Select a Pre-Trained Model:**

   Choose a pre-trained model that has been trained on a large dataset for a related task. The choice of model depends on the specific problem domain and the available resources.

   Here are some of the pre-trained models you can use:

   (a) ImageNet Models:

   - VGG (e.g., VGG16, VGG19)
   - ResNet (e.g., ResNet50, ResNet101, ResNet152)
   - Inception (e.g., InceptionV3, InceptionResNetV2)

   (b) Natural Language Processing (NLP) Models::

   - Word2Vec
   - GloVe
   - BERT (Bidirectional Encoder Representations from Transformers)
   - Transformer-XL

   (c) Object Detection Models:

   - Faster R-CNN
   - YOLO (You Only Look Once)

2. **Create a Base Model:**

   The base model is one of the architectures that we have selected in the first step to be in close relation to our task.

   We can either download the network weights which saves the time of additional training of the model. Else, we will have to use the network architecture to train our model from scratch.

   There can be a case where the base model will have more neurons in the final output layer than we require in our use case. In such scenarios, we need to remove the final output layer and change it accordingly.

3. **Freeze Layers:**

   Freezing the starting layers from the pre-trained model is essential to avoid the additional work of making the model learn the basic features.
   If we do not freeze the initial layers, we will lose all the learning that has already taken place.

4. **Add new Trainable Layers:**

   The only knowledge we are reusing from the base model is the feature extraction layers. We need to add additional layers on top of them to predict the specialized tasks of the model. These are generally the final output layers.

5. **Train the New Layers:**

   The pre-trained model's final output will most likely differ from the output we want for our model. For example, pre-trained models trained on the ImageNet dataset will output 1000 classes.
   However, we need our model to work for two classes. In this case, we have to train the model with a new output layer in place.

6. **Fine-tune the Model:**

   One method of improving the performance is fine-tuning.
   Fine-tuning involves unfreezing some parts of the base model and training the entire model again on the whole dataset at a very low learning rate. The low learning rate will increase the performance of the model on the new dataset while preventing overfitting.

## 2.7 Object Detection

Object detection is a computer vision task that involves identifying and localizing objects within an image or video. The goal of object detection is to determine the presence of objects in an image, classify them into predefined categories, and provide bounding box coordinates that indicate their locations.
Object detection combines two key components: prediction and segmentation.

1. **Object Prediction:**
   Object prediction in the context of object detection refers to identifying and classifying objects present in an image or video frame. The prediction step involves analyzing the visual content of the input and assigning labels or categories to the detected objects. This process typically employs machine learning techniques, such as deep neural networks, that are trained on large datasets to learn and recognize patterns associated with different objects.

2. **Object Segmentation:**
   Object segmentation, also known as instance segmentation, goes beyond object prediction by not only identifying objects but also precisely delineating their boundaries at the pixel level. Segmentation provides a more detailed understanding of object shapes and enables the separation of overlapping instances or objects of the same class within an image. It assigns a unique label or identifier to each pixel, indicating which object it belongs to.

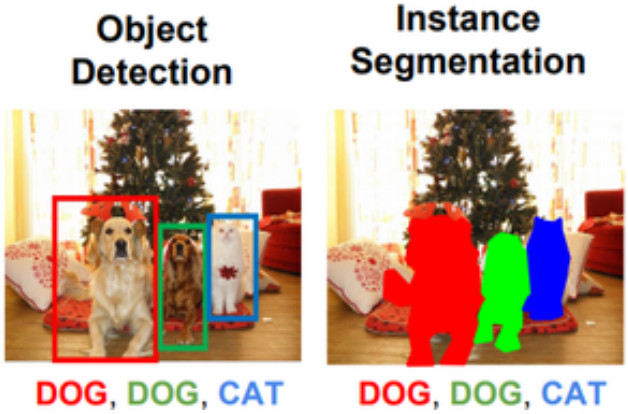


Figure 2.14: Object prediction and object segmentation[12]

### 2.7.1 Object Detection Pretrained Models

Object detection models are typically trained on large-scale annotated datasets, such as COCO (Common Objects in Context), PASCAL VOC (PASCAL Visual Object Classes), or Open

Images, to learn object representations and capture relevant features. They can be fine-tuned on specific datasets or tasks using **transfer learning techniques**, allowing for efficient adaptation to new domains or object categories
These models leverage **convolutional neural networks (CNNs)** as their backbone to extract meaningful features from input images. They employ various techniques like anchor boxes, region proposals, multi-scale feature maps, and specific loss functions to improve accuracy, speed, and robustness in object detection.

There are several popular object detection models, each with its own architecture and characteristics. Some of the commonly used object detection models include:

1. Region-based Convolutional Neural Networks (R-CNN):
   - R-CNN: The original R-CNN introduced the concept of region proposals to identify potential object locations in an image. It extracts region proposals using selective search and then classifies each proposal using a CNN.
   - Fast R-CNN: Building on R-CNN, Fast R-CNN combines region proposal generation and object classification into a single network, improving speed and efficiency.
   - Faster R-CNN: Further enhancing Fast R-CNN, Faster R-CNN introduces a Region Proposal Network (RPN) that shares convolutional features with the object detection network, making the entire process end-to-end trainable.
2. Single Shot MultiBox Detector (SSD):
   SSD is a single-shot object detection model that predicts object bounding boxes and class probabilities directly from feature maps at multiple scales.
   It achieves real-time performance by utilizing a set of default anchor boxes with different aspect ratios and scales.
3. You Only Look Once (YOLO):
   YOLO is an object detection model that performs detection in a single pass through the network.
   It divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell.
   YOLO achieves real-time performance by considering global context and leveraging shared features across the image.

#### 2.7.1.1 YOLO (You Only Look Once)

Unlike traditional object detection algorithms that require multiple passes over an image, YOLO performs detection in a single pass. It uses a single neural network to simultaneously predict bounding boxes and class probabilities for multiple objects in an image.

Yolo architecture:

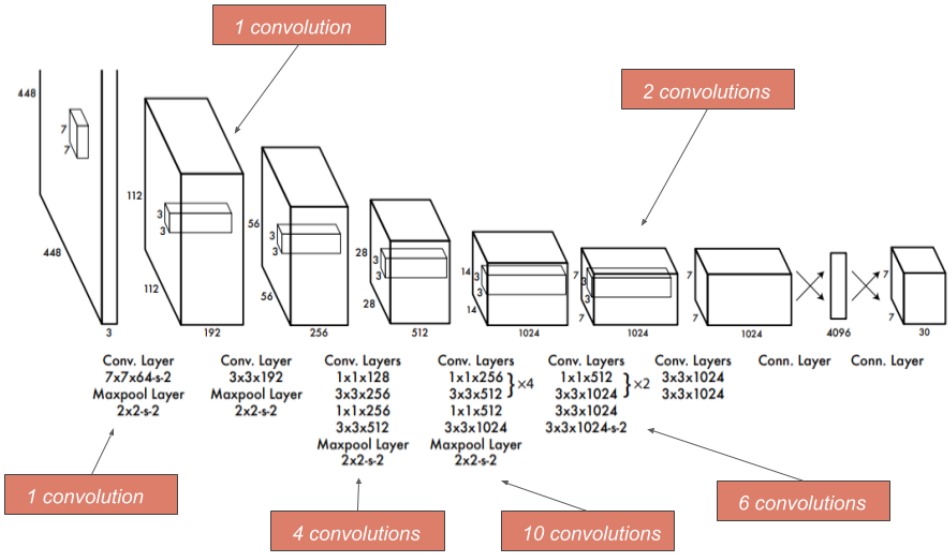


Figure 2.15: YOLO architecture
[5]

Some of the reasons why YOLO is leading the competition include its:

1. **Speed**:
   YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition, YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems, which makes it a great candidate for real-time processing.

2. **High detection accuracy** :
   YOLO is far beyond other state-of-the-art models in accuracy with very few background errors.

3. **Better generalization**:
   This is especially true for the new versions of YOLO, which will be discussed later in the article. With those advancements, YOLO pushed a little further by providing a better generalization for new domains, which makes it great for applications relying on fast and robust object detection.

4. **Open source** :
   Making YOLO open-source led the community to constantly improve the model. This is one of the reasons why YOLO has made so many improvements in such a limited time.

YOLO has evolved through different versions, each with improvements in accuracy and speed:

1. **YOLOv1 (You Only Look Once v1)**:
   - Introduced in 2015.
   - Pioneering version of YOLO that introduced the concept of single-stage object detection.
   - Divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell.
   - Achieved real-time object detection but suffered from localization errors and struggles with small objects.
2. **YOLOv2 (You Only Look Once v2) / YOLO9000**:
   - Released in 2016.
   - Improved the performance and accuracy of YOLOv1.
   - Incorporated anchor boxes to handle objects of different scales.
   - Introduced Darknet-19 architecture with additional convolutional layers.
   - Implemented multi-scale training and introduced the concept of "passthrough" layers.
   - Trained on both the COCO dataset and ImageNet, resulting in YOLO9000's ability to detect over 9,000 object categories.
3. **YOLOv3 (You Only Look Once v3)**:
   - Released in 2018.
   - Significantly improved the performance and accuracy over YOLOv2.
   - Utilized a variant of Darknet called Darknet-53 with a deeper architecture (53 layers).
   - Introduced multiple detection scales, enabling detection at different resolutions.
   - Integrated feature pyramid network (FPN) and skip connections for improved feature extraction.
   - Introduced the concept of "bounding box priors" to handle different aspect ratios of objects.
   - Achieved state-of-the-art performance in real-time object detection.
4. **YOLOv4 (You Only Look Once v4)**:
   - Released in 2020.
   - Built upon the success of YOLOv3 and aimed to further improve object detection performance.
   - Introduced various advancements, including CSPDarknet53 backbone, PANet, CIOU loss, and Mish activation function.
   - Improved network architecture for better speed and accuracy trade-offs.

- Addressed some limitations of YOLOv3, such as handling small object detection and reducing false positives.

There are more versions of Yolo, versions 5,6,7 and 8

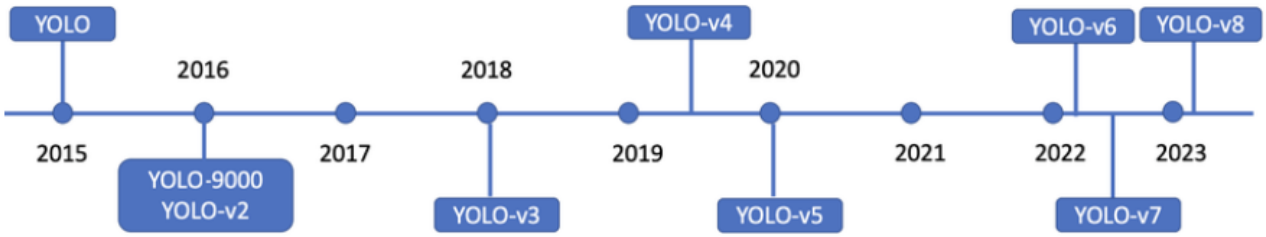


Figure 2.16: YOLO timeline[6]

## 2.8 Text Extraction

Text extraction [17]refers to the process of extracting text or information from various sources, such as images, scanned documents, PDFs, web pages, or other unstructured data formats. The goal of text extraction is to convert the text within these sources into a structured and machine-readable format for further analysis, processing, or storage.
Text extraction techniques involve utilizing a combination of language processing (NLP)[1], machine learning, and data extraction methods. Here are some key aspects and techniques related to text extraction:

1. Optical Character Recognition (OCR):
   OCR is a fundamental technique in text extraction that involves recognizing and converting text from images or scanned documents into machine-readable text. OCR algorithms employ image processing, pattern recognition, and machine learning techniques to identify and interpret characters and words within an image or document.

2. Information Extraction:
   Information extraction techniques focus on extracting specific types of structured information from unstructured text. This can involve identifying entities such as names, addresses, dates, or extracting structured data like tables, forms, or key-value pairs from documents.

3. Named Entity Recognition (NER):
   NER is a technique in information extraction that aims to identify and classify named entities, such as person names, organization names, locations, dates, or other specific entities within text. NER models utilize machine learning algorithms to recognize and categorize named entities based on contextual clues and patterns.

[1]NLP is a field of artificial intelligence and computational linguistics that focuses on the interaction between computers and human language. It involves the development of algorithms, models, and techniques to enable computers to understand, analyze, generate, and manipulate natural language data.

4. Text Parsing and Tokenization:
   Text parsing involves breaking down text into smaller units, such as sentences or words, for further analysis. Tokenization is the process of dividing text into individual tokens, which could be words, phrases, or even characters. Text parsing and tokenization are essential steps in many text extraction tasks and facilitate subsequent text processing and analysis.

5. Regular Expressions:
   Regular expressions are powerful tools used for pattern matching and extraction in text. They provide a flexible and efficient way to search for specific patterns or structures within a text and extract relevant information based on predefined rules.

### 2.8.1 Image to Text Pre-trained Models(OCR):

Text extraction models, also known as **text recognition** models or **Optical Character Recognition (OCR)** models, are designed to automatically extract text from images or scanned documents. These models aim to convert the text content within an image or document into a machine-readable format, enabling further analysis, indexing, or utilization of the extracted text.
Text extraction models typically involve a combination of computer vision (CNNs) and natural language processing (NLP) techniques such as recurrent neural networks (RNNs), to perform text recognition. They consist of two main steps:

1. **Text Detection:**
   Text detection focuses on identifying the regions of an image or document that contain text. This step involves using various computer vision techniques such as edge detection, contour analysis, and connected component analysis to locate and segment text regions. Some text detection models utilize object detection frameworks, like those mentioned in the object detection section, to specifically detect text regions within an image.

2. **Text Recognition:**
   Text recognition is the process of converting the detected text regions into machine-readable text. This step involves applying optical character recognition techniques to recognize and transcribe the individual characters or words within the text regions. Text recognition models often leverage deep learning architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to handle the complexity of character and word recognition.

Some popular text extraction models and techniques include:

- **Tesseract:**
  Tesseract is an open-source OCR engine developed by Google. It utilizes deep learning models and traditional OCR techniques to extract text from images and documents.
- **EAST (Efficient and Accurate Scene Text Detector):**
  EAST is an object detection model specifically designed for detecting text in natural scene images. It provides accurate and efficient text detection results.
- **CRNN (Convolutional Recurrent Neural Network):**
  CRNN is a deep learning architecture that combines CNNs and RNNs to jointly perform text detection and recognition. It is effective for extracting text from various sources, including images, documents, and videos.
- **Transformer-based models:**
  With the success of Transformer models in natural language processing tasks, they have also been applied to text extraction. These models, such as T2T and LayoutLM, leverage self-attention mechanisms to capture text patterns and extract information from document images.

## 2.9 Related Work

In the realm of text extraction and object detection, several notable research projects have utilized YOLO (You Only Look Once) and OCR (Optical Character Recognition) techniques to tackle similar challenges.

During our research, we discovered an approach to address our project's objectives, it's an OCR system that combines YOLO and Tesseract. The system[8] was specifically designed to extract and convert the contents of Lab Reports into editable files.
The implementation involved training YOLO on a personalized dataset to detect the specific objects of interest within the Lab Reports. Once the objects were detected, their coordinates were used to crop and store them in a separate list.
Finally, Tesseract OCR was utilized to process the cropped objects and obtain the desired output, effectively converting the Lab Reports into editable text.

For instance, in a recent final-year research project, a team employed YOLO for detecting specific regions of labels on medication boxes and subsequently utilizing OCR to extract relevant text information.

These works showcase the effectiveness of combining YOLO's object detection capabilities with OCR technology to achieve accurate and automated text extraction from various sources.

## 2.10 Summary and Conclusion

In summary, this chapter provided a comprehensive background on the tools and technologies used in our project. We explored various object detection tools, text extraction tools, and transfer learning techniques. The chapter also discussed the relevance of these tools to our solution and their applications in related works.
By gaining a solid understanding of the available tools and their capabilities we are able to make informed decisions in designing and implementing our system.

# Chapter 3

# In-Depth Exploration of Our Solution

## 3.1 Introduction

In this chapter, we delve into a comprehensive exploration of our solution, providing an in-depth analysis of our work strategy and approach. We present the system architecture, describing the various components and their interactions.
Furthermore, we detail the dataset collection and preparation process, outlining the considerations and steps involved. We delve into the implementation of object detection using YOLO, discussing the training process and customization performed.
Additionally, we explore the integration of OCR techniques for text extraction, highlighting the chosen OCR engine and any adaptations made.
Finally, we present the system implementation, validation procedures, and the obtained results. Through this examination, we aim to provide a comprehensive understanding of our solution and its effectiveness in addressing the identified problem.

## 3.2 System Architecture:

Our system architecture is designed to facilitate the automatic extraction of text from pharmaceutical labels with minimal manual intervention. It comprises several interconnected components that work cohesively to achieve accurate and efficient results. The overall flow of the system can be summarized as follows:

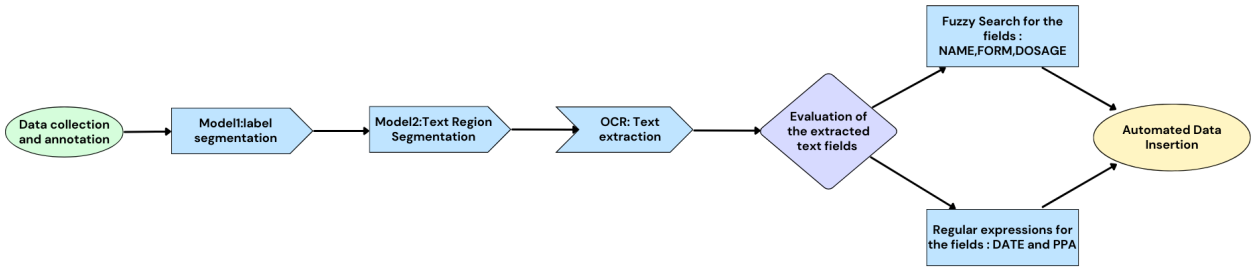


Figure 3.1: Label segmentation and text extraction system pipeline

1. **Data Collection and Annotation** :
   For the development of our two initial models, we conducted thorough data collection and annotation. we collected a set of images featuring medication boxes and then we annotated them based on regions(for image segmentation ) and classes(for classification as whiteVignette, redvignette, and greenvignette).
   Additionally, we gathered a separate dataset consisting of pharmaceutical labels and performed the annotation for specific fields such as NAME, PPA, FORM, DOSAGE, and DATE. Each of these fields was treated as a distinct class, and their respective regions within the labels were accurately annotated to perform the segmentation task.

2. **Label Segmentation Model (Model 1)** :
   The first component of our architecture is a label segmentation model. This model is responsible for identifying and isolating the regions of interest (the labels) within the input images. It utilizes YOLO V8 pre-trained model to accurately detect and segment the labels from the background. In addition, it classifies each label into its corresponding class, namely white vignette, red vignette, and green vignette.

3. **Text Region Segmentation Model (Model 2):**
   The output of the label segmentation model is then fed into a text region segmentation model. This model further refines the segmentation by identifying and extracting specific text regions (NAME, FORM, DOSAGE, DATE, and PPA )within the labeled area.
   Its purpose is to precisely isolate each textual field, excluding any unnecessary elements, and classify them with the corresponding class names (NAME, FORM, DOSAGE, DATE, and PPA).

4. **OCR for Text Extraction:**
   Once the text regions are identified, OCR techniques come into play. We used PP-OCR (Paddle OCR) to extract the actual text from the segmented regions, converting the visual characters into machine-readable text.

5. **Fuzzy Search and Regular Expression:**
   The extracted text is then subjected to additional processing steps for enhanced accuracy and verification.
   For certain fields, such as the Name, Form, and Dosage of the medication, a fuzzy search algorithm [18] is applied. By concatenating these fields and performing fuzzy matching with a text file that contains all medications.
   Regular expressions are used to extract specific patterns from the DATE and PPA fields ensuring their correct identification.

6. **Automated Field Population:**
   Finally, the extracted and verified text is utilized to automatically populate the corresponding fields in the application. The application's interface is designed to seamlessly integrate with the text extraction system, facilitating the retrieval and insertion of accurate medication information.

## 3.3 Label Segmentation with YOLOV8

YOLOv8 is a real-time object detection model developed by Ultralytics. It is the 8th version of YOLO and is an improvement over the previous versions in terms of speed, accuracy, and efficiency.

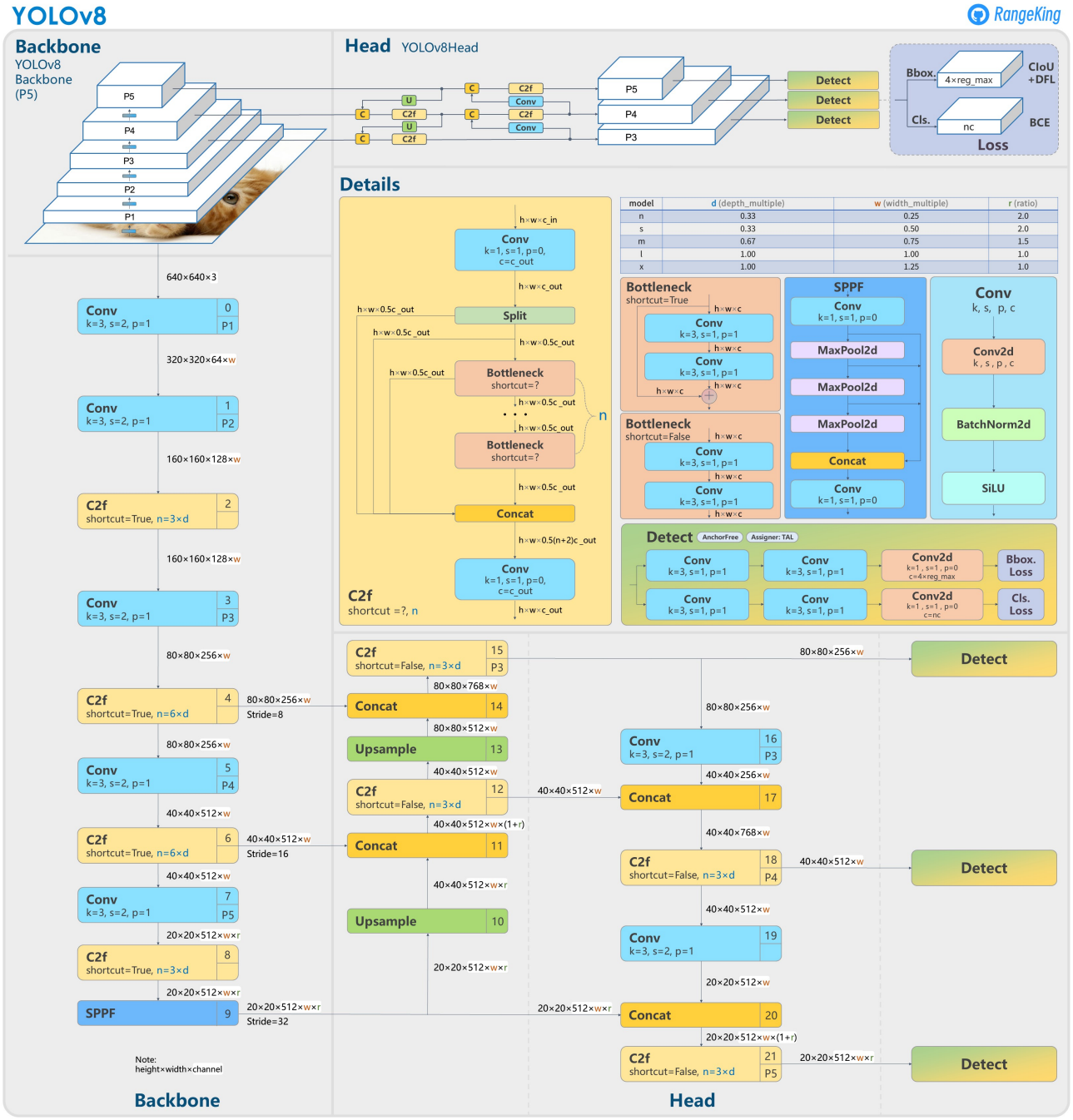


Figure 3.2: YOLOV8 architecture [9]

### 3.3.1 Comparative Advantages of YOLOv8: Unraveling the Advantages over Other Versions

The model is created in PyTorch and can run on both CPU and GPU. YOLOv8 is highly efficient and supports numerous formats such as TF.js or coreML. Like YOLOv7, YOLOv8 can be used for object detection, segmentation, and image classification.

The Mean Average Precision (mAP), of 53.9 has been marked with YOLOv8, the highest ever in YOLO history.

YOLOv8 has a new backbone network, a new anchor-free detection head, and a new loss function, making it a perfect choice for a wide range of object detection and image segmentation

tasks.

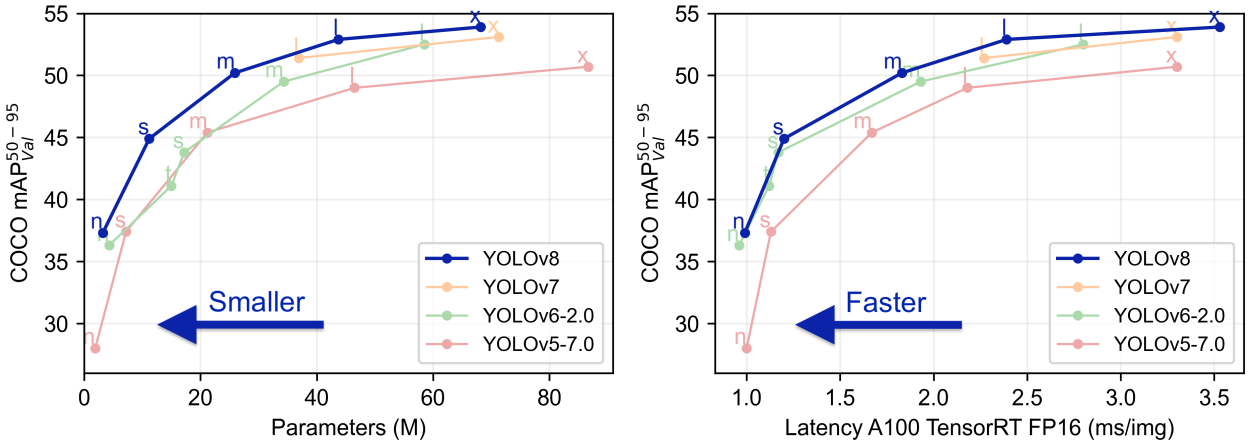


Figure 3.3: Comparison Between YOLO Versions [19]

### 3.3.2 Models for Detection, Segmentation, and Classification in YOLOv8

YOLOv8 comes bundled with the following pre-trained models:

- Object Detection checkpoints trained on the COCO detection dataset with an image resolution of 640.
- Instance segmentation checkpoints trained on the COCO segmentation dataset with an image resolution of 640.
- Image classification models pre-trained on the ImageNet dataset with an image resolution of 224.

In YOLOv8, there are five different models available for each category of detection, segmentation, and classification. YOLOv8 Nano is the smallest and fastest model, while YOLOv8 Extra Large (YOLOv8x) is the slowest yet most accurate model among them.

| Model | size (pixels) | mAP$^{val}$ 50-95 | Speed CPU ONNX (ms) | Speed A100 TensorRT (ms) | params (M) | FLOPs (B) |
|---|---|---|---|---|---|---|
| YOLOv8n | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 128.4 | 1.20 | 11.2 | 28.6 |
| YOLOv8m | 640 | 50.2 | 234.7 | 1.83 | 25.9 | 78.9 |
| YOLOv8l | 640 | 52.9 | 375.2 | 2.39 | 43.7 | 165.2 |
| YOLOv8x | 640 | 53.9 | 479.1 | 3.53 | 68.2 | 257.8 |

- **mAP$^{val}$** values are for single-model single-scale on COCO val2017 dataset.
  Reproduce by `yolo val detect data=coco.yaml device=0`
- **Speed** averaged over COCO val images using an Amazon EC2 P4d instance.
  Reproduce by `yolo val detect data=coco128.yaml batch=1 device=0/cpu`

Figure 3.4: YOLOv8's five different models

## 3.4 Text Extraction with PP-OCR (Paddle OCR)

PaddleOCR is an open source Optical Character Recognition (OCR) toolkit developed by PaddlePaddle, a deep learning platform. It provides a comprehensive set of tools and pre-trained models for text detection, recognition, and layout analysis in various languages.
In general, PaddleOCR has three parts: text detection, detected box rectification, and text recognition.
The following figure shows the overall architecture of PP-OCR.

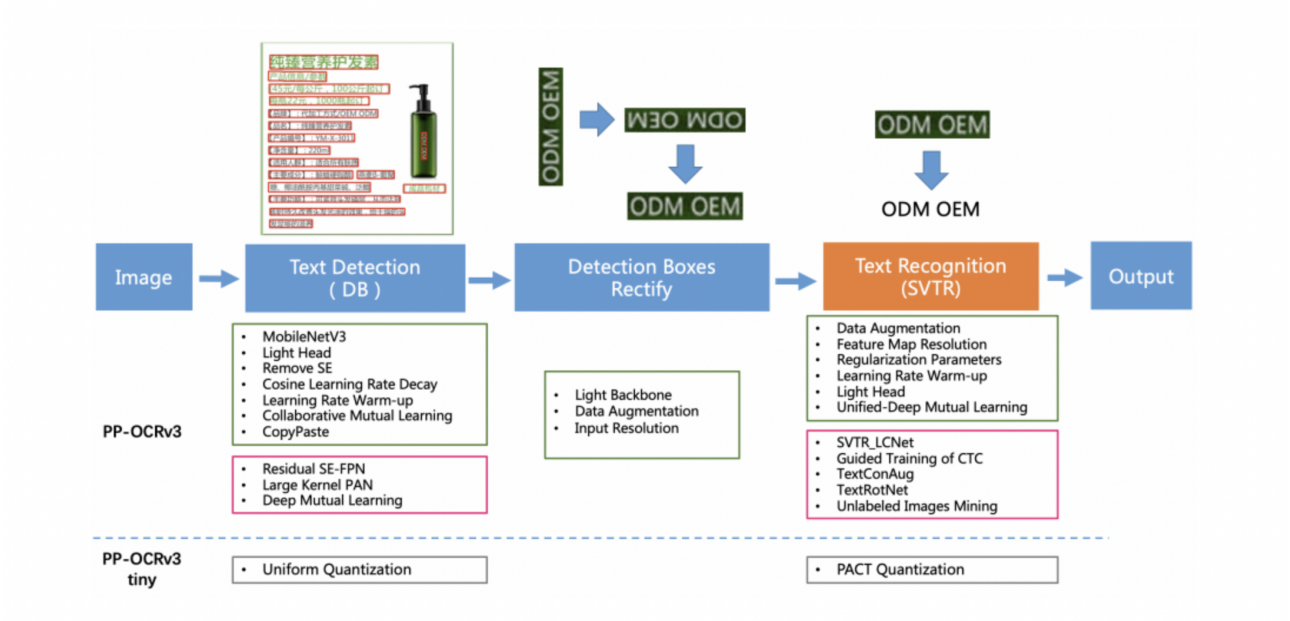


Figure 3.5: Architecture of PP-OCR [14]

## 3.5 Challenge: Lack of Consistent Pattern for Field Extraction

During the development of our system, we encountered a significant challenge due to the absence of a consistent pattern within each label of the dataset. This posed difficulties in extracting desired fields such as name, PPA, date, and form. The lack of a pattern hindered algorithm design and necessitated additional preprocessing steps.

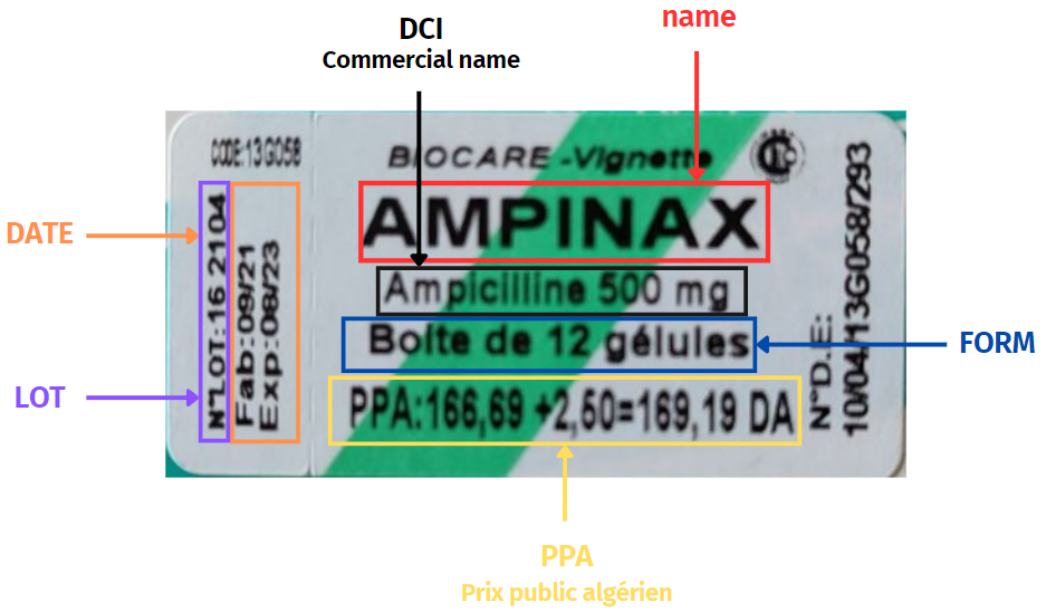


Figure 3.6: example of the fields in label 1

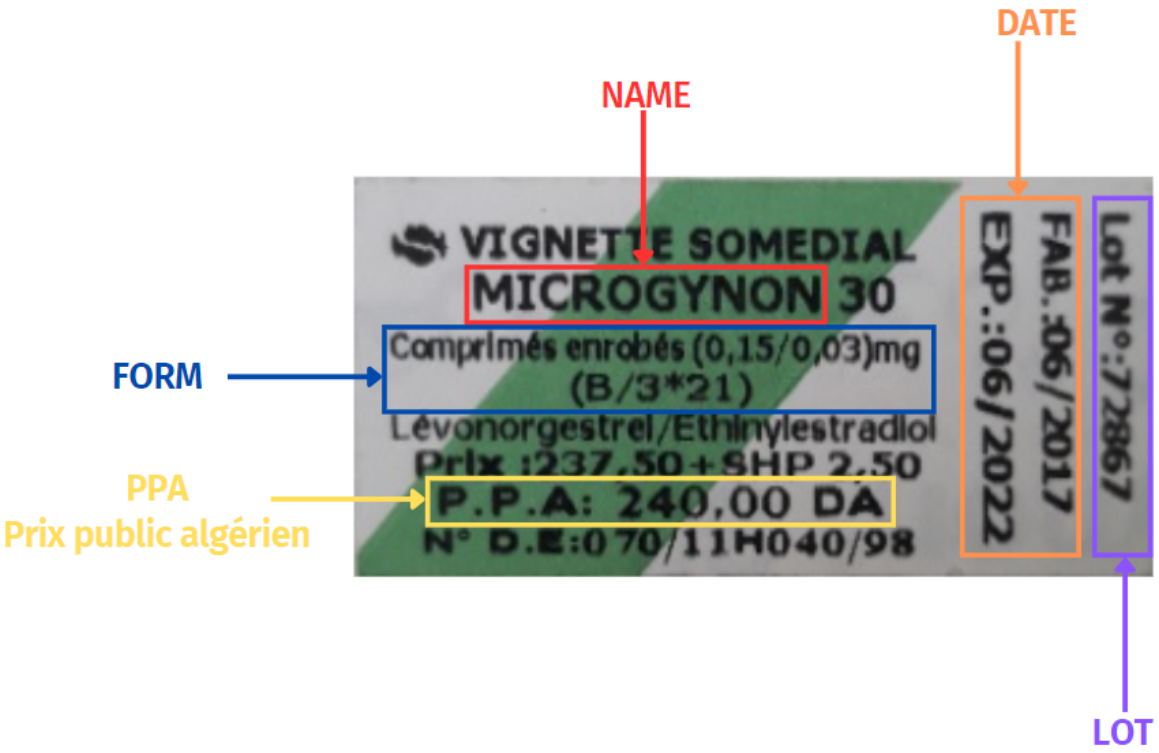


Figure 3.7: example of the fields in label 2

To address this problem, we introduced a model in the third step of our system "Text Region Segmentation Model" that focuses on extracting each field individually.

## 3.6 Dataset Collection and Annotation:

### 3.6.1 Label Segmentation Model (Model 1)

For the first model, we utilized a dataset consisting of 686 medication box pictures. Among these, 360 pictures had green labels, 175 pictures had red labels, and 151 pictures had white labels. The dataset was divided into training and validation sets, with a split of %75 for training and %25 for validation. Each label category was further divided into training and validation sets, maintaining the balance across the categories. Consequently, the training set consisted of 518 images, while the validation set contained 168 images. This division allowed us to effectively train and evaluate the model's performance on different labeled medication box images.

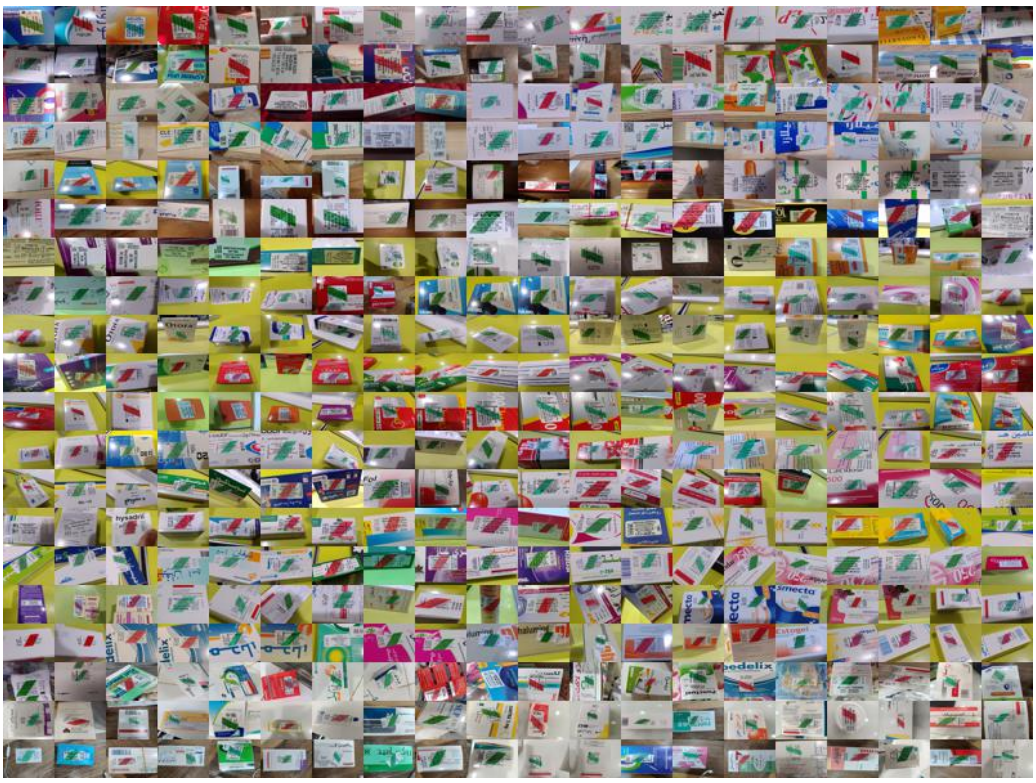

Figure 3.8: Our dataset

For the annotation task, we annotated the region of each label with its respective class (RedVignette, WightVignette, or GreenVignette), for that we used LabelMe which is an open-source image annotation tool used for creating labeled datasets by manually annotating objects or regions of interest in images with bounding boxes, polygons, or pixel-level masks.
Here is how we did the annotation of our first dataset :


Figure 3.9: Annotation of the first training dataset images using labelme

### 3.6.2 Text Region Segmentation Model (Model 2)

For the second model, we utilized a dataset of 442 segmented label images. This dataset was divided into a training set with 353 images and a validation set with 89 images.
The annotation process was performed using LabelMe, for each label, we annotated the region of each field with its appropriate class name (NAME, PPA, FORM, DOSAGE, DATE) as depicted in the image below.

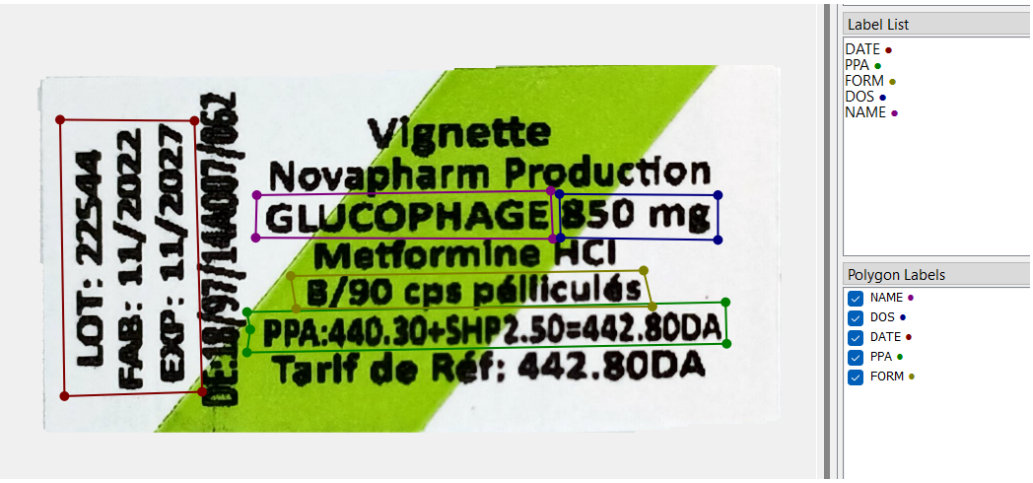


Figure 3.10: Annotation of the second training dataset images using labelme

## 3.7 System Implementation and Validation

### 3.7.1 Label Segmentation Model

For the label segmentation task, we employed the YOLOv8 model, specifically the medium-sized variant (YOLOv8m), which was pre-trained for instance segmentation. The model was trained for 100 epochs using our training set consisting of 518 images. After training, we obtained the following results:

```
Ultralytics YOLOv8.0.58 🚀 Python-3.9.16 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8m-seg summary (fused): 245 layers, 27224121 parameters, 0 gradients, 110.0 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 6/6 [00:58<00:00,  9.82s/it]
                   all        167        167      0.996      0.992      0.995      0.988      0.996      0.992      0.995      0.989
          GreenVignette       167         86       0.99          1      0.995      0.993       0.99          1      0.995      0.991
          WhiteVignette       167         37          1      0.976      0.995      0.987          1      0.976      0.995      0.985
            RedVignette       167         44      0.997          1      0.995      0.984      0.997          1      0.995       0.99
```

Figure 3.11: Classification report of the model1

We can see those metrics from these plots:

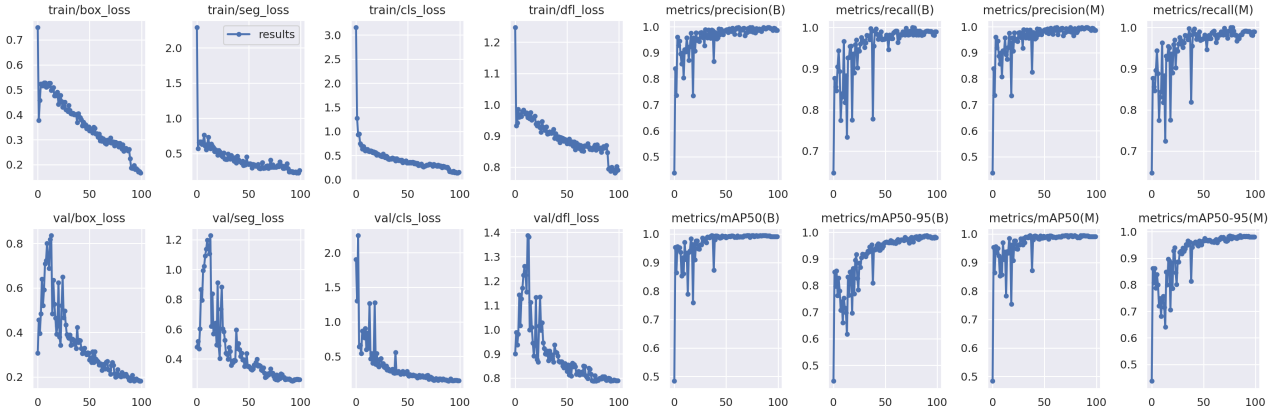


Figure 3.12: evaluation metrics plots

We tested our model on our validation set, and these are the results :


Figure 3.13: segmentation performed by model1 on one batch of the validation data set

After training our model and evaluating it, we tried it on a new image:

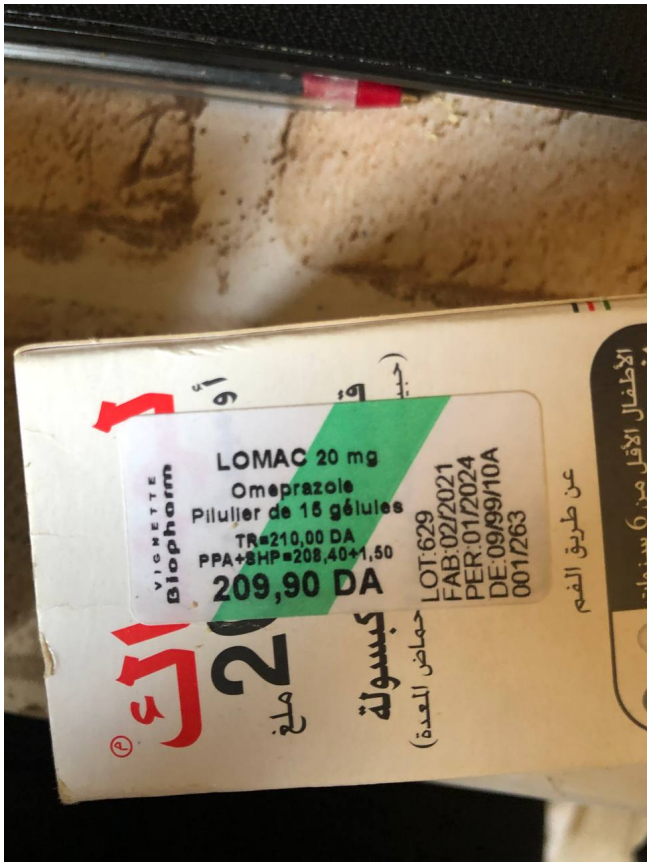


Figure 3.14: input image

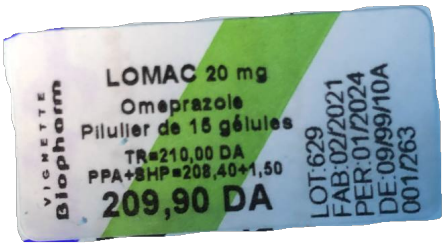


Figure 3.15: result image labeled as "GreenVignette"

### 3.7.2 Text Region Segmentation Model

For the Text Region Segmentation Model, we employed two pre-trained YOLOv8 models of the Extra Large size. Instead of using the pre-trained weights, we trained the models from scratch with randomly initialized weights on our second dataset for 100 epochs. The objective was to extract and segment the fields: Date, Name, Form, Dosage, and PPA.
The first trained model had better performance in extracting the Date, Name, and Dosage fields, while the second model proved more effective in extracting the PPA and Form fields.
To ensure optimal performance, we employed a logical function that combined the results of both models. For instance, we selected the Name, Date, and Dosage from the first model and the PPA and Form from the second model.
We obtained the following results:

```
Validating runs/segment/train4/weights/best.pt...
Ultralytics YOLOv8.0.105 🚀 Python-3.10.11 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8x-seg summary (fused): 295 layers, 71725471 parameters, 0 gradients
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 3/3 [00:07<00:00,  2.62s/it]
                   all         89        430      0.917      0.883       0.93      0.721      0.903      0.871      0.916      0.684
                  FORM         89         87      0.838      0.833        0.9      0.657      0.782      0.782      0.837      0.601
                   DOS         89         76      0.904      0.737      0.841      0.599      0.903      0.737      0.831      0.563
                   PPA         89         90      0.962      0.967      0.977      0.786       0.95      0.956      0.969      0.738
                  DATE         89         87      0.974      0.977      0.974      0.781      0.973      0.977      0.979      0.759
                  NAME         89         90      0.909        0.9      0.959      0.781       0.91      0.902      0.964      0.759
Speed: 3.1ms preprocess, 23.5ms inference, 0.0ms loss, 4.3ms postprocess per image
Results saved to runs/segment/train4
```

```
Ultralytics YOLOv8.0.106 🚀 Python-3.10.11 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8x-seg summary (fused): 295 layers, 71725471 parameters, 0 gradients
val: Scanning /content/MyChampsReadyToRun++/test/labels.cache... 89 images, 0 backgrounds, 0 corrupt: 100% 89/89 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 6/6 [00:14<00:00,  2.37s/it]
                   all         89        430      0.926      0.878      0.929       0.72      0.904      0.873      0.914       0.68
                  FORM         89         87      0.887      0.816       0.86      0.614      0.843      0.803      0.818      0.571
                   DOS         89         76      0.839      0.711      0.853      0.603       0.82      0.717      0.843      0.568
                   PPA         89         90      0.966      0.942      0.972      0.796      0.955      0.936      0.968       0.72
                  DATE         89         87      0.967          1      0.977      0.808       0.96          1      0.977      0.777
                  NAME         89         90      0.969      0.922      0.981      0.782      0.945      0.911      0.964      0.762
Speed: 5.0ms preprocess, 51.7ms inference, 0.0ms loss, 5.9ms postprocess per image
Results saved to runs/segment/val
```

Figure 3.16: Classification reports of the two models

We visualize the metrics from these plots:

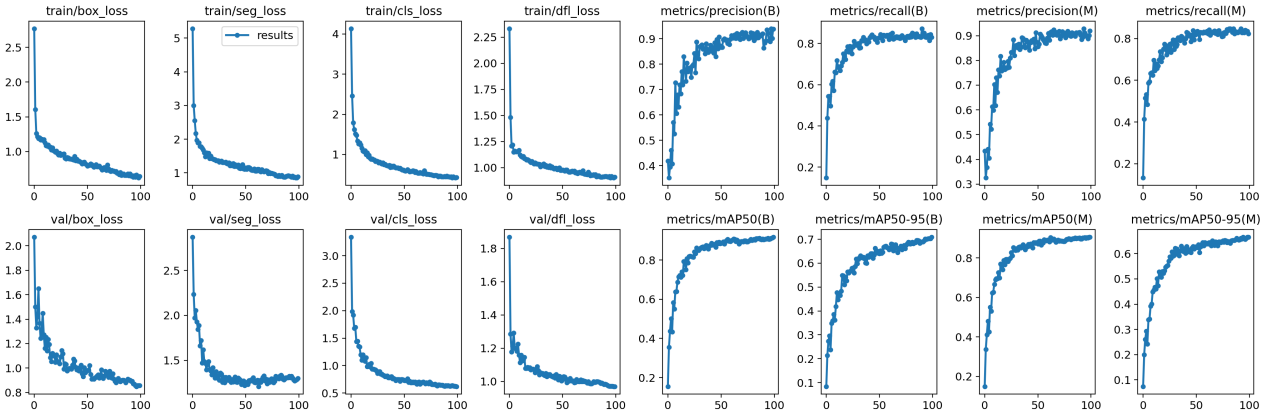


Figure 3.17: evaluation metrics plots of the first model

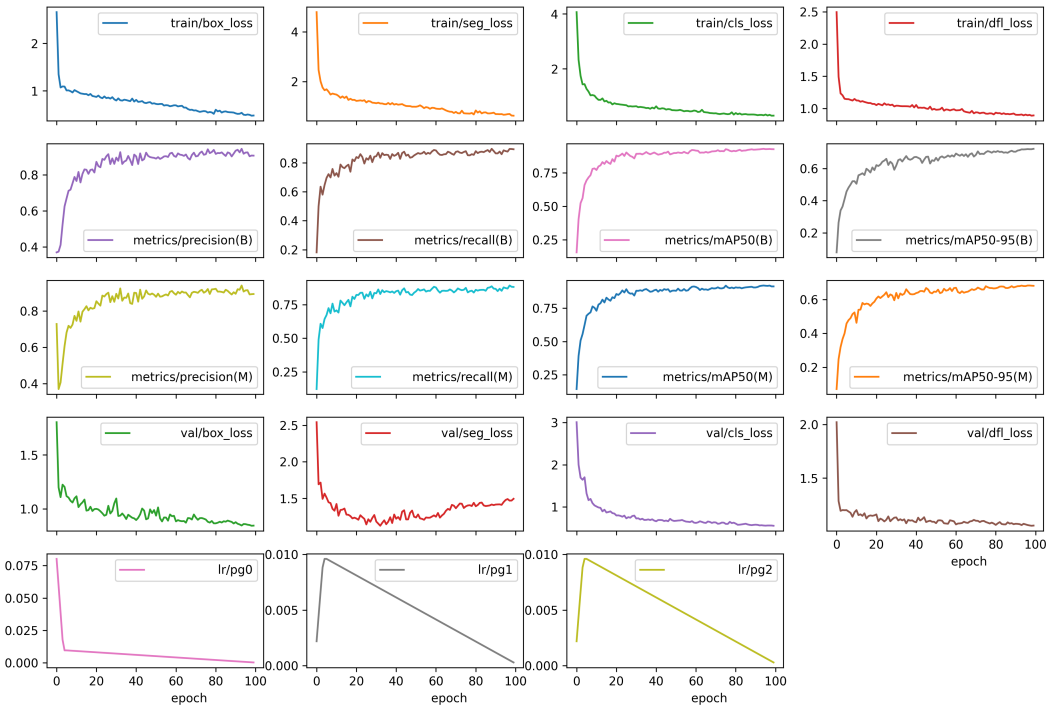


Figure 3.18: Evaluation metrics plots of the second model

We tested our model on our validation set, and these are the results :


Figure 3.19: segmentation performed by the first model on one batch of the validation data set


Figure 3.20: segmentation performed by the second model on one batch of the validation data set

After training both the first and second models, we combined their outputs using a logical function to improve performance, the output of this function is the output of what we named the "Text Region Segmentation Model". Here are some results of the model's predictions on a new image(we used the output of the first model(the label segmentation model):

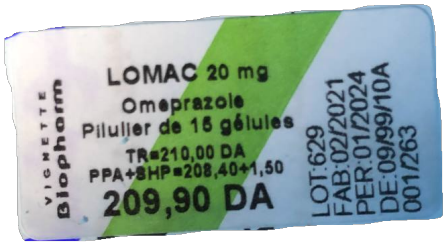


Figure 3.21: output of the label segmentation model is fed the text region segmentation model


Figure 3.22: Output NAME field


Figure 3.23: Output FORM field

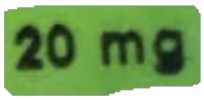


Figure 3.24: Output DOSAGE field

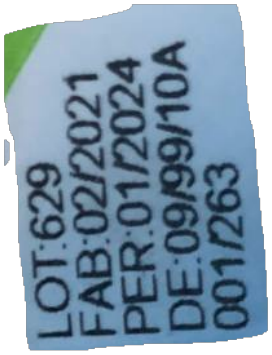


Figure 3.25: Output DATE field


Figure 3.26: Output PPA field

### 3.7.3 Text Extraction

For the specific task at hand, we utilized PaddleOCR to extract the text from each segmented field obtained in the previous step. Here are the results for the same example: we provided the five segmented text region images, and PaddleOCR generated the following text:

```
Name  : LOMAC
Dossage  : 20mg
Forme  : Piluller de16g6lules
DDP : 01/2024
DDF : 02/2021
lot : 629
ppa_value : 209.9
```

Figure 3.27: output of the PADDLE OCR

### 3.7.4 Verification Step: Fuzzy Search and Regular Expression

As a verification step, we utilize two data files to validate the OCR output and ensure accuracy. One file contains information about medications with green labels, while the other file contains information about medications with red labels. Both files encompass all the medication data available in the Algerian market for the corresponding label types.

1. **Fuzzy search for the fields: NAME, FORM, DOSAGE**
   After identifying the label type from the label segmentation model (red vignette, white vignette, green vignette), we select the corresponding data file based on the label type. We then perform a fuzzy search on the name, form, and dosage fields in this file to mitigate any errors that may have occurred in the previous steps of our system. This allows us to retrieve the remaining medication information accurately.

2. **Regular Expression for the fields: DATE, PPA**
   For the data and PPA fields, we perform additional processing to extract the required

values. Specifically, we take the OCR output and apply regular expressions to isolate the relevant information.
For instance, when extracting the date, we extract the values of DDP, DDF, and LOT separately to differentiate between the fields. Similarly, for PPA, we extract only the price without any unnecessary additions.

## 3.8 Results and Discussion:

In this section, we will present a sample medication that has been processed through our system and discuss the performance of the outputs of each step.

1. **The medication box picture is scanned by our application.**


Figure 3.28: Mediation box picture

2. **The result of the label segmentation model**

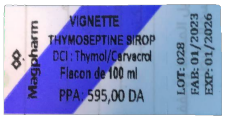


Figure 3.29: segmented label

3. **The result of the Text region segmentation model** As we analyze the output of this step, we can observe that the model is not extracting the DOSAGE field, which is expected since it is not explicitly written on the medication label. However, our system


Figure 3.30: Output NAME field


Figure 3.31: Output FORM field

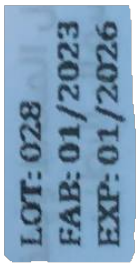


Figure 3.32: Output DATE field

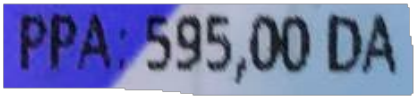


Figure 3.33: Output PPA field

effectively addresses this issue by leveraging fuzzy search and extracting the accurate information, including the missing fields, from the appropriate file (specifically, the Red labeled medication file in this case).

4. **Application of the OCR**

```
Name  : THYMOSEPTINE
Forme  : Flacon de100ml
DDF : 01/2023
DDP : 01/2026
lot : 028
ppa_value : 595.0
```

Figure 3.34: OCR output

5. **Observations:**
   The observed output of the OCR aligns with our expectations (correct text extraction).
   To further enhance the accuracy and reliability of the extracted information, we incorporate additional verification steps such as fuzzy search and regular expressions to extract only the necessary information and feed it into our system.
   By comparing these observations with the metrics calculated during the validation phase of our model, we can confidently state that our system performs well.

## 3.9 Summary and Conclusion

To conclude, this chapter delved into the detailed exploration of our solution and the steps involved in implementing our system architecture. We presented an overview of the system pipeline, highlighting the key components such as label segmentation, text region segmentation, OCR, and data verification.

The chapter also demonstrated the effectiveness of our approach through the evaluation of individual models and their results.

# Chapter 4

# Desktop Application

## 4.1 Application Overview

The desktop application is a user-friendly interface designed to support pharmacists in managing medication sales, product addition, and stock management. It features a visually appealing UI and provides support for the French language, catering to the needs of pharmacists who predominantly use it.

The application incorporates three main functionalities: sale, add product and stock management. These functionalities aim to streamline the medication management process and ensure accurate record-keeping.

### 4.1.1 Logo Display

The logo of our desktop application, SpeedPharm, features a pharmaceutical label (vignette), representing our target domain.We have creatively integrated an IT element into the logo, symbolizing our utilization of artificial intelligence (AI) technology. This signifies that our application harnesses the power of advanced computing capabilities to deliver innovative solutions and enhance the workflow in the pharmaceutical industry.

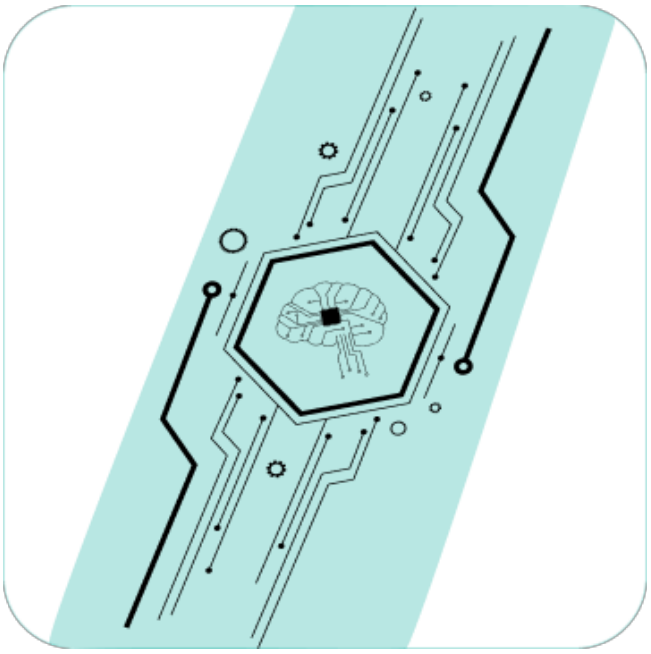

Figure 4.1: SpeedPharm Logo

### 4.1.2 User Interface Design

The user interface of the desktop application is thoughtfully designed to enhance usability and efficiency. It follows modern design principles, incorporating intuitive navigation, clear labeling, and visually appealing elements. The UI's layout ensures easy access to the application's functionalities, allowing pharmacists to quickly perform tasks without confusion or frustration.

### 4.1.3 French Language Support

In order to cater to the language preferences of pharmacists, the desktop application, SpeedPharm, provides comprehensive support for the French language. The user interface elements, labels, and prompts within the application are all in French, ensuring a seamless and intuitive experience for French-speaking users.

The decision to prioritize French language support is rooted in the fact that a significant number of pharmacists in Algeria have studied in French. Additionally, medication vignettes, which contain crucial information about medications, are often available in French. Therefore, incorporating French language support allows us to effectively serve the needs of the majority of pharmacists who rely on the French language in their daily practice.

## 4.2 Application Architecture

The desktop application is built using a combination of Java and FastAPI. Java is used for client-side development, providing robustness and cross-platform compatibility. The JavaFX framework is employed to create an interactive GUI. On the server side, FastAPI is utilized for integration and API implementation. FastAPI enables efficient communication between the client and server, allowing seamless data exchange and responsiveness. It also enables the application to interact with external resources such as databases.
This combination of Java and FastAPI provides a robust and user-friendly client-side experience while enhancing server-side communication and integration with external resources.

## 4.3 Functionalities

The desktop application encompasses three principal functionalities: sale, add product and stock management. Each functionality serves a specific purpose in medication management and offers a streamlined workflow for pharmacists.

### 4.3.1 Sale Functionality

The sale functionality allows pharmacists to process medication sales efficiently. Pharmacists can scan the barcode or manually enter the medication details. The application retrieves the relevant information, such as medication name, dosage, price, and patient details, from the medication database. The calculated total amount is displayed, and the sale is recorded in the system, updating the stock accordingly.

#### 4.3.1.1 Screenshot: Sale Interface


Figure 4.2: Sale Interface

#### 4.3.1.2 Sequence Diagram: Sale Process

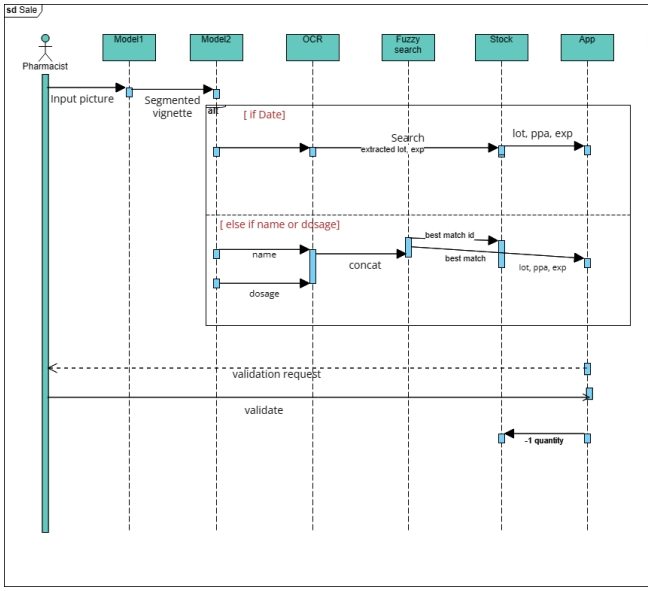


Figure 4.3: Sale Process

#### 4.3.1.3 Pipeline: Sale

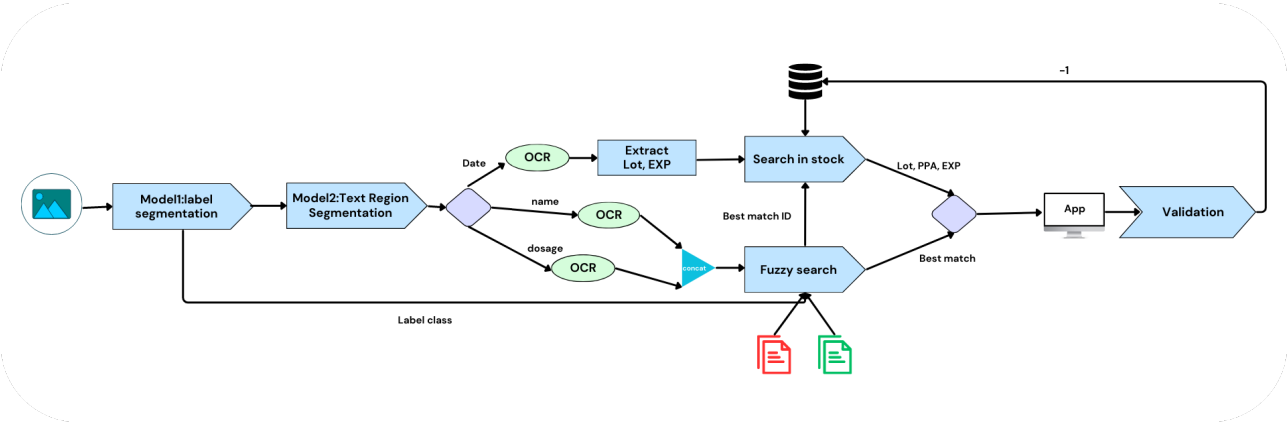


Figure 4.4: Sale pipeline

### 4.3.2 Add Product Functionality

The add-product functionality enables pharmacists to add new medications to the system. The application incorporates the label segmentation and text extraction system to extract medication information from images. The extracted medication information is checked against existing files, which are categorized as either green-labeled or red-labeled. If the medication is not found, it is added to the appropriate file based on its label type, facilitating future verification steps.

#### 4.3.2.1 Screenshot: Add Product Interface

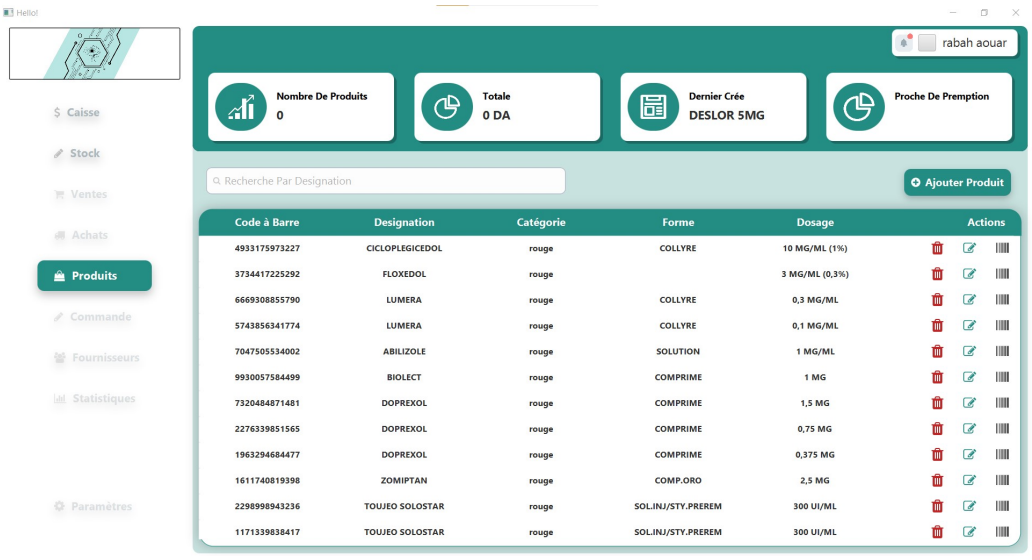


Figure 4.5: Add Product Interface

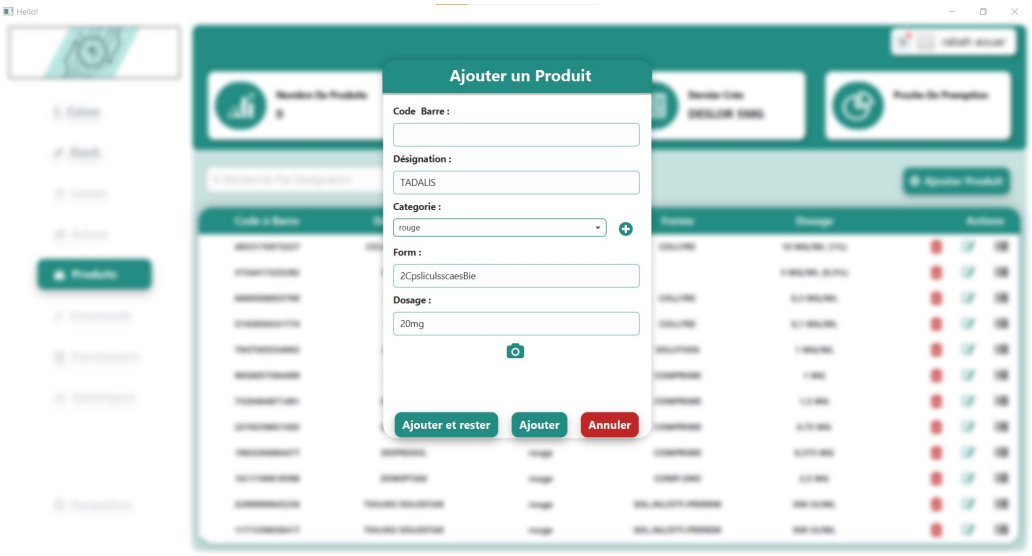


Figure 4.6: Add Product Interface

#### 4.3.2.2 Sequence Diagram: Add Product Process

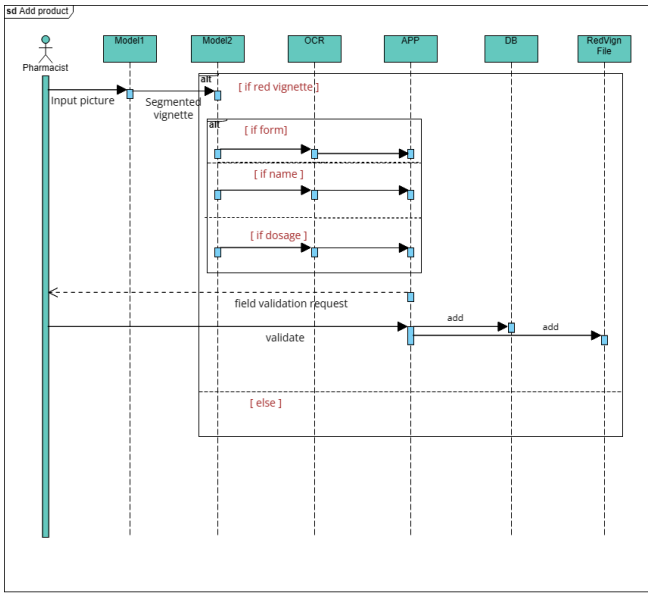


Figure 4.7: Add Product Process

#### 4.3.2.3 Pipeline: Add product

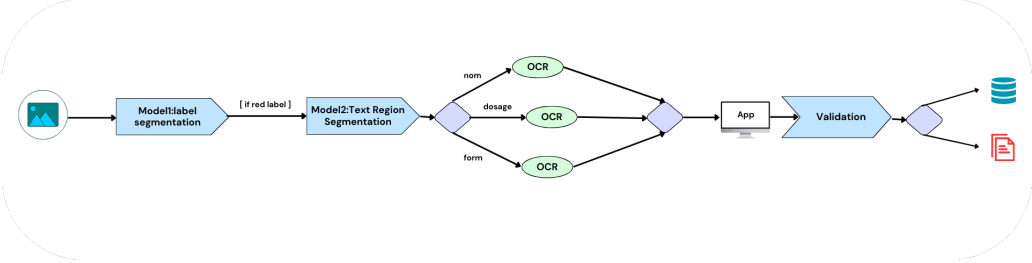


Figure 4.8: Add product pipeline

### 4.3.3 Stock Management Functionality

The stock management functionality allows pharmacists to effectively manage medication inventory. The system scans images of medication labels and applies fuzzy search and regular expression verification to extract pertinent information, including medication name, quantity, and supplier details. Users can review and edit the extracted values before they are automatically inserted into the application. This updated information is then added to the medication stock database, ensuring accurate and up-to-date inventory records.

#### 4.3.3.1 Screenshot: Stock Management Interface

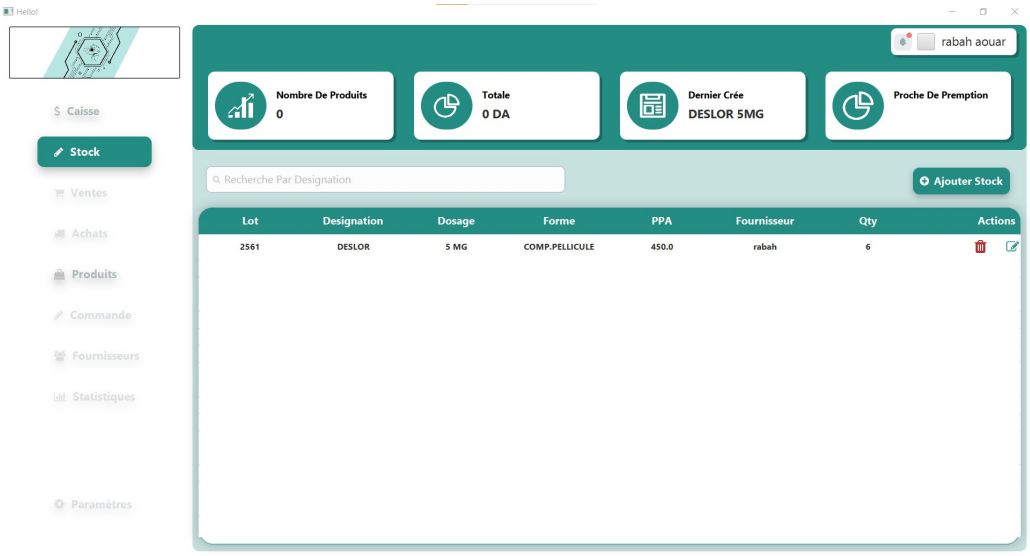


Figure 4.9: Stock Management Interface

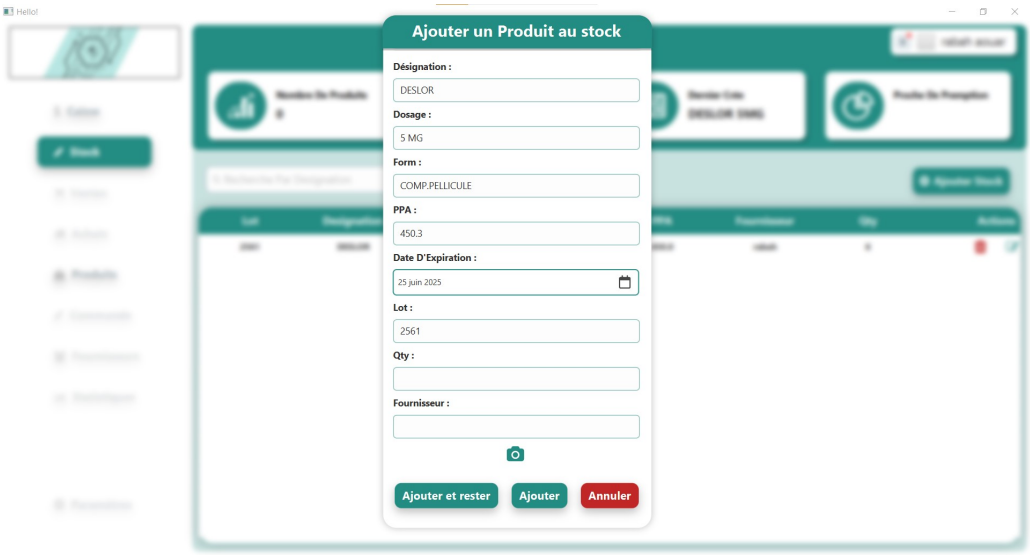


Figure 4.10: Stock Management Interface

#### 4.3.3.2 Sequence Diagram: Stock Management Process

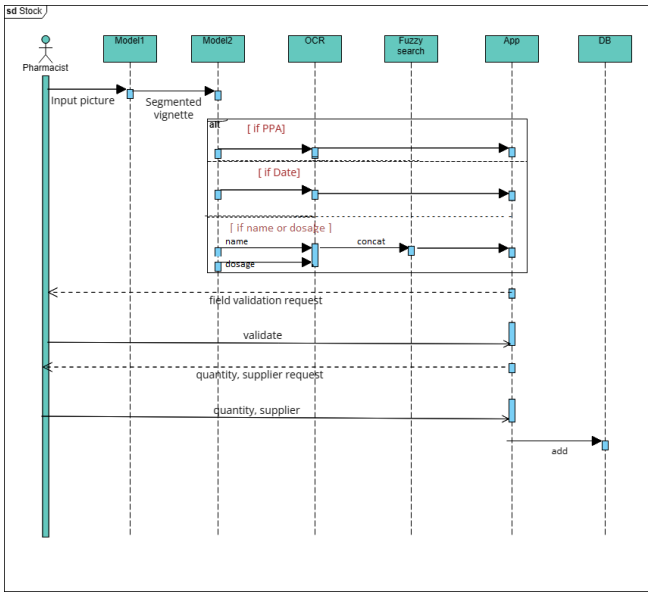


Figure 4.11: Stock Management Interface

#### 4.3.3.3 Pipeline: Stock

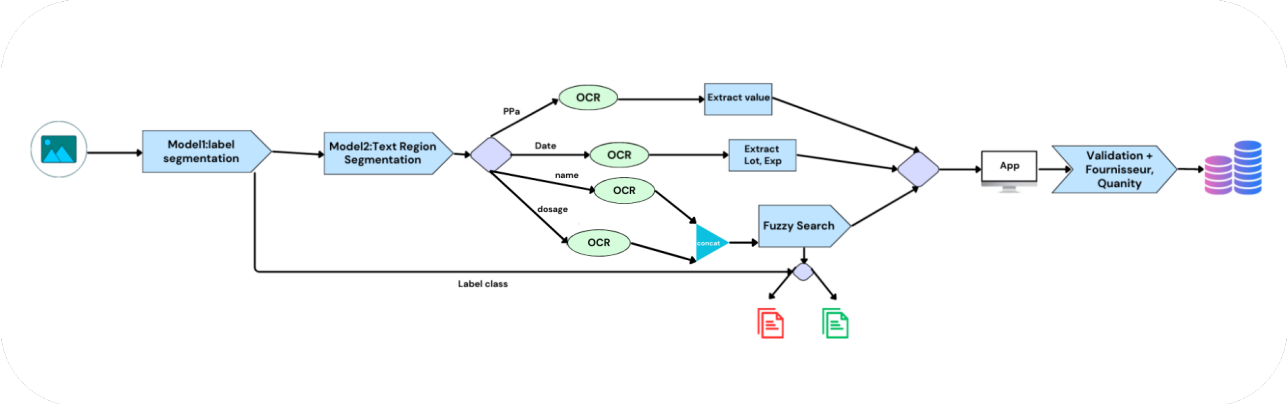


Figure 4.12: Stock pipeline

## 4.4 Summary and Conclusion

In summary, SpeedPharm is a powerful desktop application that enables pharmacists to streamline medication management tasks. With an intuitive interface and comprehensive features for sale, product addition, and stock management, SpeedPharm simplifies workflows and ensures accurate record-keeping. It provides robust language support for French-speaking pharmacists and leverages the integration of FastAPI for enhanced performance and seamless interaction. SpeedPharm is a valuable tool that empowers pharmacists to work efficiently and effectively in managing medications.

# Conclusion

The SpeedPharm project is a desktop application that revolutionizes medication management for pharmacists. By incorporating AI technologies such as OCR, AI models, and fuzzy search algorithms, SpeedPharm automates tasks, improves accuracy, and enhances efficiency. With its intuitive interface and AI-driven functionalities, SpeedPharm empowers pharmacists to streamline workflows, save time, and provide better care to their patients. It extracts information from medication labels, categorizes products, and ensures accurate stock management. SpeedPharm is a powerful tool that leverages AI to transform the management of medications and optimize pharmacy operations.

# Bibliography


[1] Medium arcticle. five popular cnn architectures clearly explained and visualized. https://towardsdatascience.com/5-most-well-known-cnn-architectures-visualized-af76f1f0065e.

[2] Medium arcticle. What is convolutional neural network cnn (deep learning). https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5.

[3] Medium article. Image filters in python. https://medium.com/@dreikoen/image-filters-in-python-6423103a08f6.

[4] Medium article. Understanding morphological image processing and its operations. https://towardsdatascience.com/understanding-morphological-image-processing-and-its-operations-7bcf1ed11756.

[5] blent. Yolo : détection sur les images avec tensorflow. https://blent.ai/blog/a/detection-images-yolo-tensorflow.

[6] Reaserch Gate. owstallnumbers: A small dataset for stall number detectionof cow teats images. https://www.researchgate.net/figure/Timeline-of-You-Only-Look-Once-YOLO-variants_fig1_369379818.

[7] Research Gate. Modeling course achievements of elementary education teacher candidates with artificial neural networks. https://www.researchgate.net/publication/326417061_Modeling_Course_Achievements_of_Elementary_Education_Teacher_Candidates_with_Artificial_Neural_Networks.

[8] Github. Custom ocr yolo. https://github.com/Borahb/Custom-OCR-YOLO.

[9] Github. yolov8 in depth. https://github.com/akashAD98/yolov8_in_depth.

[10] i2tutorials. Explain about gaussian filtering. https://www.i2tutorials.com/explain-about-gaussian-filtering/.

[11] i2tutorials. Explain in detail about median filtering. https://www.i2tutorials.com/explain-in-detail-about-median-filtering/.

[12] itzone. Deep learning network architectures in solving semantic segment (image semantic partitioning). https://itzone.com.vn/en/article/part-1-deep-learning-network-architectures-in-solving-semantic-segment-image-semant

[13] MDPI Open Access Journals. Incorporating a novel dual transfer learning approach for medical images. https://www.mdpi.com/1424-8220/23/2/570.

[14] Medium. Paddleocr: The latest lightweight ocr system. https://medium.com/axinc-ai/paddleocr-the-latest-lightweight-ocr-system-a13171d7ea3e.

[15] Microchip. Sobel filtering before and after. https://onlinedocs.microchip.com/pr/GUID-37AD5EEE-6FAB-48FC-89F6-CAA649534B2A-en-US-1/index.html.

[16] pinecone. Introduction to transfer learning. https://www.pinecone.io/learn/transfer-learning/.

[17] Serengeti. Deep learning : Ocr text detection and text recognition. https://serengetitech.com/tech/deep-learning-ocr-text-detection-and-text-recognition/.

[18] stackoverflow. how token sort ratio works?

[19] Github ultralytics. Ultralytics yolov8:the state-of-the-art yolo model. https://github.com/ultralytics/ultralytics.