People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

IBN KHALDOUN UNIVERSITY OF TIARET

Dissertation

Presented to:

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTEMENT OF COMPUTER SCIENCE

in order to obtain the degree of :

MASTER

Specialty: Artificial Intelligence and Digitalization

Presented by:
GAAMOUCI Mohamed Mounsif
OULD AHMED Yacine

On the theme:

Extracting meaningful information from a scanned document

Defended publicly on 01/06/2025 in Tiaret in front the jury composed of: :

Mr DJAAFRI Laouni	MCA	Tiaret University	Chairman
Mr GAFOUR Yacine	MCA	Tiaret University	Supervisor
Mrs HAMDANI Abdia	MAA	Tiaret University	Examiner

2024-2025

Abstract

The extraction of information from scanned documents is a crucial process in numerous fields such as administrative digitization, medical record management, business automation, and information retrieval. This project aims to develop a methodology to assist administrators in extracting relevant data from scanned documents. The proposed system combines techniques from image processing, object detection, and OCR (Optical Character Recognition) to detect and extract structured data such as names, dates, phone numbers, and addresses. This extracted information can be used to populate forms, generate reports, or serve as input for automated systems, ultimately making the processing of scanned documents more efficient and reliable.

الملخص

يُعد استخراج المعلومات من الوثائق الممسوحة ضوئيًا عملية أساسية في العديد من المجالات مثل الرقمنة الإدارية، إدارة السجلات الطبية، أتمتة الأعمال، واسترجاع المعلومات. يهدف هذا المشروع إلى تطوير منهجية تساعد المسؤولين في استخراج البيانات المهمة من الوثائق الممسوحة ضوئيًا. يعتمد النظام المقترح على تقنيات معالجة الصور، واكتشاف الكائنات، والتعرف البصري على الأحرف (OCR) لاكتشاف واستخراج بيانات منظمة مثل الأسماء، التواريخ، أرقام الهواتف والعناوين. يمكن استخدام هذه البيانات لإنشاء تقارير، ملء النماذج، أو تغذية أنظمة الأتمتة، مما يجعل معالجة الوثائق أكثر كفاءة وموثوقية.

Acknowledgements

At the end of this work, we would like to express our deepest gratitude and sincere appreciation to all those who, directly or indirectly, contributed to the completion of this thesis.

First and foremost, we extend our heartfelt thanks to our supervisor, **Mr. GAFOUR Yacine**, for his support, availability, continuous guidance, and valuable advice throughout this project. His encouragement and expertise were instrumental in bringing this work to fruition.

We would also like to express our sincere appreciation to all the staff of the Computer Science Department, in particular **Mr. MEGHAZI Hadj Madani** and **Mr. BOUGUESSA Abdelkader**, for their dedication and the knowledge they have shared with us throughout our academic journey.

Dedication

I dedicate this work to my beloved parents.
Thank you for your unconditional love, your endless support, and your constant belief in me.
Everything I am today is because of your guidance and sacrifices.

To my dear sisters your encouragement, love, and presence have meant the world to me.
Thank you for always being there.

With all my love and gratitude.
Gaamouci Mohamed Mounsif

Table of Content

Tables List

Figures List

1 General Introduction

1.1 Context and Motivation

Today, technology is advancing at a greater pace than ever before. Organizations across industries are going for the deployment of digital solutions for greater productivity, traceability, and ease of access. Digital transformation is the process of transforming paper records to electronic records that are easy to search, utilize, and maintain. It is not just a technological upgrade but a big change that helps businesses to move away from traditional processes to intelligent, automated systems.

Banks, hospitals, government offices, and insurance companies generate significant volumes of paperwork every day like customer forms, medical reports, legal documents, and transaction records. By digitizing these types of documents, these entities are able to speed up their processes, decide faster, and respond to their customers faster. Electronic documents are easy to store for long periods of time, are retrievable anywhere, and save paper, being eco-friendly.

One of the most important uses of digitization, in this case human resource, is resume processing. Thousands of resumes in different forms like PDFs, word document, scanned document, or image. All the resumes are in different layouts, designs, and languages, and reading and interpreting the information that is included like personal information, education, experience, and skills is not easy.

Manual resume sorting is time-consuming, where the best workers might not be spotted or incorrect decisions might be made. Resume processing becomes a reality with the aid of automation, thus enabling companies to select the best workers at any given time, and even make more well-based hiring decisions. However, there's no point in simply reading a resume as an image. It must be processed so that the words and data contained within are computer-readable. Otherwise, digital resumes will be just images, and companies will have to read them manually.

In most companies, the workload is higher due to the presence of thousands of resumes. For example, companies trying to extract information from old resumes or filter through new job applications may be hampered if done manually. Therefore, extracting useful information from electronic resumes is no longer just a technical challenge. Most specifier resumes.

1.2 Challenges of Information Extraction from Scanned Documents

It is a difficult problem to obtain quality data from scanned documents. First of all, scanned document quality can be poor which can include blurred text, poor lighting, skewed pages, handwritten notes, or background noise making the text hard to read. Second, documents have complex layouts, i.e., multi columns, tables, stamps, signatures, as well as lists, which need to be detected and processed correctly.

Aside from that, texts can have very diverse forms, languages, and handwriting, so the extraction becomes even harder. Common OCR software is hardly able to read things such as this, especially if the structure is poor or even missing.

1.3 Importance in Real-World Applications

The power to extract data from scanned documents routinely is applicable to many industries. With businesses seeking to automate processes and manage vast volumes of data more effectively, the scanning of documents into electronic form with valuable content extraction is at the core of their success. Below are some exemplary real-world applications of extracting data from scanned documents routinely:

- Automating data entry: It is a tedious and time-consuming process of data entry for banking, insurance, as well as government sectors. Handwritten documents and forms are entered quickly through computerized systems that fill customer data, financial data, or application data directly into databases. It helps reduce the workload of humans and minimizes chances of error.
- Resume Processing in HR: Businesses typically receive thousands of resumes, many of which are scanned documents or PDFs. Automated extractions of items such as names, contact details, education, work history, and qualifications simplify it for HR departments to screen, enter résumé data into applicant tracking software, and compare applicants against keyword lists. This improves the recruiting pace and helps to provide fair consideration.

- Archiving and Knowledge Management: Government agencies, museums, and libraries are converting historical documents into digital formats to save culture and make them more accessible. Intelligent document processing enables browsing of archives and retrieval of specific information without the need to view each page.
- Invoice and Financial Document Processing: Organizations that have a high volume of invoices, orders, and receipts can leverage automated applications that extract vendor name, date, amount, and invoice number. Such applications typically integrate with enterprise software to create end-to-end workflows.
- Healthcare and Legal Sectors: Handwritten medical reports, prescriptions, and lab results need to be scanned and digitized for creating electronic health records (EHRs) within healthcare. Legal documents and contracts need to be digitized and searchable for legal research, compliance, and case handling.

These cases demonstrate that information extraction using machines, aside from accelerating the processing and accuracy of data, cuts the cost as well as enables better decision-making. For this purpose, building robust, adaptable, and scalable extracting mechanisms is now a key part of digital transformation.

1.4 Problem Statement

Despite the advancements of deep learning and the developments of OCR, structured data extraction from scanned documents is a multi-problematic issue, particularly in the presence of diverse layouts, document types, and low-quality inputs. Although the majority of the state-of-the-art approaches take a two-stage approach to handling OCR and layout analysis. They lower the performance if documents follow to non-standard templates or contain complex layouts.

The goal of this project is to solve these issues by creating a system that can both interpret the structure of scanned documents and extract valuable information in a structured form. It needs to be able to work with a range of documents and be able to adapt to learning new structures without having to be trained anew.

1.5 Objectives of the Project

The overall goal of this project is to create an automatic system that extracts useful, structured information from scanned documents. The system converts raw visual information in the form of plain text and scanned images to machine-readable, structured data. It aims to:

- Correctly identify the major sections of a given document.
- Convert the scanned document to editable, searchable format.
- extract information like names, date, telephone number, titles, etc., that are the key information.
- Provide a scalable, flexible solution that is suitable for the majority of documents.

It will make the document processing process faster, more accurate, and time-efficient.

1.6 Methodology Overview

The project employs a three-stage modular process for extracting data from scanned documents based on the combination of natural language processing techniques with the concepts of computer vision:

- Document Layout Detection: Identifying the most important document structure elements with an object detection model trained on a custom document data set.
- Text Extraction: Using Optical Character Recognition (OCR) with Tesseract to extract text from detected regions.
- Post Processing (NLP): Pulling information from raw text with the help of Natural Language Processing techniques like regular expressions and spaCy to transform raw text to organized information, including pulling section headers, dates, emails, and phone numbers.

1.7 Report Organization

This document is structured as follows:

- Introduction introduces the context, motivation, and objectives of the project.

- Chapter 1 presents the preprocessing and image analysis techniques used for layout detection and OCR.
- Chapter 2 provides an overview of the current state-of-the-art approaches and tools in document information extraction.
- Chapter 3 explains the proposed pipeline in detail, along with experimental results and a discussion of the findings.
- Conclusion concludes the report by summarizing the main contributions, highlighting limitations, and suggesting directions for future work.

2 Image Processing

2.1 Introduction

In the era of increasing digitalization, visual information plays a pivotal role in how humans and machines perceive and process the world. Image processing, once a specialized technical niche, has become a central pillar of modern computing, ranging from the digitization of basic documents to medical diagnostics and self-driving cars. This discipline lies at the intersection of mathematics and physics, and increasingly of signal processing, and now also of artificial intelligence.

In short, image processing refers to the capture, conversion, examination, and interpretation of digital images to extract useful information or record their visual appearance. Early advances in the early 1960s involved processes such as noise reduction, geometric correction, and contrast enhancement, which are still central to the subject today and are referred to as traditional or classical techniques.

But the growing availability of large datasets, the significant increase in computing power, and the success of machine learning in recent years, particularly deep learning, have marked a turning point. Unlike manually designed filters or rules, modern systems now learn to represent and process images based on examples, and internal representations automatically adapt to perform complex tasks such as object recognition, semantic segmentation, or image generation.

This chapter presents a comprehensive history of image processing around two fundamental principles:

- Traditional methods use mathematical models, manually developed rules, and deterministic filters to perform operations such as enhancement, edge detection, and segmentation.
- Modern AI methods use data-driven models, particularly convolutional neural networks (CNNs), to automatically learn hierarchical representations from images.

This chapter aims to:

Present the theoretical foundations of digital image processing and its key components. Provide examples and describe traditional practices based on real-life experiences;

Highlight the strengths, weaknesses, and complementarities of the two paradigms.

At the end of the chapter, the reader will have a two-part perspective: a solid understanding of traditional image processing concepts and an in-depth understanding of the new AI-inspired approaches shaping the next generation of visual intelligence.

2.2 Digital Image Representation

Image processing starts with a rigorous mathematical definition of an image. In digital systems, an image is usually a two-dimensional function: f(x, y) where x and y are spatial coordinates (columns and rows), and f(x,y) is the brightness value at a particular location. A discretization of both spatial and intensity dimensions creates a digital image.

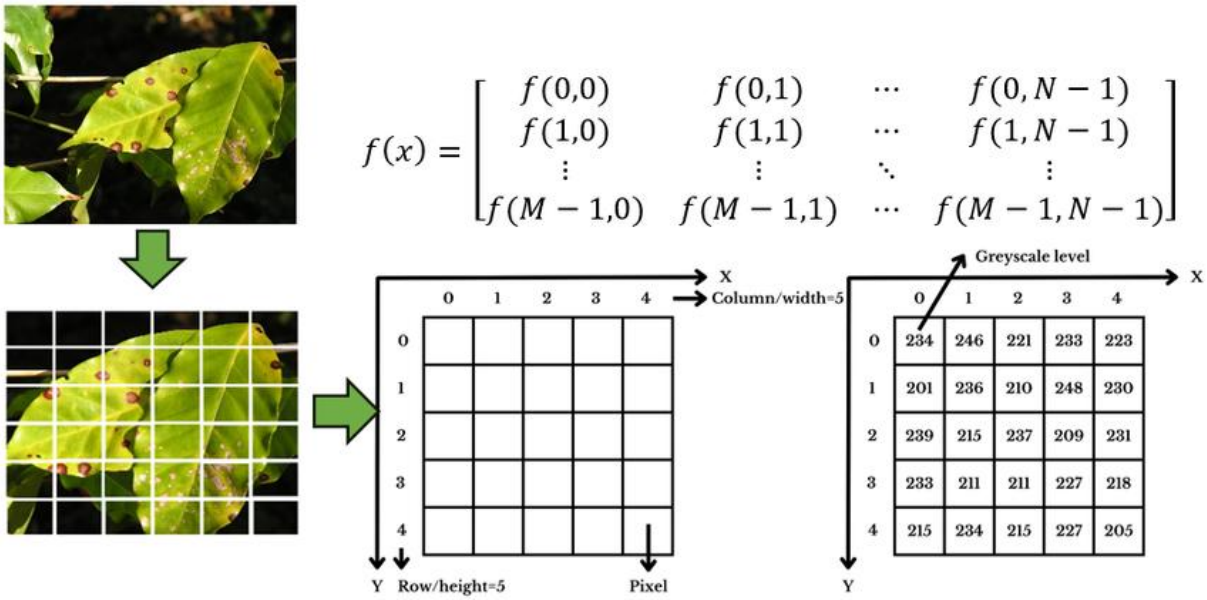


***Figure 2.1**: Representation of Digital image processing*

2.2.1 Pixels and Image Matrix

Is a 2D array of pixels. Each pixel is characterized by its (x, y) coordinates and its value. Digital images are characterized by matrix size, pixel depth and resolution. The matrix size is determined from the number of the columns (m) and the number of rows (n) of the image matrix (m × n).

Or an RGB model color value consisting of three 8-bit values: Red, Green, and Blue.

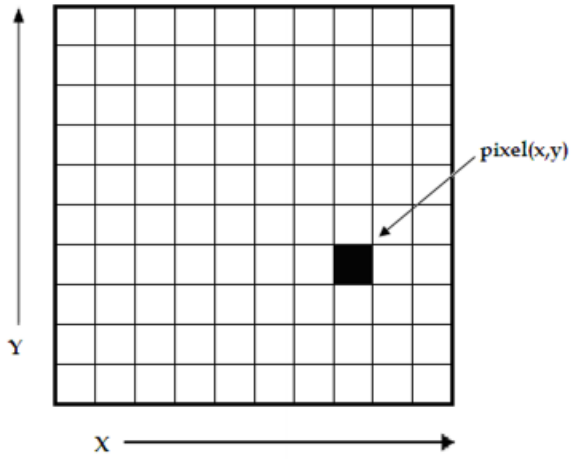


***Figure 2.2**: The input image and its interpretation in pixels Image*

2.2.2 Image Resolution and Size

Spatial resolution is an estimate of the number of pixels per unit length (e.g., pixels per inch – ppi).

Image size is the number of pixels: height × width.

Examples:

- Full High Definition image: 1920×1080 ≈ 2 megap
- MRI scan: typically 256×256 or 512×512

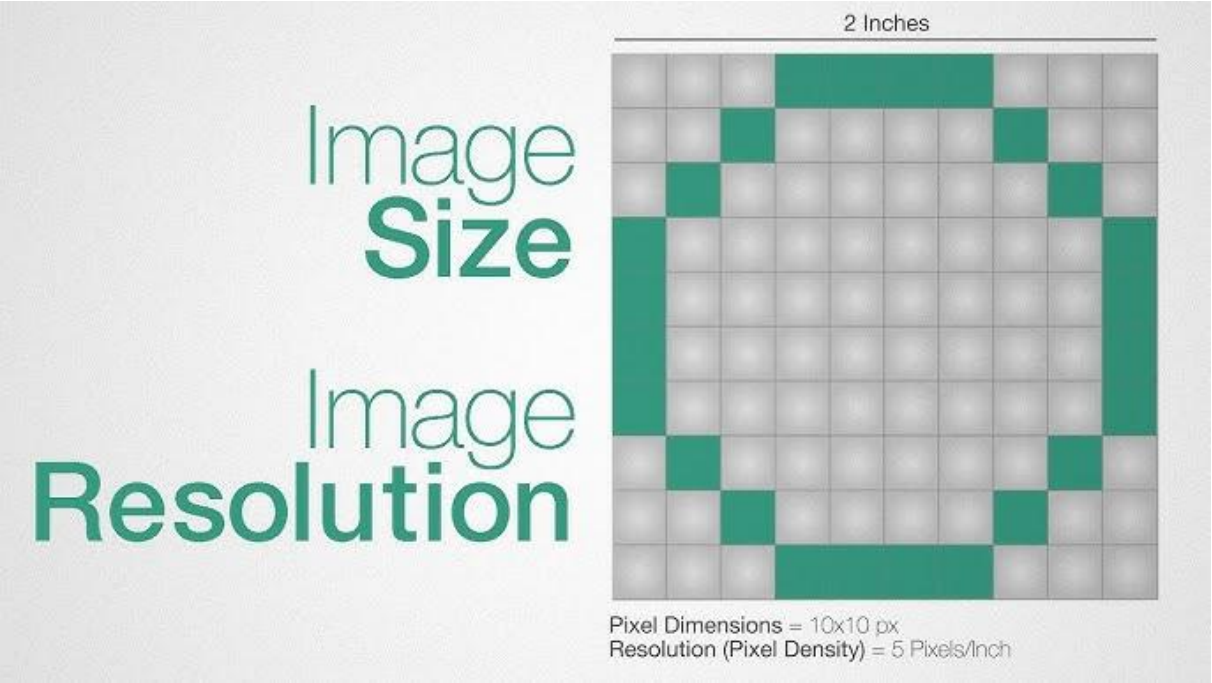


Figure 2.3: *Image Size And Resolution*

2.2.3 Dynamic Range and Color Depth

Color depth is an estimate of the number of bits representing a pixel:

- 8 bits → 256 shades of gray
- 16 bits → 65,536
- 24 bits (8 bits × 3 channels) → Approximately 16.7 million colors

Dynamic range is contrast between shadows and highlights of an image. More dynamic range, more details in shadows and highlights.

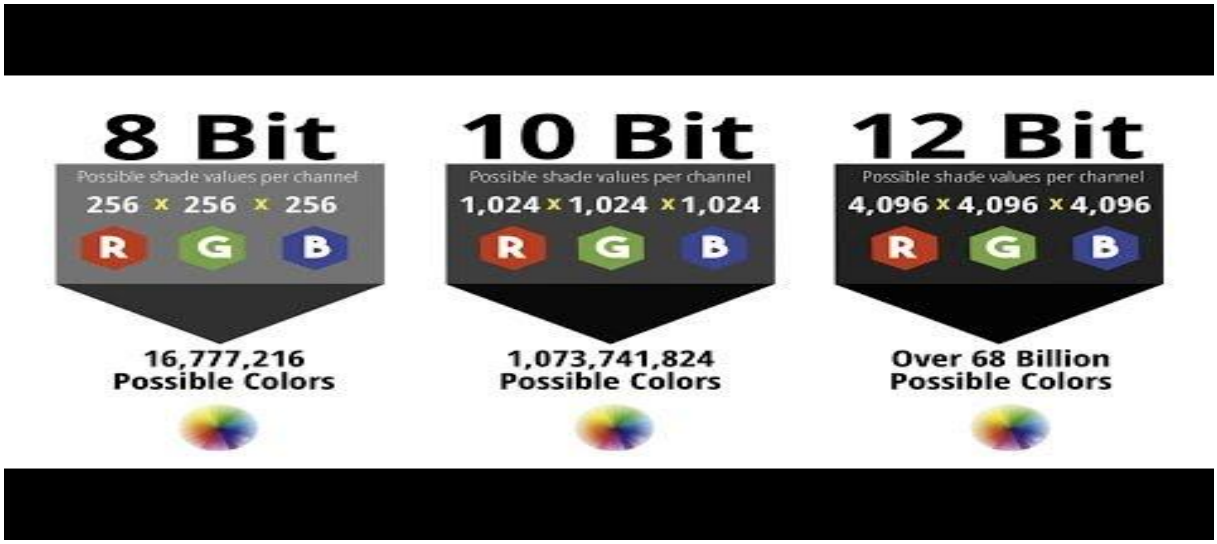


Figure 2.4: *Dynamic Rang and Color depth*

2.2.4 Image Histogram

Graphical representation of intensity value distribution:

- In grayscale images: x-axis = intensity values (0–255), y-axis = frequency.
- In color images: histograms for R, G, and B channels.

Applications:

- Brightness and contrast analysis
- Under exposure or overexposure detection
- Inform operations like histogram equalization or thresholding

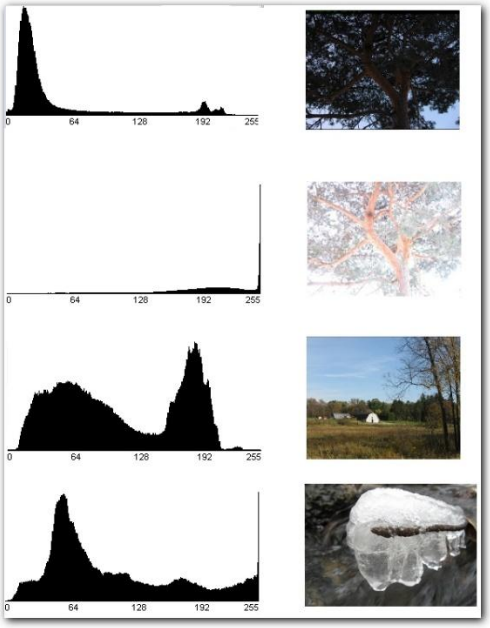


Figure 2.5: *Example of Image Histogram*

2.2.5 Types of Images

- **Binary :** Only 0 and 1 (black and white); masks

Figure 2.6: *Example of Binary Image*

- **Grayscale :**256 levels (monochrome)

Figure 2.7: *Example of Grayscale Image*

- **RGB Color :**Three channels (Red, Green, Blue)

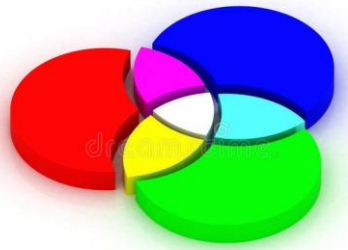


Figure 2.8: *Example of RGB Color Image*

- **Multispectral :**More than 3 channels used in remote sensing

Figure 2.9: *Example of Multispectral Image*

- **3D Volumetric :**Used in medical imaging (CT, MRI volumes)

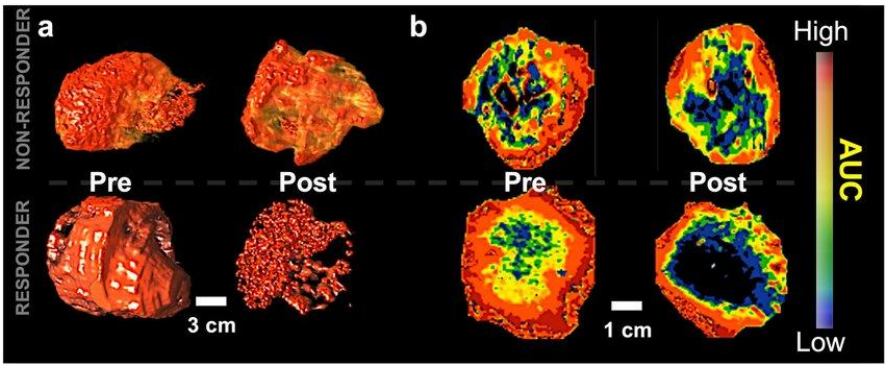


Figure 2.10 : *Example of 3D Volumetric Image*

A clear understanding of digital image representation is essential before applying any processing technique whether classical or AI-based. This foundation allows us to manipulate, analyze, and transform visual information in meaningful ways.

2.3 Image Acquisition and Preprocessing

Before processing or analysis of the image is done, the image needs to be captured in a format that is compatible with computational systems. Image acquisition is the entry point to the image processing pipeline. Acquired images are normally subjected to preprocessing to enhance the quality of the images so that the artifacts that would otherwise cause problems later may be minimized.

2.3.1 Image Acquisition

Image acquisition is the process of capturing an image of a real scene with a sensor and converting it to a digital format.

Primary acquisition devices

- Digital Cameras: Capture images with CMOS or CCD sensors. Suited for consumer applications and machine vision.
- Scanners: Provide high-resolution line-by-line scans, appropriate for artwork and documents.
- Frame Grabbers: Acquisition Cards: Capture analog video signals from devices like security or medical cameras.
- Professional cameras: Infrared cameras, X-ray cameras, multispectral cameras, and hyperspectral cameras for medical, remote sensing, or science.

All acquisition systems are characterized according to:

- Spatial resolution
- Sensor sensitivity
- Noise characteristics
- Bit depth

Digitization steps

1. Sampling: Converts continuous spatial data to a discrete pixel space.
2. Quantization: Assigns numerical values of intensity (gray level or RGB triplet) to every sample point.

2.3.2 Preprocessing

The purpose of preprocessing is to prepare the image for more complex operations such as segmentation or object detection. It typically addresses:

- **Noise reduction**
- **Contrast enhancement**
- **Normalization**
- **Artifact correction**

Technique	Purpose
Histogram equalization	Enhances contrast
Median filter	Removes salt-and-pepper noise
Gaussian blur	Smooths image, reduces high-frequency noise
Normalization	Scales intensity values into a uniform range
Color space conversion	Transforms RGB to HSV, LAB, etc. for better segmentation

Table 2.1: *Common preprocessing techniques*

2.3.3 Importance in the Processing Pipeline

Improper acquisition or preprocessing can greatly impair the performance at later stages such as:

- Edge detection
- Object recognition
- Machine learning inference

Thus, a high quality input is important to provide stable and good outcomes in the process chain.

2.4 Geometric Transformations

Geometric transformations are operations that alter the spatial configuration of an image without modifying its pixel values directly. They are fundamental in aligning, scaling, rotating, or repositioning images—tasks commonly required in image registration, robotics, medical imaging, and data augmentation for machine learning.

2.4.1 Basic Transformations

1. Translation

Moves the entire image or object by a fixed number of pixels in a given direction.

$$\begin{bmatrix} \grave{x} \\ \grave{y} \end{bmatrix} = \begin{bmatrix} x + & tx \\ y + & ty \end{bmatrix}$$

t_xHorizontal and vertical shift values

2. Scaling

Resizes the image either uniformly or non-uniformly.

$$\begin{bmatrix} \grave{x} \\ \grave{y} \end{bmatrix} = \begin{bmatrix} sx.\,x \\ sy.\,y \end{bmatrix}$$

.$S_{x,}, S_y$Scale factors for width and height

3. Rotation

Rotates the image about a point (typically the center).

$$\begin{bmatrix} \grave{x} \\ \grave{y} \end{bmatrix} = \begin{bmatrix} cos\theta - & sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

θ: rotation angle in radians

2.4.2 Affine and Projective Transformations

1. Affine Transformations

Preserve points, straight lines, and planes. Parallel lines remain parallel after transformation.

General affine matrix:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & tx \\ c & d & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2. Projective (Homographic) Transformations

Used when perspective effects are present (e.g., camera calibration, AR). Lines remain lines, but parallelism may not be preserved

2.4.3 Applications of Geometric Transformations

- **Image alignment** (e.g., in satellite imagery or panoramic stitching)
- **Object tracking** (adjusting bounding boxes)
- **Medical image registration** (aligning CT, MRI, and PET scans)
- **Data augmentation** (rotations and flips for training neural networks)

2.4.4 Implementation Notes

- In practice, transformation is applied via **inverse mapping** to avoid holes in the output image.
- **Interpolation methods** such as nearest-neighbor, bilinear, or bicubic are used to estimate pixel values at non-integer coordinates

2.5 Image Enhancement

Image enhancement is the process of employing techniques to enhance the visual quality of an image or to prepare it for future analysis. Enhancement differs from restoration in that restoration seeks to restore the original image, whereas enhancement seeks subjective improvement—enhancing the visual distinction of features or highlighting structures of interest.

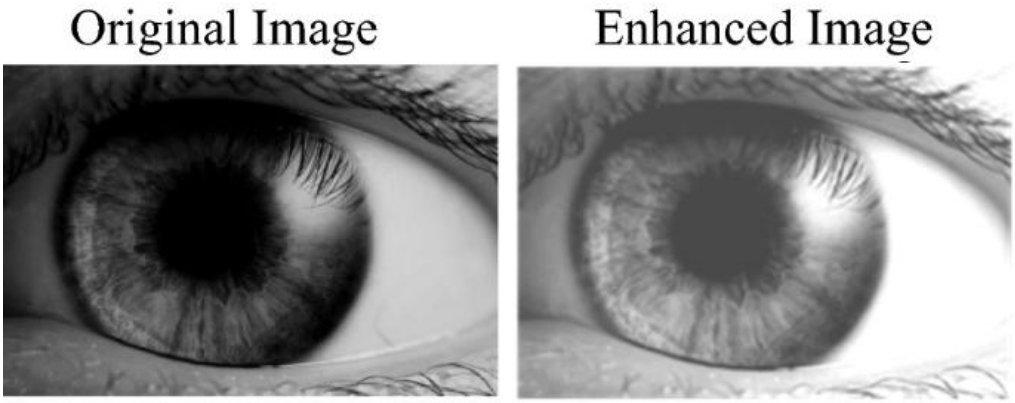


Figure 2.11: *Example Of Image Enhancement*

2.5.1 Purpose of Image Enhancement

Enhancement methods can help:

- Improve human visual interpretation,
- Boost contrast and visibility,
- Highlight edges or textures,
- Enhance features for downstream tasks like segmentation or classification.

2.5.2 Spatial Domain Techniques

These operate directly on pixel values.

1. Contrast Stretching

Enhances contrast by expanding the range of intensity values:

$$f'(x,y) = \frac{f(x,y) - Imin}{Imax - Imin} . 255$$

Where $Imin$and $Imax$are the minimum and maximum pixel intensities.

2. Histogram Equalization

Redistributes pixel intensities to produce a more uniform histogram, improving global contrast.

- Works best for images with poor dynamic range.
- Can also be applied adaptively (*CLAHE* – Contrast Limited Adaptive Histogram Equalization).

3. Log and Power-Law (Gamma) Transformations

Used to compress or expand intensity ranges non-linearly

$$f'(x,y) = c.\log(1 + f(x,y)) \quad or \quad f(x,y) = c.f(x,y)^{\gamma}$$

2.5.3 Frequency Domain Techniques

These manipulate the image in its **Fourier transform** space, often for periodic noise removal or global pattern enhancement.

- **Low-pass filters** (blurring): remove high-frequency noise.
- **High-pass filters** (sharpening): emphasize edges.

Steps:

1. Apply 2D Fourier Transform
2. Filter the frequency spectrum
3. Inverse transform to return to spatial domain

2.5.4 Filtering and Smoothing

Filtering helps remove noise and refine the visual appearance.

Filter Type	Effect	Example Use
Mean filter	Smooths image, reduces noise	General blur
Median filter	Removes salt-and-pepper noise	Document cleanup
Gaussian filter	Preserves edges better	Preprocessing step
Bilateral filter	Smooths while preserving edges	Face smoothing

Table 2.2: *Type Of filter and its uses*

2.5.5 Edge Enhancement

Enhancing edges can make structural information more prominent:

- **Unsharp masking**: Adds a scaled version of the Laplacian or high-pass filtered image back to the original.
- **Laplacian filtering**: Emphasizes rapid intensity changes.

2.5.6 Color Enhancement

- **Color balancing**: Adjusts color channels to correct white balance.
- **Saturation and brightness adjustment**: Performed in HSV or LAB color spaces.
- **Color histogram equalization**: Applied per channel or globally (with caution).

Figure 2.12: *Example of LAB Color Enhancement*

Image enhancement is often the **first visual step** users or algorithms take in understanding an image. When chosen carefully, enhancement techniques can greatly improve clarity and downstream processing performance.

2.5.7 Edge Detection

Edge detection is perhaps the most important operation in computer vision and image processing. It identifies points in an image where intensity changes suddenly those points typically correspond to object boundaries, texture boundaries, or structural contours.

2.5.8 Why Detect Edges?

- To extract **structural information** from images
- To facilitate **object recognition**, **segmentation**, and **shape analysis**
- To reduce data while preserving key features

2.5.9 Types of Edges

Edges can appear in several forms depending on the change in intensity:

Edge Type	Description
Step	Sudden change in intensity (e.g., object edge)
Ramp	Gradual change (e.g., soft shadows)
Roof	Peak in intensity, then return
Line	Narrow edge with intensity drop on both sides

Table 2.3: *Types of Edges and their descriptions*

2.5.10 Gradient-Based Methods

Gradient methods detect edges by calculating the **first derivative** of intensity

- Gradient Magnitude

$$\|\nabla f(x,y)\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2} + \sqrt{\left(\frac{\partial f}{\partial y}\right)^2}$$

- Direction of the Gradient

$$\theta(x,y) = \tan -1\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

Operator	Kernel Size	Description
Sobel	3×3	Good for vertical/horizontal edges
Prewitt	3×3	Similar to Sobel, simpler weights
Roberts	2×2	Fast, but sensitive to noise

Table 2.4: *Discrete Operators*

2.5.11 Laplacian-Based Methods

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

Laplacian (Rafael C. Gonzalez & Richard E. Woods, 2018 (4th edition)) highlights regions of rapid intensity change but is **sensitive to noise**, so it is often used with smoothing filters (e.g., LoG – Laplacian of Gaussian).

2.5.12 Canny Edge Detector

The **Canny algorithm** (John Canny, 1986) is a gold standard in edge detection due to its optimal balance between noise sensitivity and edge accuracy.

2.5.13 Canny steps:

1. **Gaussian smoothing** to reduce noise.
2. **Gradient calculation** (magnitude + direction).
3. **Non-maximum suppression** to thin edges.
4. **Double thresholding** to classify strong/weak edges.
5. **Edge tracking by hysteresis** to finalize edges.

2.5.14 Key strengths:

- Good localization
- Low false detection rate
- Thin and connected edge output

Method	Noise Resistance	Edge Localization	Output Quality
Sobel	Medium	Medium	Medium
Prewitt	Medium	Medium	Medium
Laplacian	Low	Low	Low
Canny	High	High	Excellent

Table 2.5: *Comparison of Edge Detectors*

Edge detection is often a **prerequisite step** in high-level tasks like segmentation, feature extraction, or object tracking. While traditional methods are effective, deep learning-based detectors are increasingly used for more complex or noisy scenes (see Section 6).

2.6 Classical Segmentation Techniques

Image segmentation is the process of dividing an image into meaningful, non-overlapping regions, typically based on pixel similarity. In classical (non-learning-based) image processing, segmentation methods rely on **intensity, texture, or geometric features** to group pixels that share common properties.

2.6.1 Why Segment an Image?

- To isolate **objects of interest** (e.g., organs, vehicles, text)
- To simplify image representation for **feature extraction** or **analysis**
- To enable object-based **classification** and **recognition**

2.6.2 Thresholding

Thresholding is one of the simplest and most widely used techniques for segmenting grayscale images.

1. Global Thresholding

$$g(x,y) = \begin{cases} 1, if\ f(x,y) \geq T \\ 0, otherise \end{cases}$$

- Works well when the histogram is **bimodal** (two distinct peaks).
- T is often chosen using methods like **Otsu's algorithm** (minimizes intra-class variance).

2. Adaptive Thresholding

- Computes local thresholds based on neighborhood statistics.
- Useful for non-uniform illumination or shadows.

2.6.3 Region-Based Segmentation

These techniques group neighboring pixels that are similar based on a homogeneity criterion.

Region Growing

- Start from seed pixels and expand the region by adding neighbors with similar properties.
- Requires a similarity threshold and connectivity rule (e.g., 4- or 8-neighborhood).

Region Splitting and Merging

- Splitting: Recursively divides the image (e.g., via quadtrees) until regions become homogeneous.
- Merging: Combines adjacent similar regions that were over-divided.
- Often combined in Split-and-Merge algorithms for efficiency.

2.6.4 Edge-Based Segmentation

Detects closed boundaries and separates regions based on edge continuity.

Accuracy depends heavily on the quality of the edge detection step (Section 5).

Common techniques:

- Border tracing (e.g., Moore-neighborhood tracing)
- Watershed transformation (see below)

2.6.5 Morphological Segmentation

Morphological operators are particularly effective on **binary or thresholded images**.

Operation	Purpose
Erosion	Shrinks white regions; removes noise
Dilation	Expands white regions
Opening	Erosion followed by dilation; removes small objects
Closing	Dilation followed by erosion; fills holes

Table 2.6: *Types of Morphological Segmentation*

Applied to segmentation masks, these operations help clean and refine object boundaries.

2.6.6 Watershed Segmentation

The **watershed algorithm** treats the image like a topographic surface, where pixel intensities represent elevation.

- Flooding starts from **local minima**.
- Watershed lines are formed where different regions meet.
- Requires preprocessing (e.g., gradient magnitude) and **marker selection** to prevent over segmentation.

Limitation	Description
Sensitive to noise	Small variations can break continuity
Struggle with complex scenes	Difficult to separate overlapping textures
Hard-coded thresholds	May not generalize well across images
No semantic understanding	Can't differentiate between similar-looking objects

Table 2.7 : *Limitations of Classical Methods*

Despite these limitations, classical segmentation techniques remain relevant in well-controlled environments and form the basis for many hybrid or preprocessed pipelines in modern AI systems.

2.6.7 Limitations of Traditional Image Processing

Traditional image processing techniques filtering, edge detection, and thresholding have formed the cornerstone of **computer vision. Although still useful in some contexts, they are crippled by a number of intrinsic** weaknesses, especially when applied to high-variability real-world data.

2.6.7.1 Sensitivity to Noise and Illumination

- Many classical methods, especially those based on gradients or histograms, are highly **sensitive to noise**.
- Variations in **lighting conditions**, shadows, or sensor quality can cause substantial errors in tasks like thresholding or segmentation.
- Without robust preprocessing, these methods often fail on natural images.

2.6.7.2 Parameter Dependency

- Classical techniques rely heavily on **hand-tuned parameters** (e.g., filter sizes, threshold values).
- These parameters must often be **adjusted per image or dataset**, reducing scalability and automation.
- Small changes in parameters can produce drastically different results.

2.6.7.3 Lack of Semantic Understanding

- Traditional methods work at the **pixel level** and do not understand image content.

- For example, they can detect edges but cannot distinguish between a human face and a street sign.
- They cannot generalize patterns or "learn" from data, which limits their adaptability.

2.6.7.4 Rigid Rule-Based Logic

- Rule-based systems (e.g., fixed thresholds, logic conditions) lack flexibility.
- They struggle with **complex scenes** involving occlusions, texture variations, perspective distortions, or background clutter.

2.6.7.5 Difficulty Handling High-Dimensional Data

- Multispectral or hyperspectral imagery involves dozens or hundreds of channels.
- Traditional methods, designed for 1–3 channels, often fail to exploit **inter-channel correlations** effectively.
- Likewise, time-series images (e.g., videos or medical volumes) require models that can track spatial-temporal patterns—something classical tools cannot do efficiently.

2.6.7.6 Absence of Learning or Adaptation

- Traditional algorithms do **not learn** from experience or data.
- They cannot improve their performance over time, adapt to new environments, or generalize from examples.
- In contrast, **machine learning models** (explored in Part 2) can automatically learn robust features and adapt to new inputs.

Limitation	Classical Methods
Generalization	Poor
Noise robustness	Limited
Semantic interpretation	None
Data adaptability	Fixed rules only
Parameter tuning	Manual

Table 2.8: *Summary of classical methods*

2.7 Modern AI-Based Image Processing

2.7.1 Introduction to Data-Driven Image Processing

The past of image processing has taken a revolutionary leap over the last couple of years, driven by the rise of learning-based and data-driven methods. Instead of using hand-designed rules and filters, modern systems learn to process and understand images from examples, with the help of powerful statistical models, most notably deep neural networks.

2.7.2 From Rule-Based to Learning-Based

Approach Type	Description
Traditional Methods	Use fixed filters, thresholding, edge detection, etc.
Learning-Based	Use data to learn patterns, extract features, and generalize

Table 2.9: *Types Of Approach's*

The major reasons for this shift:

- Classical techniques perform poorly in open areas.
- Large-scale annotated image datasets such as COCO and ImageNet.
- Application of high-performance computers such as GPUs and TPUs.
- Deep learning and machine learning platform advancements (TensorFlow and PyTorch).

2.7.3 What Makes Modern Methods Different?

1. **Automatic Feature Extraction**

Instead of manually designing filters to detect features (such as edges and shapes), neural networks **learn optional features** directly from the data.

2. **End-to-End Learning**

Deep learning models are trained to go directly from raw input (pixels) to desired output (e.g., object class, segmentation mask), eliminating the need for intermediate steps like edge detection or histogram analysis.

3. **Generalization**

Well-trained models can recognize objects and structures across **variations in lighting, pose, background**, and **scale**, which traditional methods struggle with

Component	Description
Convolutional Neural Networks (CNNs)	Core engine for visual feature learning
Transfer Learning	Reusing pre-trained models on new tasks
Data Augmentation	Synthetic variation for improved training
Loss Functions	Drive learning toward correct predictions
Optimizers	Algorithms like SGD, Adam for training

Table 2.10: *Key Components of Modern Pipelines*

2.7.3.1 Application Areas

Modern image processing powers a wide range of applications:

- **Medical imaging** (tumor detection, organ segmentation)

- **Autonomous vehicles** (scene parsing, obstacle detection)
- **Satellite and drone imagery** (land cover classification)
- **Security and biometrics** (face recognition, surveillance)
- **Augmented reality and entertainment** (real-time filters, pose tracking)

2.7.3.2 Challenges and Considerations

Despite their power, AI-based methods introduce new challenges:

- **Need for large datasets**
- **Bias and fairness issues** in training data
- **Explain ability** of predictions (deep networks are often black boxes)
- **High computational cost** during training

These are active research areas, and hybrid approaches combining traditional insights with modern learning models are also emerging.

This section sets the stage for a deeper dive into specific AI-based image processing techniques, beginning with the foundational tool of modern vision: the **Convolutional Neural Network (CNN)**

2.7.4 Convolutional Neural Networks (CNNs)

Inspired by the human visual cortex, CNNs (Alex Krizhevsky, Ilya Sutskever, & Geoffrey E. Hinton, 2012) are designed to recognize spatial patterns and hierarchical structures in visual data. They automatically learn to extract and combine visual features through a layered architecture optimized for image understanding.

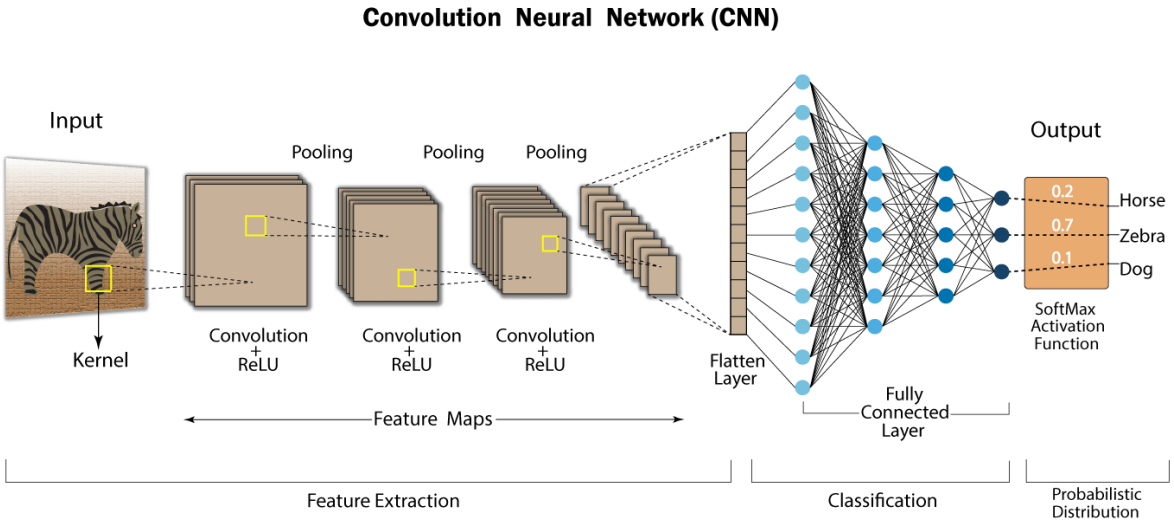


Figure 2.13: Example of Convolution Neural Network Architecture

2.7.5 Why CNNs for Image Processing?

CNNs have revolutionized image analysis because they:

- Exploit the 2D structure of images efficiently
- Share weights across space (translation invariance)
- Automatically learn low- to high-level features (e.g., edges → textures → objects)

They outperform classical methods in tasks such as:

- Image classification
- Object detection
- Image segmentation
- Super-resolution
- Style transfer and generation

2.7.6 CNN Architecture Overview

A typical CNN is composed of a **series of layers**, each transforming the input image into a more abstract representation

Layer Type	Function
Convolutional	Applies filters to detect features like edges, corners
Activation (ReLU)	Introduces non-linearity
Pooling	Downsamples features to reduce size and overfitting
Fully Connected	Maps features to output predictions
Softmax / Sigmoid	Produces final probabilities for classification tasks

Table 2.11: *Key components of Layer Type and Function*

2.7.6.1 Convolution Operation

The **convolutional layer** applies small filters (kernels) that scan over the image to detect local patterns.

$$g(x, y) = \sum_{i=-1}^{k} \sum_{j=-1}^{k} w(i, j).f(x + i, y + j)$$

- f: input image
- w: filter
- Output: a **feature map** indicating where patterns are found

Multiple filters in the same layer allow the network to detect different features.

2.7.6.2 Pooling Layers

Pooling reduces spatial resolution while retaining the most important information. It helps:

- Lower computational cost
- Control overfitting
- Create location invariance

Type	Description
Max Pooling	Takes the maximum value in a region
Average Pooling	Computes the mean value in a region

2.7.6.3 Feature Hierarchies

CNNs build **hierarchical representations** of an image:

This hierarchical structure enables CNNs to generalize across complex and varied images.

Layer Depth	**Learned Feature Example**
Shallow (early)	Edges, gradients, corners
Intermediate	Textures, patterns, shapes
Deep (late)	Object parts, faces, categories

Table 2.12: *layer Depth And Learned Feature Examples*

2.7.7 Training a CNN

Training involves:

1. Feeding images with **labels** (supervised learning)
2. Computing the **loss** (e.g., cross-entropy)
3. Updating weights via **backpropagation** using optimizers like **SGD** or **Adam**

Large annotated datasets (e.g., CIFAR-10, ImageNet) are used to pre-train networks that can then be adapted (via **transfer learning**) to new tasks

2.7.8 Limitations of CNNs

- Require **large amounts of labeled data**
- Sensitive to **adversarial examples**
- Poor at capturing **long-range dependencies** (solved partially by attention mechanisms and transformers)
- Difficult to **interpret decisions** (black-box behavior)

Despite these limitations, CNNs remain the **de facto standard** for visual recognition tasks and are foundational in modern AI-based image pipelines.

2.8 Semantic and Instance Segmentation

Segmentation in the context of deep learning refers to the process of assigning a class label to each pixel in an image. It goes beyond detecting *what* is in an image it also tells you *where* each object is

Type	Description
Semantic Segmentation	Assigns a class to every pixel (e.g., road, tree, sky)
Instance Segmentation	Distinguishes between separate instances of the same class (e.g., 2 cars)
Panoptic Segmentation	Combines both semantic and instance segmentation

Table 2.13: *Types of Segmentation*

2.8.1 Semantic Segmentation with CNNs

1. Fully Convolutional Networks (FCNs)

FCNs (Jonathan Long, Evan Shelhamer, & Trevor Darrell, 2015) were among the first deep architectures tailored for semantic segmentation. They:

- Replace fully connected layers with convolutional layers
- Use **up sampling layers** to recover spatial resolution
- Output a **pixel-level probability map**

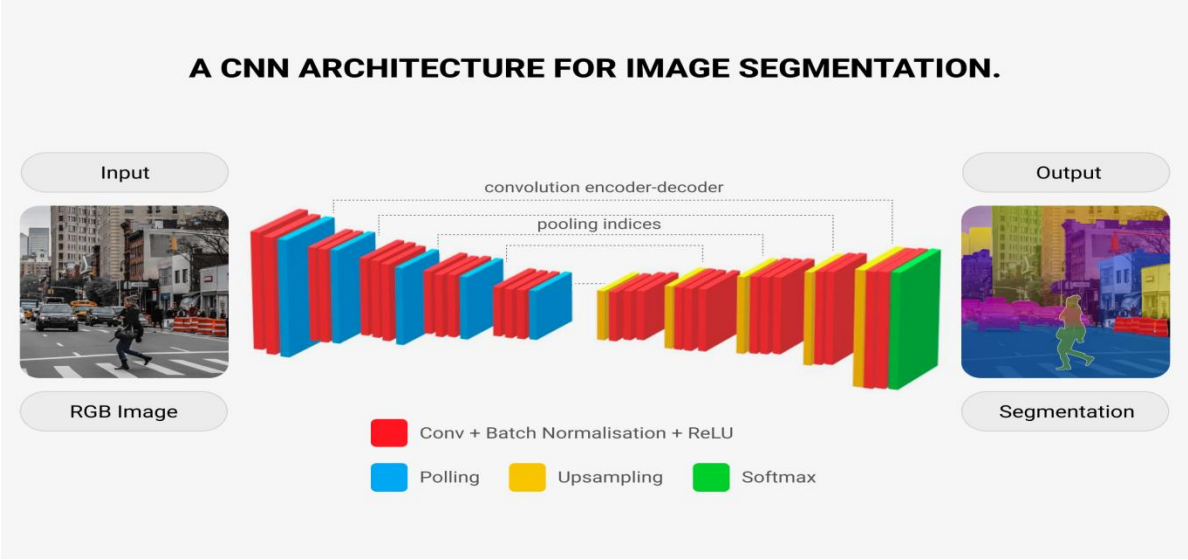


Figure 2.14: *CNN Architecture For Image Processing*

2. **U-Net (for medical and small datasets)**

Originally developed for biomedical segmentation, U-Net (Olaf Ronneberger, Philipp Fischer, & Thomas Brox, 2015) features:

- **Encoder-decoder architecture**: down sampling followed by up sampling
- **Skip connections**: fuse low-level and high-level features for better accuracy
- **Efficient on small datasets**

2.8.2 DeepLab family (DeepLabv3, v3+)

- Use **dilated convolutions** for multi-scale context
- Combine CNN backbones (e.g., ResNet) with **Atrous Spatial Pyramid Pooling (ASPP)**
- Achieve state-of-the-art performance on benchmarks like PASCAL VOC and Cityscapes

2.8.3 Instance Segmentation

While semantic segmentation assigns class labels, **instance segmentation** identifies and separates each occurrence of an object class.

1. **Mask R-CNN**

An extension of Faster R-CNN, Mask R-CNN (Kaiming He, Georgia Gkioxari, Piotr Dollár, & Ross B. Girshick, 2017) adds a branch for predicting object **masks** in parallel with bounding box and class predictions.

Key components:

- **Region Proposal Network (RPN)** (Shaoqing Ren, Kaiming He, Ross B. Girshick, & Jian Sun, 2017): proposes candidate object regions
- **RoIAlign**: ensures precise spatial alignment
- **Mask head**: outputs a pixel-wise binary mask per object

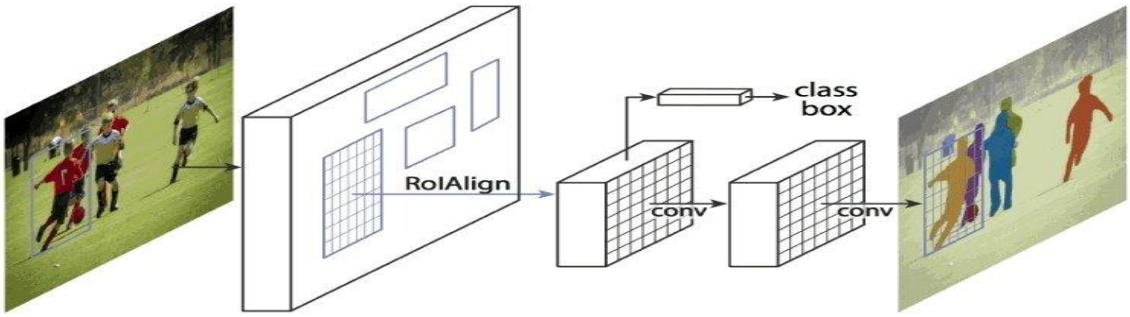


Figure 2.15: *Mask R-CNN Architecture*

2.8.4 Other Approaches

- **YOLACT** (Daniel Bolya, Chong Zhou, Fanyi Xiao, & Yong Jae Lee, 2019): real-time instance segmentation with high speed
- **CondInst**: uses conditional convolution to generate instance-specific masks

Domain	Application Example
Medical imaging	Tumor and organ segmentation (MRI, CT scans)
Self-driving	Road, pedestrian, lane, sign detection
Agriculture	Crop type segmentation, weed detection
AR/VR	Real-time background separation
Industrial	Surface defect detection, part identification

Table 2.14: *Applications of Segmentation*

Metric	Purpose
IoU (Jaccard Index)	Overlap between predicted and ground truth masks
Dice Coefficient	Similar to IoU, commonly used in medical imaging
Pixel Accuracy	Ratio of correctly classified pixels
Mean IoU	Average IoU over all classes

Table 2.15: *Evaluation Metrics*

2.8.5 Challenges in Segmentation

- **Class imbalance** (e.g., small vs. large objects)
- **Precise boundary localization** (especially for tiny or overlapping objects)
- **Real-time performance requirements**
- **Lack of labeled data** in some domains (e.g., medical)

These challenges are being addressed with advanced architectures, **self-supervised learning**, and **synthetic data generation**

2.9 Object Detection with Deep Learning

Segmentation annotates each pixel but object detection detects and positions objects within the image by drawing bounding boxes around them and classifying them into categories. It is central to applications such as surveillance, autonomous driving, robots, and retail analytics.

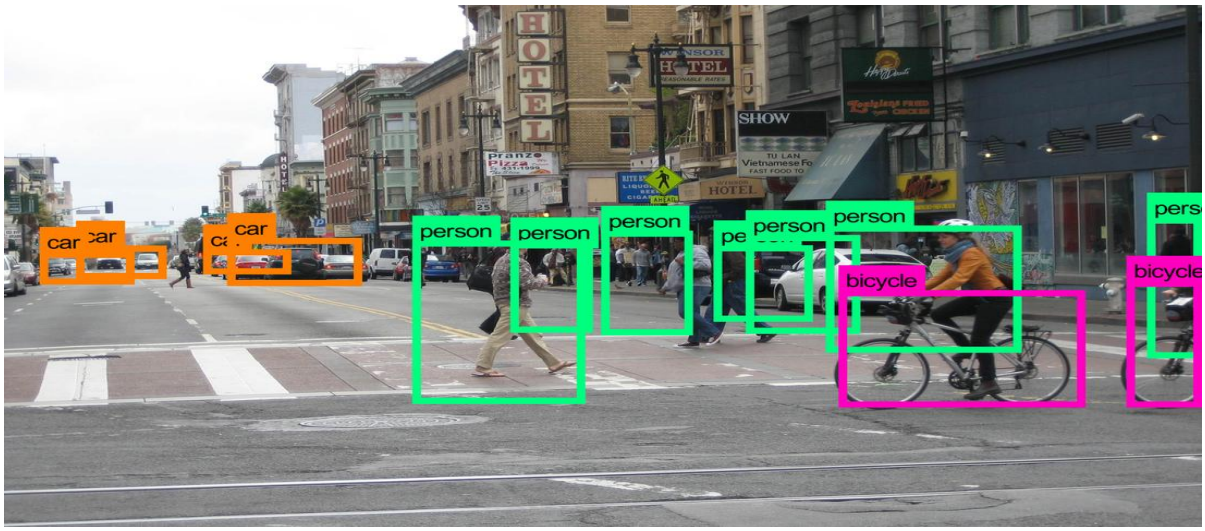


Figure 2.16: *Example of Labeling And Data Annotation*

2.9.1 What is Object Detection?

Object detection answers:

- What is in the image? (classification)
- where is it? (localization)

It delivers:

- One or more bounding boxes
- The respective class label
- Optional: confidence scores

Generation	Examples	Description
Two-stage models	R-CNN, Fast R-CNN, Faster R-CNN	Generate object proposals, then classify them
One-stage models	YOLO, SSD, RetinaNet	Perform detection and classification in one pass
Transformer-based	DETR, DINO-DETR	Replace convolutions with attention-based mechanisms

Table 2.16: *Evolution of Object Detection Models*

2.9.2 Two-Stage Models: Accuracy-Oriented

2.9.2.1 Faster R-CNN

- Introduces the **Region Proposal Network (RPN)** (Shaoqing Ren, Kaiming He, Ross B. Girshick, & Jian Sun, 2017) to generate candidate boxes.
- Uses a CNN backbone (e.g., ResNet, VGG) to extract features.
- Offers high accuracy but can be slower in real-time scenarios.

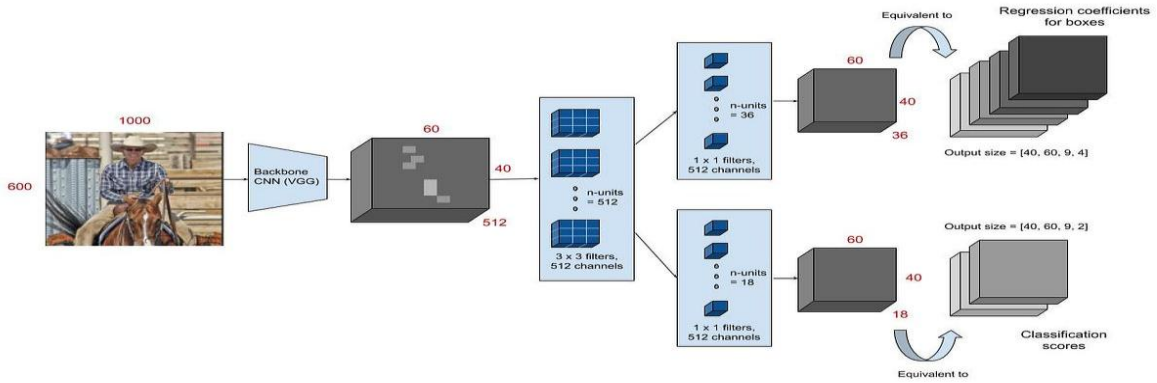


Figure 2.17: *Architecture of Faster R-CNN*

2.9.3 One-Stage Models: Speed-Oriented

2.9.3.1 YOLO (You Only Look Once)

YOLO (Ultralytics, 2023) reframes detection as a **regression problem**:

- Divides the image into an S×S grid.
- Each grid cell predicts bounding boxes and class probabilities.

Variants:

- **YOLOv3**: Fast and accurate for real-time detection.
- **YOLOv5/YOLOv8**: More compact and efficient for deployment on edge devices.

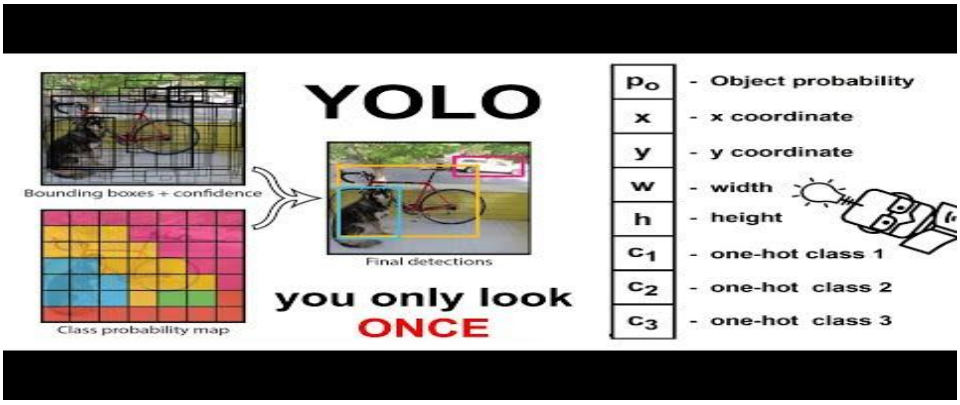


Figure 2.18: *Example Of Detection Object By Yolo*

2.9.3.2 SSD (Single Shot MultiBox Detector)

- Detects objects (Wei Liu, et al., 2016)at multiple scales using feature maps from different CNN layers.
- Good trade-off between speed and accuracy.

2.9.4 Transformer-Based Models

2.9.4.1 DETR (DEtection TRansformer)

DETR (Carion, Nicolas, et al., 2020) reformulates object detection as a direct set prediction task using a Transformer-based encoder-decoder architecture. It eliminates the need for anchor boxes and NMS, simplifying the pipeline but requiring high computational resources.

Component	Role
Classification loss	Predict correct class for each box (e.g., cross-entropy)
Localization loss	Match predicted box to ground truth (e.g., Smooth L1)
IoU loss	Measures box overlap to fine-tune localization

Table 2.17: *Training and Loss Functions*

Metric	Description
mAP (mean Average Precision)	Measures precision at different IoU thresholds
IoU (Intersection over Union)	Evaluates box overlap accuracy
FPS (Frames Per Second)	Measures inference speed (real-time or not)

Table 2.18: *Evaluation Metrics*

Domain	Example Use Case
Security	Intrusion detection, face recognition
Autonomous driving	Detecting cars, pedestrians, traffic signs
Retail	Shelf analysis, customer tracking
Industrial	
Defect detection, package inspection	Identifying lesions or abnormalities

Table 2.19: *Applications of Object Detection*

2.9.5 Challenges

- **Small object detection** (e.g., distant pedestrians)
- **Occlusion and overlapping** objects
- **Real-time inference on edge devices**
- **Training with limited labeled data**

Advanced models now combine detection with **segmentation**, **tracking**, and **language understanding**, expanding capabilities far beyond basic object recognition

In many real-world systems, deep learning is **not used alone**. Traditional image processing techniques are still useful for:

Stage	Classical Technique Used
Post processing	Morphological operations, connected components
Weak supervision	Edge maps or contours used as pseudo-labels

Table 2.20: *Hybrid Image Processing Pipelines*

Example: A pipeline for cell segmentation might use:

- Classical thresholding for seed generation
- CNN for region refinement
- Morphological operations for mask cleanup

This hybrid approach enhances **speed**, **interpretability**, and **data efficiency**.

2.9.6 Vision Transformers (ViTs)

Transformers, initially designed for natural language processing, have recently been adapted to vision tasks with excellent results.

2.9.6.1 Vision Transformer (ViT)

- Divides images into patches (e.g., 16×16)
- Embeds them into a sequence of tokens
- Uses **self-attention** to model long-range dependencies

ViTs outperform CNNs in some benchmarks when trained on **large-scale datasets**, and are:

- Highly parallelizable
- Better at modeling global context
- Flexible across modalities (image + text)

2.9.6.2 Hybrid CNN-ViT Models

Some architectures combine the **local strength of CNNs** with the **global attention** of transformers:

- CvT (Convolutional Vision Transformer)
- Swin Transformer (hierarchical, windowed attention)

2.9.6.3 Self-Supervised and Few-Shot Learning

Data labeling is expensive. New learning strategies reduce the dependence on annotated datasets:

Approach	Description
Self-supervised learning	Models learn from unlabeled data via pretext tasks (e.g., predicting rotation, missing patches)
Contrastive learning	Learn feature similarity/dissimilarity (e.g., SimCLR, MoCo)
Few-shot learning	Train models with very few examples (e.g., Meta-learning)

Table 2.21: *Vision Transformers Approachs*

They are especially useful in

- Medical imaging
- Remote sensing
- Rare object detection

2.10 Conclusion

Here, we have embarked upon a step-by-step tour of the field of image processing, from traditional methods all the way up to state-of-the-art AI systems.

We began with traditional methods

- Model- and hand-designed, and algorithm-based
- Filtering, edge detection, geometrical transformation, and threshold-based segmentation
- Commonly used for decades due to their simplicity, interpretability, and effectiveness.

But we also acknowledged their shortcomings vulnerability to noise, lack of semantic understanding, and rule-based reasoning inflexibility initiating the emergence of data-driven models.

Part II described how image processing was transformed by deep learning:

- Machines are able to automatically learn from the data with CNNs architectures.

- Object detection models like YOLO and Mask R-CNN support real-time recognition and localization.
- Segmentation and its types
- Traditional techniques are still effective, particularly in preprocessing and in low-resource setups.
- Deep learning models are dominating modern-day vision applications due to their performance and flexibility.
- Hybrid pipelines combining traditional logic with learned representations are common in practice.
- The future of image processing is increasingly multi-modal, ethical, and edge-aware.
- Real-time, Low-Power Vision
- Mobile phone, wearables, and embedded system optimized models.
- Quantum and Neuromorphic Vision
- Exploring hardware beyond silicon through quantum vision and brain-inspired chips.
- Aim for Social Good
- Applications in health, disaster recovery, climate, and accessibility

Final Word It is no longer a matter of pixels, but a matter of perception. As machines learn to see, understand, and even imagine the world around them, this field will continue to define the boundaries of artificial intelligence, machine-human interaction, and technological innovation. Whether it is through traditional filters or through generative networks, the end result is the same - converting visual data into actionable intelligence.

3 State of the art

3.1 Introduction

Information extracted from scanned documents is an indispensable task in many fields, e.g., finance, healthcare, and legal. Automated processing of information and documents is required for optimal business performance with minimal human errors. Prior to the advent of deep learning, rule-based methods towards such a task were heavily reliant upon domain knowledge as well as hand-crafted features. Following the advent of deep learning, there have been more efficient and nimble methods, and there have been improvements in terms of precision, alongside scale. The critical factors for effective information extraction are:

• Perception of layout: For inferring relations among different elements, there needs to be perception of document layout, i.e., tables, paragraphs, and key-value pairs. Layout analysis helps make the future systems more efficient in retrieving the relevant information to search and extract.

• Optical Character Recognition or OCR plays an important part whenever we scan documents to render them machine-readable. The quality of OCR decides the quality of the pipeline of information. The progress made in the field of deep learning over the past couple of years revolutionized the quality of OCR even when the quality is poor, i.e., scanned or handwritten.

• Post-processing: Post-processing techniques, i.e., natural language processing (NLP), and machine learning (ML), are utilized when extracting text and layout information to improve the quality of extracted data. These techniques detect errors, recognize relevant data, and make the output structured and usable. In the following, we present the state of the art for all of these building blocks, with an emphasis on layout analysis, OCR, and post-processing. In addition, we describe end-to-end pipeline systems which combine these building blocks into high-performance systems for document understanding

3.2 Document Layout Analysis

Document layout analysis is a critical task in information extraction from a scanned document. It is used for the identification and understanding of the physical layout of the document with a view towards locating different elements such as headings, paragraphs, tables, forms, and images. Layout analysis is to understand the visual layout of the document for extracting useful information from specific regions such as key-value pairs or tables.

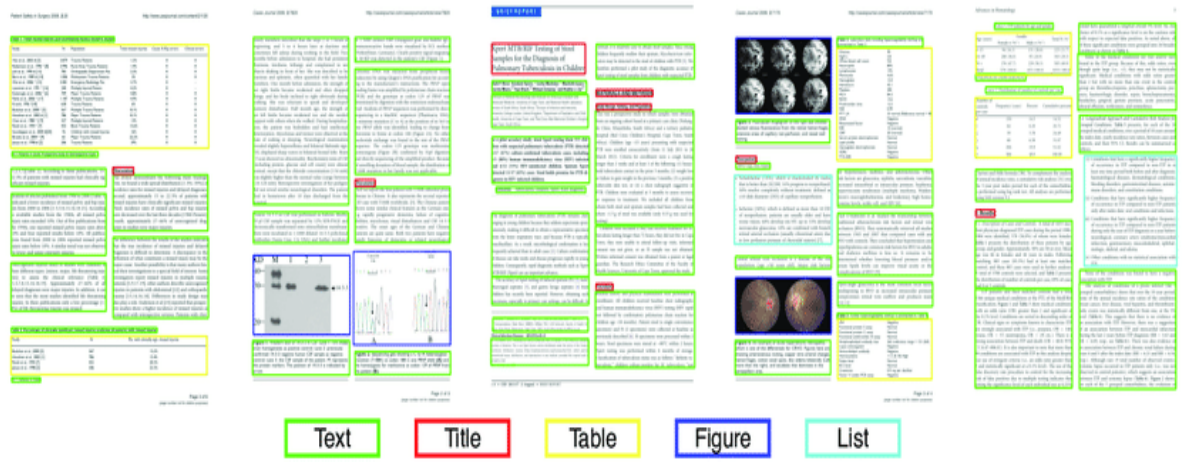


Figure 3.Representative *examples of the document layout analysis*

3.2.1 Traditional Methods

Before the era of deep learning, layout analysis techniques were rule-based. These were generally composed of heuristics and hand-written rules to identify and distinguish between various layout components.

3.2.2 Rule-based Segmentation

Rule-based segmentation is a traditional method used for segmenting digitized or scanned documents into individual regions based on pre-defined rules and heuristics.

Document segmentation is a fundamental step for document layout analysis that helps the system identify and distinguish individual components of a document, e.g., text block, images, table, headers, and footers. The primary purpose of rule-based segmentation is to provide an accurate structural representation of content within a document that is process able for subsequent OCR, information extraction, or post-processing.

Key Concepts

Rule-based segmentation involves applying a set of well-defined rules to divide a document according to its visual and textual features. These rules often rely on spatial information (such as coordinates), text density, and the presence of visual elements like lines or boxes. Unlike machine learning approaches, rule-based methods do not require large training datasets and are especially effective for simple, well-structured documents.

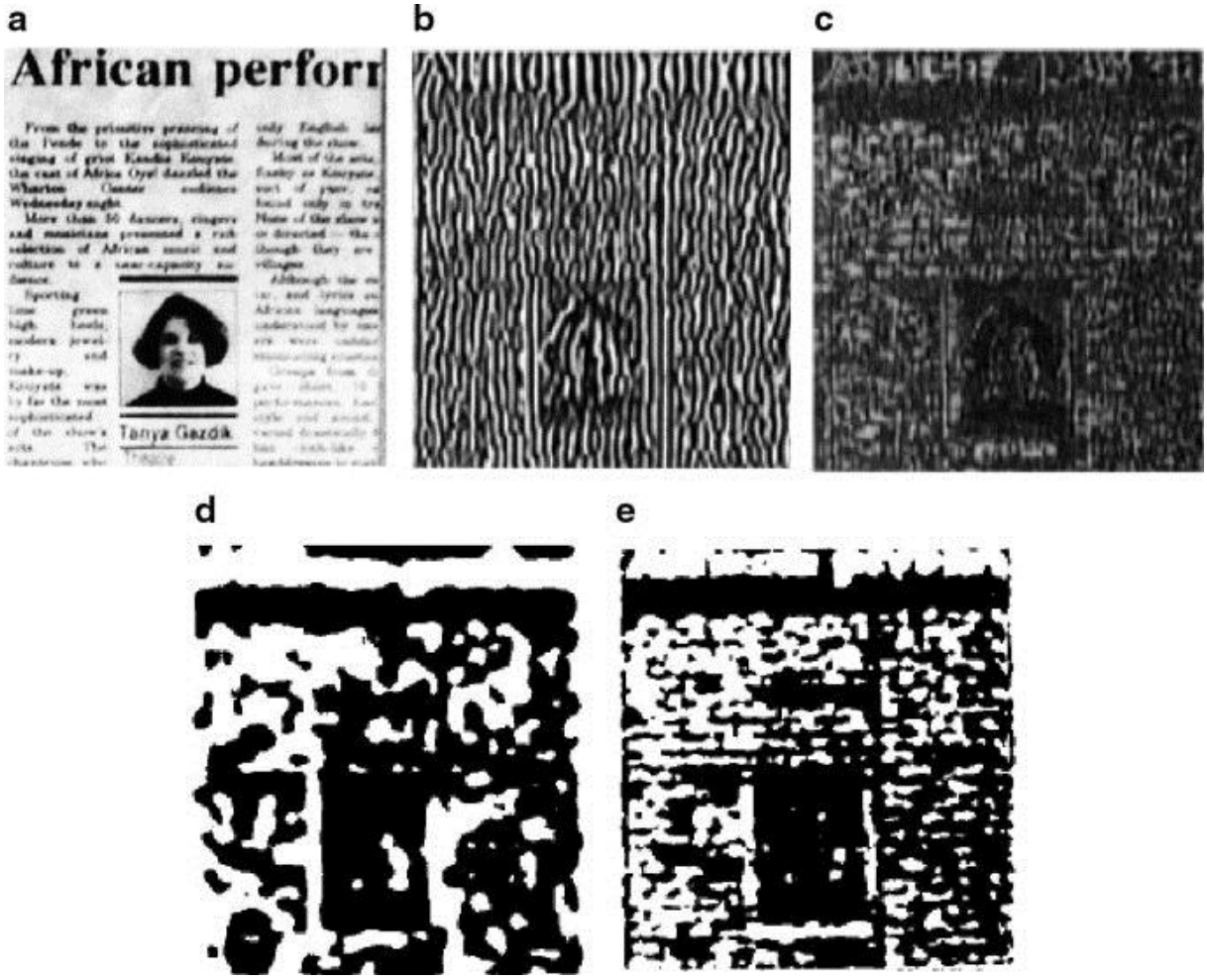


Figure 3.1: *Page segmentation techniques in document layout analysis*

3.2.3 Methodologies in rule-based segmentation

There are several rule-based methods in common use.

3.2.3.1 X-Y Cut Method

The X-Y (K. T. Wong, A. Cai, & P. Shi, 1995) cut is the simplest and most common method of document partitioning. It is an operation of two cuts in the document, one along the X-axis (horizontal) and one along the Y-axis (vertical), along the distribution of the text and whitespace in the document, and divides it into areas.

- **X-Cut:** The text is divided into horizontal slices by cutting along text lines. The engine reads the file line-by-line and detects lines or text blocks. When there is significant white space separating two lines of text, this is a sign of a break among two independent parts (e.g., between the header or the paragraph and the body text).
- **Y Cutting**: Following horizontal cutting, the system further divides the text blocks into columns or parts in the direction of the y-axis. It can also be utilized for detecting multi-column documents, where the text is broken into individual columns (e.g., newspapers or newsletters).

Benefits:

X-Y cut technique is computationally straightforward and quick, and it is, therefore, suitable for documents with well-defined visual organization.

Advantages:

This technique falters with documents with complicated structures where there are more than simple row-column structures involved (e.g., documents with non-rectangular text areas or irregularly shaped layouts).

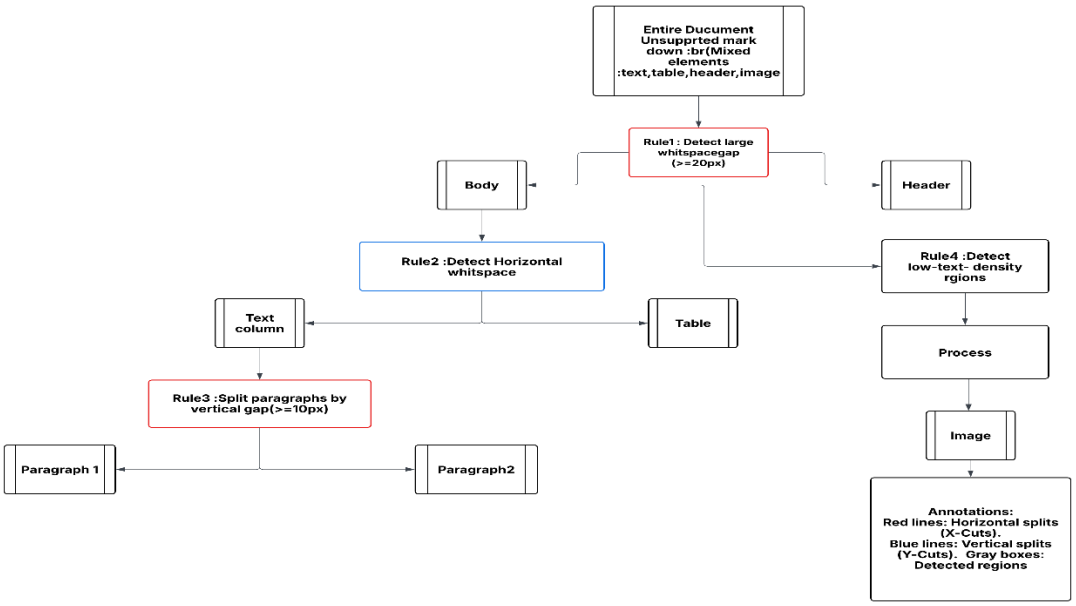


Figure 3.2: *Representative diagram of the X-Y cut method*

Projection Profiles

Projection profiles are another technique used for rule-based segmentation. This method computes the horizontal and vertical projections of the document, which essentially calculates the sum of pixel intensities along the horizontal and vertical axes. The projection profile can be used to identify the boundaries of text blocks, images, and other components.

- Horizontal Projection Profile: This technique computes the sum of pixel values along each horizontal line of the document image. Areas with high pixel intensity correspond to lines of text, while regions with low pixel intensity represent blank spaces. The peaks and valleys in the projection profile correspond to the top and bottom boundaries of text blocks, respectively.
- Vertical Projection Profile: Similar to the horizontal profile, the vertical projection profile computes the sum of pixel intensities along each vertical column. This is useful for detecting column boundaries in multi-column documents.

- Advantages: Projection profiles are relatively simple and fast to compute, and they work well with structured documents that follow a regular grid layout.
- Disadvantages: Projection profiles may not perform well on complex documents with irregular layouts or overlapping components (e.g., when text and images are interspersed).

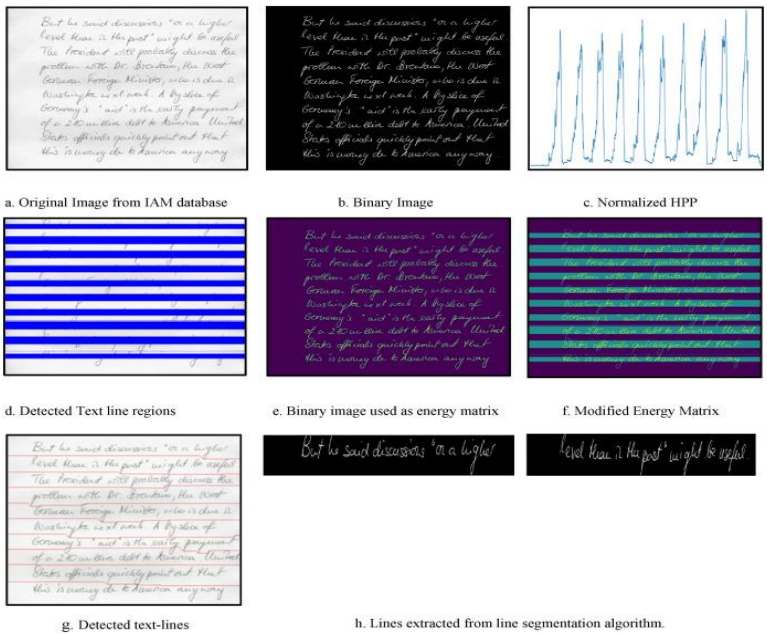


Figure 3.3: *Horizontal projection profile and contour tracing for line and word segmentation*

3.2.4 Connected Component Analysis:

Connected component analysis, or CCA, is an operation where all pixels of the document are treated as graph nodes. The algorithm finds groups of pixels that belong to common objects (i.e., image, text, or graphic). These groups are then combined into larger regions.

Text Region Identification:

Text regions in an ordinary document are often represented as character-related parts. Through examining component connectivity's, the algorithm can group them into text areas.

3.2.4.1 Line Segmentation

Once the regions of text are identified, these elements are then segmented into lines. This involves analyzing the spatial relationship of the elements and segmenting them according to proximity to each other.

3.2.4.2 Page Segmentation

Subsequent to line segmentation, the method can then further be utilized to segment the identified text units, images, and other page features into higher-level units, i.e., into columns, paragraphs, and headers.

Advantages

CCA performs very well in distinguishing areas of text and images in documents which have clear boundaries and well-separated objects. Fine-grained feature

partitioning can also be achieved with CCA, which can handle documents with irregular layouts to a certain extent.

Limitations

It is challenging when there is superimposition of text, images, or graphics over documents. It also requires meticulous parameter calibration, e.g., pixel connectivity, for optimal performance.

3.2.5 Use of Rule-Based Segmentation

Rule-based segmentation is especially well suited for use in documents with predictable, regular structures, such as:

- Forms: Forms with well-defined fields (e.g., address, signature, and name) can be broken down with rule-based methods.
- Invoices: The receipts and invoices have pre-designated places for item listings, sums, and dates, and hence are good candidates for rule-based segmentation.
- Legal Documents: Legal documents with properly structured, standardized form (e.g., contracts, agreements) are likewise suitable for processing with rule-based methods.
- Text-based reporting with clearly organized sections and paragraphs can best take advantage of rule-based segmentation techniques.

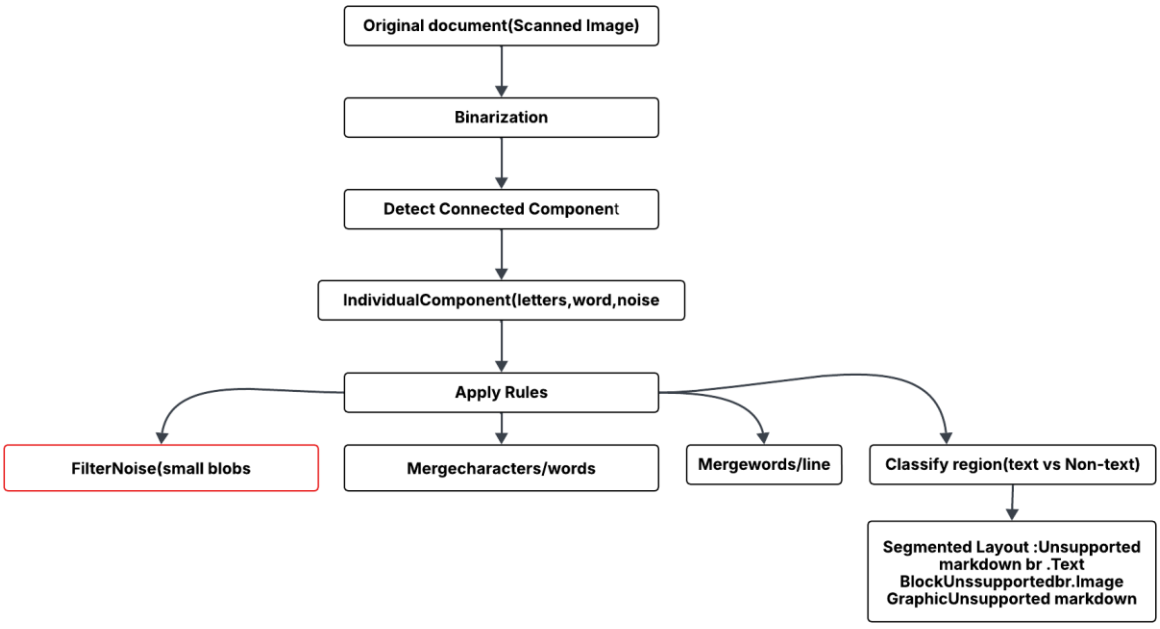


***Figure 3.4**: Connected Component Analysis (CCA) Segmentation Workflow*

3.2.6 Limitations and Difficulties

Despite its strengths, rule-based segmenting also has some drawbacks:

• Sensitivity to layout inconsistencies: rule-based methods do not fare well with documents with nonstandard layout or inconsistent layout style. Such documents have inconsistent styled layout or combinations of content types (e.g., mix of text and image in the same area).

• Inflexibility: The rule-based methods are inflexible and do not support new document structures or new formats unless the rules are manually reconfigured. The inflexibility restricts optimization of rule-based methods in changing environments.

• Propagation of Errors: A single mistake in a segmentation step (e.g., wrong text block bounds) can propagate all the way through the document understanding pipeline, impairing the quality of OCR and other extracted data.

3.2.7 Algorithmic Foundations

The foundation of rule-based segmentation lies in the heuristics derived from document geometry. For example, equal application of margins, line heights, and font sizes can be utilized for developing segmentation rules.

One standard approach is to binaries the image initially (e.g., Otsu's thresholding), followed by morphological treatment (e.g., dilation, erosion) to enhance such elements like table edges and lines of text. Features like aspect ratio, size of connected components, and spatial relationships govern the segmentation reasoning afterwards.

3.2.8 Hybrid Rule-based Systems

Many modern systems employ hybrid techniques combining rule-based segmentation with ML classifiers for instance:

- Use rules to detect probable text regions.
- Apply a lightweight CNN to classify whether a region is a paragraph, table, or figure.

Such pipelines benefit from the interpretability and speed of rule-based heuristics and the adaptability of ML models.

Specialized Use Cases

Some documents have unique structures that are best handled with rule-based logic:

- Passports and ID cards: Specific zones like MRZ (Machine Readable Zone) follow strict formatting.
- Bank checks: Regions like date, payee, and amount are fixed by regulation.
- Academic transcripts: Repeated structures in tabular form are often easy to extract with predefined row/column logic.

Aspect	Rule-based	Deep Learning
Data requirement	None	High(thousands of labeled images)
Adaptability	Low	High
Interpretability	High	Often opaque
Processing speed	Fat (real-time)	Depend on model complexity
Layout Flexibility	Poor for irregular layouts	Robust to complex design
Hardware Requirement	Low	Require GPU/TPU for training

Table 3.1: *Comparison with Learning-based Approaches*

3.3 Recent Techniques

3.3.1 Layout Analysis using Deep Learning

Deep learning techniques transformed layout analysis over the last three years with stronger, more precise, and more flexible means of document structure understanding and layout extraction. Deep learning models outshined most rule-based and heuristic techniques with the capability to handle complex and heterogeneous layouts more accurately. This significant shift is triggered by uses of Convolutional Neural Networks (CNNs), which achieve excellent performance with image data in the shape of document images. CNNs and other deep learning architectures like Transformer-based architectures have enabled auto-extraction of layout items like text blocks, images, tables, figures, and headings from digital images or scanned documents. These methods overcome conventional computer vision methods by learning from data directly without human intervention or the requirement of handcrafted features.

Deep learning-based layout analysis methods are particularly useful in real-world applications where documents have diverse structures and layouts. These methods are capable of learning to generalize across a number of layouts and even on noisy or distorted images, thus proving to be very helpful in applications like Optical Character Recognition (OCR) and document understanding. Also, since they can use huge annotated data sets and are trained on challenging document types, deep learning models can attain human-like performance in the majority of scenarios and are thus the solution of choice in current document processing systems.

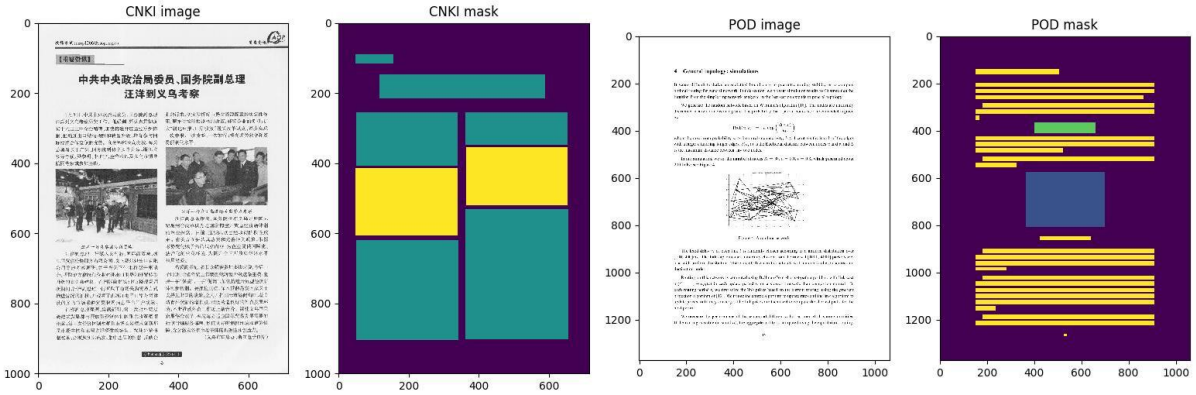


Figure 3.5*: Implementation of deep learning in Document Layout Analysis*

3.3.2 YOLO Family (YOLOv3 to YOLOv8)

The "You Only Look Once" (YOLO) (Ultralytics, 2023) detector family has made object detection in real time, and their application in document layout analysis has experienced staggering gains in the speed and effectiveness of document processing. YOLO is a one-stage detector that does classification along with localization in a single pass in the forward direction, i.e., can process images rapidly with no compromise in performance. Thus, YOLO is best suited to real-time applications where timely response is most critical, such as document scanning and processing on mobile devices or embedded systems.

The YOLO models are used extensively for layout analysis since they are capable of detecting multiple document elements at the same time and hence are ideal to extract complicated document layouts from scanned documents. The YOLO family has evolved from YOLOv3 to YOLOv8, and every version has become more accurate, efficient, and generalizable. These models are specifically used for the detection of text blocks, headings, tables, figures, and logos from scanned documents, which typically appear in the form of scientific papers, invoices, receipts, and forms.

3.3.2.1 Working Mechanism of YOLO Models

YOLO (Ultralytics, 2023) models divide an input image into a grid and estimate the bounding boxes of objects of interest within the grid. Simultaneously, they allocate each bounding box class probabilities so that the model can detect and classify multiple objects simultaneously. By doing so, YOLO can identify various parts of a document within shorter processing times, thus low latency, and hence the system being suitable for real-time document analysis systems. YOLO models must employ anchor boxes to make bounding box size predictions, and by training on big sets of documents with annotations, they can become proficient at locating very dissimilar parts of the document even on complex layouts.

3.3.2.2 Applications of YOLO Models in Document Layout Analysis

YOLO (Ultralytics, 2023) models have also been used effectively to a number of different document layout analysis tasks:

Text Block Detection: Text area detection in a document, most important in OCR systems for text material detection and processing.

Heading Detection: Heading and subheading detection to structure content of a document for easier navigation and understanding.

Table Detection: Table and cell detection to support structured data extraction from documents.

Figure and Image Detection: Image detection, charts, and graphs that are generally integrated in the visual content of a document.

Logo Detection: Logo or watermark detection one can find on documents for authentication or branding.

Pre-trained YOLO models are exported in such applications as PubLayNet and DocLayNet to document classes, be they scientific papers or bills. When exporting, models better identify the document structure and document format of a specific field, which boosts their performance and accuracy further.

3.3.2.3 YOLOv5 and YOLOv8

YOLOv5 is among the most popular PyTorch-based YOLO versions. It is very popular due to the fact that it is simple and efficient. YOLOv5 comes with anchor learning and mixed precision training capabilities, which improve training efficiency and inference speeds.

YOLOv8 by Ultralytics is a better version of its predecessors. It has fewer architectures and newer accuracy advancements, making it faster and more efficient. Apart from this, YOLOv8 also has an important addition: instance segmentation. This allows the model not only to identify boxes but also to segment a single object within a box. This feature is particularly well worth the addition for precise boundary detection in the layout, where proper separation of overlapping fragments (such as text blocks or figures) is essential.

3.3.2.4 YOLOv11: A Radical Leap in Object

Ultralytics' YOLOv11 is a step forward in the development of the You Only Look Once series of real-time object detection models released in September 2024. The model boasts notable architectural changes that make it more accurate and effective.

3.3.2.5 Architectural innovations

- YOLOv11 integrates a number of state-of-the-art components to enhance feature extraction and computation performance:
- C3k2 (Cross Stage Partial kernel size 2): Improves feature representation while mitigating redundant calculations
- SPPF (Spatial Pyramid Pooling - Fast): Enabling rapid spatial pooling of information across different scales.

3.3.2.6 Parallel Spatial Attention Convolution (C2PSA)

Enhances the model ability to attend to significant regions in the image through parallel attention.

These improvements enable YOLOv11 (Ultralytics, 2023) to attain a better mean Average Precision (mAP) on the COCO dataset but with 22% fewer parameters than YOLOv8m. Consequently, YOLOv11 is lighter and faster.

Flexibility in Supported Activities YOLOv11 is designed for a wide range of different computer vision tasks and is thus a highly versatile (Ultralytics, 2023) model

- Object Detection: Identifies and detects multiple objects in an image.
- Instance Segmentation: Generates accurate masks for each instance of an object.
- Image labeling: Assigns one tag to an entire image.
- Human pose estimation involves finding out key body points.

- Oriented Bounding Box Detection (OBB): Handles detection of rotated objects—particularly useful for skewed document photos or aerial images.

This type of flexibility makes YOLOv11 especially suitable for document layout analysis where accurate positioning of things such as headings, paragraphs, tables, and images is essential.

3.3.2.7 Performance and deployment

There are several variations of YOLOv11 (Ultralytics, 2023) ranging from nano to extra-large that balance speed against accuracy depending on specific deployment needs.

- YOLOv11n (nano) : Optimized for low-power edge devices
- YOLOv11x (extra large): Focuses on maximum accuracy for fine-grained detection

Furthermore, YOLOv11 is also deployable on a large variety of platforms ranging from edge and cloud-based systems to systems equipped with GPUs (e.g., NVIDIA systems), for maximum flexibility for developers.

Feature	YOLOv5	YOLOv8	YOLOv11
Release Year	2020	2023	2024
Framework	PyTorch (Ultralytics)	PyTorch (Ultralytics)	PyTorch (Ultralytics)
Instance Segmentation	None	Yes	Yes
Rotated Object Detection (OBB)	None	None	Yes
Pose Estimation	None	Limited	Yes
Speed and Efficiency	Fast	Faster	Fastest (optimized with fewer params)
Edge Deployment	Yes	Yes	(lighter with same accuracy)
Model Sizes	s, m, l, x	n, s, m, l, x	n, s, m, l, x
Advanced Attention Mechanisms	None	Limited	C2fPSA, C3k2)
Best For	General OCR tasks	Overlapping layout segmentation	Multitask doc layout with OBB & pose

***Table 3.2: YOLO** Model Comparison for Document Layout Analysis*

3.3.2.8 Benefits of YOLO Models in Layout Analysis

Fast Inference Time: YOLO models are optimized for real-time object detection, i.e., they can produce output at quick rates, which is beneficial for applications that need to process fast, i.e., mobile apps and embedded systems.

Good Generalization: Domain-specific YOLO models fine-tuned on such datasets are of good generalization, i.e., capable of dealing with varied document layouts and forms. This proves best for document processing tools using documents with mixed sources like invoices, receipts, or research documents.

Effective Deployment: YOLO models are lightweight and can be deployed on edge devices, thus they can be used in applications where computational resources are constrained, i.e., smart phones, IoT devices, or embedded systems.

3.3.2.9 Drawbacks of YOLO Models in Layout Analysis

- **Bounding Box Accuracy:** While YOLO models are fast and efficient, they may not be as accurate in the bounding box when the parts overlap heavily or are of unusual shapes. Here, the model predictions could be less precise, and this may lead to document interpretation issues.
- **Instance-Level Limited Knowledge:** YOLO models, being a basic example, are object detection-focused but not necessarily endowed with in-depth knowledge of instance-level inter-component relationships. This limitation can be overcome with the use of versions like YOLOv8-seg, possessing instance segmentation capability for better disentanglement and comprehension of overlapped components.

Overall, the YOLO series of models have served wonderfully for real-time document layout analysis tasks. As model design, training practices, and domain-specific components like instance segmentation develop further, YOLO models keep pushing limits of what can be done in document processing and analysis. Nonetheless, as is the case with any machine learning method, they are not without their flaws, particularly when it comes to processing complicated layouts or where precision matters most.

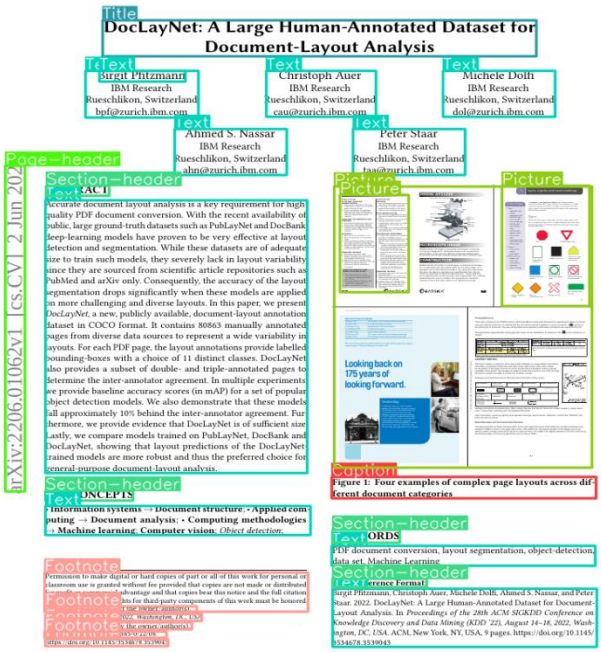


***Figure 3.6**: How to Analyze document Layout by Yolo*

3.3.3 Faster R-CNN and Mask R-CNN

Powerful Two-Stage Architectures for Document Object Detection and Segmentation
Faster R-CNN and Mask R-CNN are pioneering models in computer vision, particularly object detection and segmentation. They have a two-stage detection model, which, while computationally more costly than one-stage models, is far more accurate and precise—attributes that are especially desirable when working with document image analysis.

3.3.4 Faster R-CNN

R-CNN (Region-based Convolutional Neural Network) (Girshick, 2014) is a fast object detection framework that introduced the concept of an end-to-end trainable Region Proposal Network (RPN). The model operates in two consecutive phases:
In the first step, the RPN scans the input image to propose a set of candidate regions in which the object of interest might be located. The proposals are generated with learned convolutional features and therefore the approach is far more efficient compared to previous methods like selective search.
Stage two sees the recommended areas being handed to a bounding box regressor and a classifier, which simultaneously predict the class of the object and adjust its spatial location.
The framework enables the model to accurately define and classify the location of an object in a document, as in text regions, tables, figures, or icons, and has been prevalently applied for layout analysis processes owing to the fact that it has robust localizing capabilities.

3.3.5 Mask R-CNN

Mask R-CNN adds to the Faster R-CNN architecture an extra segmentation branch that is executed in parallel with the classification and bounding box regression tasks. The third branch predicts a binary mask for every object, enabling instance-level semantic segmentation.
In document analysis, this addition offers a number of strong capabilities:
Fine-grained segmentation of document elements such as form fields, handwritten signatures, annotations, and graphical elements.
Effective disambiguation of rival elements, such as closely clustered paragraphs and tables that occupy common visual space.
Pixel-level understanding of structure, necessary to rebuild rational hierarchies within complex documents such as scientific articles, contracts, and technical reports.

3.3.5.1 Benefits in Document Processing

Instance Segmentation: Unlike bounding box detection, which offers coarse localization, instance segmentation can provide extraction of precise shapes and edges—a central aspect for understanding complex page layout.

Overlapping Element Differentiation: The models excel at disentangling overlapping or nested objects, which in many cases present themselves in richly formatted text.

3.3.5.2 Limitations and Considerations

Though they are highly accurate, these models have significant trade-offs:

- **High Computational Cost**: Because of their multi-stage processing and convolution-rich architecture, Faster R-CNN and Mask R-CNN both require a high amount of computational resources. They are generally slower during inference than one-stage detectors such as YOLO, making them unsuitable for real-time applications.

- **Data Dependency:** Annotation of such big models from the beginning requires vast amounts of labeled data. Moreover, fine-tuning them to domain-specific document layout normally needs accurate bounding boxes and segmentation masks.

- **Infrastructure Requirements**: Production deployment normally needs high-end GPUs, which may not be feasible in low-resource environments.

3.3.6 Donut

OCR-Free Transformer for End-to-End Document Understanding
Donut (Document Understanding Transformer) is a revolution in document image processing technology. Unlike traditional pipelines, which are heavily reliant on Optical Character Recognition (OCR) for text extraction, Donut completely does away with the OCR step and offers a fully end-to-end solution that takes raw document images as input and produces structured textual content as output.

3.3.6.1 Core Characteristics

- OCR-Free Architecture: Donut processes the content of the image directly using transformer-based vision-language modeling without any intermediate OCR and subsequent error drift.

- End-to-End Transformer: The model consists of a vision encoder and sequence decoder, where it functions similar to encoder-decoder transformer architecture used in machine translation.

Structured Output: Donut generates a structured text description such as key-value pairs, table contents, or JSON objects from an input image, hence very well suited for information extraction.

3.3.6.2 Why Donut is a Breakthrough

Eliminates OCR Limitations: The traditional OCR engines usually do not work well on non-standard fonts, handwriting, noisy scans, or low-resolution images. Donut avoids all these limitations by learning to directly infer textual content from visual representations.
Multiformat and Multilingual Compatibility: Absent reliance on character-level recognition, Donut is language and script tunable to those that do not perform well with OCR technology.
Single-Model Joint Layout and Content Modeling: Contrary to layout detection as well as text recognition being addressed in modular pipelines as separate tasks, Donut addresses the same jointly in one model.

3.3.6.3 Applications

Form Understanding: Information extraction from administrative forms in insurance, healthcare, and finance automated.

- Low-Quality Document Processing: Excellent performance on low-quality documents, faxes, or scanned handwritten text where OCR traditionally fails.
- Rapid Prototyping: As Donut is an end-to-end solution, it simplifies development pipelines and reduces dependency on multiple software pieces.

3.3.6.4 Limitations

- Early Development Stage: As a relatively new development, Donut is still evolving. Optimal performance often requires task-specific tuning.
- Interpretability Concerns: As Donut merges layout and text modeling into a single transformer, it could be harder to debug or interpret compared to a modular architecture where each step is standalone and inspect able.
- Training Resource Requirements: While inference is efficient, training Donut in an optimal manner still requires access to large-scale annotated document data sets and significant computational capacities.

3.3.7 Layout LM (v1, v2, v3)

Multimodal Transformers for Document Layout Understanding
Layout LM is a family of transformer models designed by Microsoft particularly for document comprehension through joint modeling of textual, visual, and positional data. Unlike general-purpose NLP models that handle regular text, Layout LM considers spatial layout information on text components—an important aspect of document semantics.

3.3.7.1 Architectural Overview

Textual Embedding: Layout LM inherits BERT and embodies the semantic context of each textual token.
Positional Embedding: Tokens have their 2D positions (bounding box) in the document layout space so that the model can learn positional structures (e.g., labels followed by values).
Visual Embedding (v2 and beyond): The model incorporates visual features from the raw image with convolutional backbones like ResNet, adding even more multimodal capability.

3.3.7.2 Real-World Applications

Form Parsing: Automatically detecting and labeling form fields and their related values (e.g., "Name: John Smith").
Key-Value Pair Extraction: Key-value pair extraction of structured data like invoice dates, payment amounts, or item descriptions.
Document Classification: Document classification of documents according to structure or content (resume vs. invoice).
Question Answering: Supporting tasks like DocVQA, where answers need to be implied from structured visual-textual data.

Model Evolution

- LayoutLMv1: Established multimodal input via mixing textual embedding’s with spatial positions in a paradigm-breaking leap ahead over text-only baselines.

- LayoutLMv2: Integrated visual features from doc images and augmented retraining tasks (e.g., masked visual-language modeling).

- LayoutLMv3: Integrated all three modalities (text, layout, image) into a unified framework, achieving state-of-the-art performance on benchmarking sets such as FUNSD, SROIE, and DocVQA.

3.3.7.3 Strengths

Layout Awareness: Captures document structure globally, required to comprehend complex layouts such as receipts, certificates, or academic papers.
Cross-Modal Integration: Strong performance in tasks that require simultaneous reasoning over visual and textual features.

3.3.7.4 Transfer Learning

LayoutLM models are retrained on large datasets of documents, which allows them to be easily adapted (fine-tuned) for specific tasks or domains.

Limitations

- **High Computational Cost**: Because the model uses both image and text features and has a complex structure, training and running it require a lot of computing power.
- **OCR Dependency**: Unlike models like Donut, LayoutLM depends on OCR to extract text and its position in the document before processing. This adds complexity and can introduce errors if the OCR is not accurate.
- **Difficult Fine-Tuning**: Adapting the model to a new task often requires a lot of manual work, including labeling data and trying different settings for layout-specific inputs.

Model	Architecture Type	Key Features	Strengths	Limitations
Faster R-CNN	Two-stage detector	RPN for region proposals, classification & bounding box regression	High detection accuracy; effective for locating objects in structured layouts	High computational cost; slower than one-stage models (e.g., YOLO)
Mask R-CNN	Two-stage detector + segmentation	Adds a mask prediction head for instance segmentation	Pixel-level segmentation; excels at distinguishing overlapping elements	Requires more annotations (segmentation masks); slower inference
Donut	Transformer (OCR-free	End-to-end image-to-text model; directly outputs structured sequences	No OCR dependency; robust on low-quality scans; suitable for multiple languages	Less interpretable; requires large datasets for fine-tuning
LayoutLM (v1-v3)	Transformer + Multimodal	Fuses text, layout (2D	Captures visual-textual	Depends on OCR for text +

	Fusion	position), and image features	relationships; SOTA results on form parsing, DocVQA, etc.	bounding boxes; computationally intensive

Table 3.3: *Comparison Table: Document Analysis Models*

3.4 Optical Character Recognition(OCR)

Optical Character Recognition, or OCR, is the technology to pull printed words out of the interior of an image—scanned paper, photographs, even handwriting—and get it into an editable, machine-readable form. A central part of document digitization and verification processes, OCR is a foundation upon which steps such as information extraction, semantic interpretation, indexing, and searching are built.

OCR is applied extensively for a range of purposes like finance (e.g., invoice processing), health (e.g., scanning patient records), law (e.g., court documents), and education (e.g., correction of exam papers). It is also an important application for automatic forms processing, print archiving, and assistive technology for visually impaired users.

The initial OCR systems were rule-based and relied extensively on hand-crafted features and traditional computer vision techniques. These tended to be a sequence of operations, including image preprocessing (e.g., thresholding, denoising), word- or character-level segmentation, and character recognition using template matching or basic machine learning classifiers.

These approaches were limited by image quality, fixed fonts, and rigid layouts. They did not perform well with noisy scans, mixed layout documents, and multilingual documents.

Deep learning revolutionized OCR. Today's OCR is now based on convolutional neural networks (CNN), recurrent neural networks (RNN), and more recent transformer models, which can deliver strong, high-quality recognition for distorted, complicated, or handwriting input.

These deep-learning techniques allow OCR engines to extract generalizable text and context visual abstractions on a big, domain-independent scale, and also languages. Moreover, the combination of OCR with layout analysis in end-to-end document understanding pipelines combined with natural language post-processing now allows you to extract, aside from raw text, semantic and structured content.

The following is a summary of how OCR technologies progressed from earlier pipeline frameworks to current deep learning frameworks, challenges, metrics, and real-world applications.

3.4.1 Traditional OCR Systems

Before the revolution of deep learning, OCR systems relied largely on rule-based and conventional machine learning approaches. Conventional OCR pipelines were generally multi-stage with image preprocessing, segmentation, feature extraction, and character recognition. Although very effective for structured and high-quality inputs, their accuracy would generally weaken when given noisy, degraded, or complex layouts.

3.4.1.1 Image Preprocessing

The first step of traditional OCR is to improve the input image so that it becomes less noisy and text will be more prominent. Techniques such as binarization (e.g., Otsu's method), morphological processing (e.g., dilation, erosion), skew correction, and denoising filters (e.g., Gaussian or median filter) are used. All these steps make the text more distinguishable from the background and reduce the likelihood of recognition failures.

3.4.1.2 Text Segmentation

Segmentation is the process of splitting the image into text lines, words, and ultimately characters. Rule-based heuristics, connected component analysis (CCA), and projection profiles are common techniques. It is strongly affected by layout variation; overlapping characters or inconsistent spacing can significantly reduce accuracy.

3.4.1.3 Feature Extraction

Once characters are separated, features are extracted to describe their shape and structure. Traditional OCR systems traditionally use hand-designed features such as edge detection (Sobel, Canny), zoning (dividing character space into zones), and geometric descriptors (e.g., aspect ratio, stroke width).

3.4.1.4 Character Recognition

In this step, the system classifies characters using simple classifiers like:

- Template Matching: Comparing segmented character images to a library of known characters.
- K-Nearest Neighbors (KNN) or Support Vector Machines (SVM): Used on extracted features for classification.
- Hidden Markov Models (HMMs): Sometimes employed for recognizing sequences of characters, particularly in printed or cursive scripts.

3.4.1.5 Post-processing

Spell-checkers, dictionaries, and domain-specific correction rules are applied to refine the output, especially for noisy documents or poor-quality scans. This step attempts to correct recognition errors that occur due to visually similar characters (e.g., 'O' vs. '0', 'l' vs. '1').

3.4.1.6 Limitations of Traditional OCR

Despite being useful, traditional OCR systems suffer from the following limitations:

- Not so excellent a generalization to new fonts, languages, or layouts.
- Prone to image noise in low-resolution or deteriorated images.
- Poor performance with handwritten or cursive scripts.
- Hand-crafted rules and features are required, thus decreased adaptability.

These limitations provided opportunities for application of data-learned deep learning-based OCR.

3.4.2 Deep Learning-based OCR

With the advent of deep learning, OCR systems have dramatically improved in both accuracy and robustness, especially in handling complex layouts, low-quality scans, and cursive or handwritten text. Deep learning-based OCR eliminates the need for handcrafted features and instead learns powerful visual representations directly from data through convolutional and recurrent neural networks.

3.4.2.1 Convolutional Neural Networks (CNNs) for Feature Extraction

Deep learning OCR systems start with CNNs, which automatically extract hierarchical visual features from input images. CNNs such as VGG, ResNet, or EfficientNet form the backbone of most modern OCR architectures, capturing patterns in fonts, textures, and structures.

These networks handle:

- Variability in fonts and styles
- Noisy or low-contrast backgrounds
- Complex document layouts (e.g., overlapping or rotated text)

3.4.2.2 Sequence Modeling with RNNs and LSTMs

Text in scanned documents is sequential. To model this, OCR systems employ Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) units. These networks capture dependencies between characters and words over time, improving recognition accuracy for entire text lines.

Combined models, often referred to as CRNNs (Convolutional Recurrent Neural Networks), leverage CNNs for spatial feature extraction and RNNs for sequence decoding.

3.4.2.3 Connectionist Temporal Classification (CTC)

Many OCR systems use CTC loss for training. This allows the model to predict variable-length sequences (such as words) without requiring character-level alignment between input images and output sequences.

Advantages of CTC :

- No need for detailed annotation (only sequence labels are needed)
- Robust to input distortions and irregular spacing

3.4.2.4 Attention-based Encoder-Decoder Models

Inspired by machine translation, attention mechanisms have been integrated into OCR pipelines. These models consist of an encoder (CNN or CNN-RNN) and a decoder (LSTM or Transformer) that generates character sequences, guided by attention over relevant image regions.

Notable models:

- Show, Attend and Read (SAR)
- ASTER
- TRBA (TPS-ResNet-BiLSTM-Attention)

These models outperform traditional CRNNs on irregular or curved text.

3.4.2.5 Transformer-based OCR Models

Transformers have recently become dominant in OCR due to their powerful sequence modeling capabilities and global attention mechanism.

3.4.2.6 State-of-the-art models include:

- TrOCR (Transformer for OCR): developed by Microsoft, pre-trained on large corpora, and fine-tuned on OCR tasks. It supports printed and handwritten text.
- Donut (Document Understanding Transformer): a fully OCR-free model for structured document understanding. Although Donut bypasses OCR in the traditional sense, its architecture performs OCR implicitly as part of document comprehension.
- LayoutLMv3: integrates visual, textual, and layout information in a unified Transformer framework for end-to-end document processing.

3.4.2.7 Advantages of Deep Learning-based OCR

- High Accuracy on diverse fonts, styles, and noisy inputs
- End-to-End Learning, minimizing need for manual preprocessing
- Adaptability to various scripts and languages through transfer learning
- Handwriting Recognition and irregular layout handling

3.4.2.8 Limitations

- Requires large annotated datasets for training
- High computational demands (during training and inference)
- Might still struggle with extremely poor quality scans or artistic fonts

3.4.3 Pre-trained OCR Systems and Frameworks

In recent years, various powerful pre-trained OCR systems and open-source frameworks have emerged, enabling researchers and developers to implement high-quality OCR solutions with minimal training. These systems combine advanced deep learning techniques with robust preprocessing and post processing pipelines.

3.4.3.1 Tesseract OCR (with LSTM)

- Overview: Originally developed by HP and now maintained by Google, Tesseract is one of the most widely-used open-source OCR engines.
- Deep Learning Upgrade: Since version 4, Tesseract integrates LSTM-based neural networks, dramatically improving accuracy for printed text.

- Languages Supported: 100+ languages.

Strengths:

- Good for multi lingual printed documents
- Supports custom training

Limitations:

- Struggles with layout understanding
- Less effective for handwritten or artistic text

3.4.3.2 EasyOCR

Overview: A lightweight, Python-based OCR library that supports over 80 languages.

Architecture: Based on CRNN and CTC loss.

Strengths:

- Easy to use
- Good performance for printed and handwritten text

Use Case: Frequently used for mobile and embedded applications.

3.4.3.3 PaddleOCR

Overview: Developed by Baidu, PaddleOCR is a highly optimized and modular OCR framework that supports over 80 languages and multiple model types.

Features:

- Includes detection + recognition modules
- Supports layout analysis and table recognition
- Offers Lightweight and Server-grade models

Advanced Models: Includes support for Transformer-based architectures like PP-OCRv3 and LayoutXLM.

3.4.3.4 TrOCR

Overview: A Transformer-based OCR model developed by Microsoft and available through the Hugging Face Transformers library.

Architecture:

- Vision Transformer (ViT) as encoder
- Text Transformer decoder

Capabilities:

- Handles printed and handwritten text
- Pre-trained and fine-tuned on large-scale OCR datasets

Performance: Achieves state-of-the-art accuracy on benchmarks such as IAM and SROIE.

3.4.3.5 Donut (Document Understanding Transformer)

Key Idea: An OCR-free model designed for end-to-end document parsing.

Functionality: Learns to generate JSON-structured outputs directly from document images.

Strengths:

- No intermediate OCR step
- Excellent for structured data extraction

Use Case: Invoices, forms, ID cards

3.4.3.6 Layout LM Series

LayoutLMv1, v2, v3: These models fuse visual, textual, and positional information using a multimodal Transformer architecture.

Input: Takes in images, OCR text, and layout coordinates.

Strengths:

- Ideal for document classification, QA, and information extraction
- Can capture document structure effectivel

Use Cases:

- Form understanding (FUNSD)
- Invoice parsing (SROIE)
- Document classification (RVL-CDIP)

Framework	Type	Deep Learning	Handwriting	Layout-Aware	Language Support	Strengths
Tesseract	Open-source OCR	LSTM	Limited	No	100+	Easy to use, widely supported
EasyOCR	Python library	CRNN + CTC	Yes	No	80+	Lightweight, good accuracy

PaddleOCR	Full pipeline	CNN + Transformer	Yes	Yes	80+	Modular, supports layout/table
TrOCR	Transformer-based	Yes	Yes	No	Pretrained	SOTA performance
Donut	OCR-free model	Vision Encoder	Yes	Yes	JSON output	Best for structured document tasks
LayoutLMv3	Multimodal	Yes	Yes	Yes	Pretrained	Best for document understanding

Table 3.4: *Summary Table of Key OCR Frameworks*

3.5 Post-processing Techniques in OCR Pipelines

Even the most advanced OCR models may produce noisy results from complex layouts, scanning noise, or handwriting difference. Post-processing is therefore an important step to enhance the quality, consistency, and usability of extracted text. These techniques typically follow layout detection and OCR steps in an attempt to correct errors, structure information, and extract key data points.

3.5.1 Text Correction and Normalization

Spell Checking and Correction: Integration of language models (e.g., BERT, GPT, or n-gram based) helps fix OCR errors such as incorrect characters, missing punctuation, or spacing mistakes.

Noise Reduction: Techniques such as Levenshtein distance or edit distance algorithms are used to clean malformed words and non-ASCII artifacts.

3.5.2 Named Entity Recognition (NER)

Goal: Extract structured information such as names, dates, organizations, and monetary values from unstructured OCR text.

Tools:

- SpaCy and Hugging Face NER models
- Custom-trained BERT/Transformer-based NER pipelines for specific domains (e.g., finance, legal)

Example: From a scanned invoice, NER can extract vendor names, invoice numbers, and payment terms.

3.5.3 Table Structure Recognition and Parsing

Structure recovery from OCR output is crucial for documents like invoices or scientific papers.

Techniques include:

- Rule-based cell alignment
- Deep learning models (e.g., TableNet, CascadeTabNet, and Graph Neural Networks)
- OCR + Vision integration to reconstruct rows/columns and headers

3.5.3.1 Relationship Extraction

Some documents (e.g., forms or medical records) require linking related pieces of text.

Post-processing can apply:

- Dependency parsing
- Relation classification models
- Semantic similarity measures to pair fields (e.g., linking a patient's name with ID number)

3.5.3.2 Output Structuring and Export

Final step is to export the cleaned and organized data in usable formats such as:

- JSON (structured entities)
- XML (hierarchical layout)

CSV/XLS (for table-heavy documents)

3.6 Conclusion

In short, layout-aware information extraction from scanned documents is a high-level task reliant on several interdependent factors. Among them, layout analysis, OCR, and post-processing are essential elements of the pipeline for document understanding.

Accurate layout analysis enables the system to understand the spatial and logical structure of a document, improving identification and clustering of content blocks. Robust optical character recognition (OCR) translates visual data into readable text by machines, giving the basis for any future processing. Finally, intelligent post-processing enables refinement and structuring of the data that has been extracted, usually based on NLP and machine learning methodology to detect errors and improve semantic understanding.

The combination of these components forms the basis of current end-to-end scanned document processing systems. In the next chapter, we will be detailing OCR technologies and approaches, which form the next crucial step in our document analysis pipeline.

4 Chapter 3: Proposed Methodology and Experimental Results.

4.1 Introduction

In this chapter, we introduce in depth the pipeline that will be used to extract structured data from scanned document (Resume). The system integrates layout analyses and optical character recognition (OCR) techniques to automatically detect key areas within the documents and accurately extract text content from them. The ultimate goal is to transform unstructured document images into machine-readable, structured data and process or store them in databases further.

4.2 Proposed Methodology Architecture

The developed pipeline for the document information extraction (resume) is composed of several interconnected stages, each responsible for a critical task within the end-to-end process. The architecture is designed to be modular, scalable, and adaptable to different document layouts and formats. Figure 0.1: Proposed Methodology illustrates a high-level overview of the proposed method.

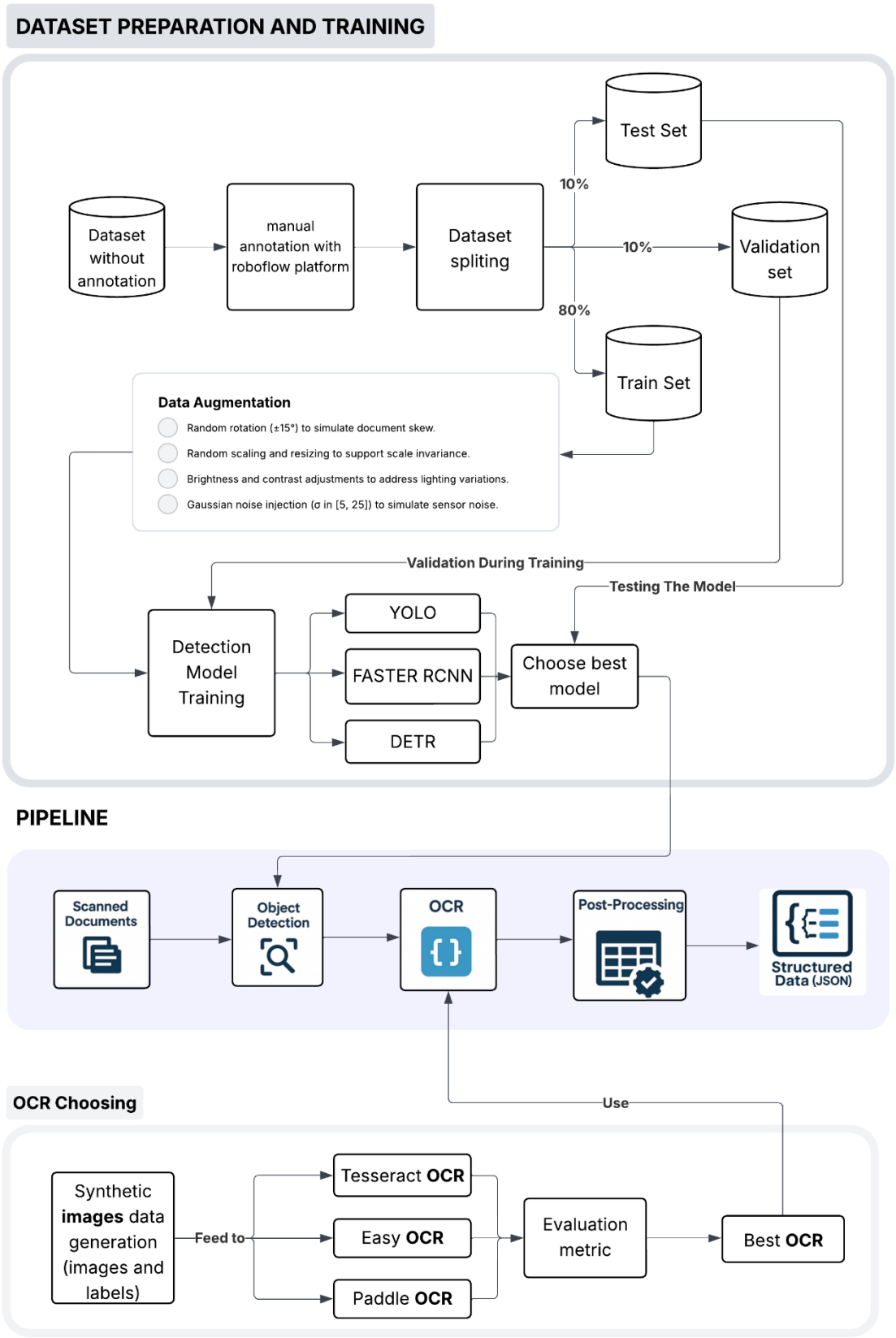


Figure 0.1: *Proposed Methodology*

4.2.1 Dataset Collection and Annotation

A critical preliminary step in the development of the information extraction pipeline was the collection and annotation of a custom dataset personalized for object detection in scanned resume documents. Given the absence of publicly available datasets with bounding box annotations for resume components, a custom dataset was collected and carefully labeled.

A total of approximately 2,300 resume images were gathered from a variety of online sources, encompassing diverse formats, templates, and visual layouts. These images were then manually annotated using the **Roboflow** platform, with particular attention given to labeling semantically meaningful fields relevant to resume parsing.

The annotation schema comprised 13 distinct classes, defined as follows:

Class ID	Class Name	Description
0	Extracurricular	Consolidated class representing Achievements and Community involvement, due to low individual sample counts
1	Certifications	Certificates, licenses, or professional accreditations
2	Contact	Email, phone number, address, and other contact information
3	Education	Academic qualifications and institutions attended
4	Experience	Work history, job titles, responsibilities
5	Interests	Hobbies and personal interests
6	Languages	Spoken and written language proficiencies
7	Name	Candidate’s full name
8	Profile	Summary or personal statement
9	Projects	Descriptions of academic or personal projects
10	Image	Profile photograph or headshot, if present
11	Resume	Entire resume container or page layout
12	Skills	Technical and soft skills listings

Table 0.1: *Classes Description*

Two additional classes **Achievements** and **Community** were originally defined as separate classes, but due to extreme class imbalance, they were merged into a single class named **"Extracurricular"** for the purposes of annotation consistency and successful model training.

4.2.2 Resume Fields Detection (Object Detection Stage)

After the dataset preparation phase, the second key component of the proposed methodology is to identify semantically significant fields in CV documents using object detection mechanisms. A process often known as layout analysis. This aims to automatically find and detect key information fields, such as "name," "experience," "skills," and "education," by predicting their bounding boxes.

To achieve this, we trained and compared three state-of-the-art object detection models YOLOv11 (Ultralytics, 2023), Faster R-CNN (Ren, Shaoqing, He, Kaiming, Girshick, Ross, & Sun, Jian, 2015), and DETR (Carion, Nicolas, et al., 2020) on the prepared dataset. All models were compared based on standard object detection evaluation metrics, and the best-performing model was selected for inclusion in the final pipeline.

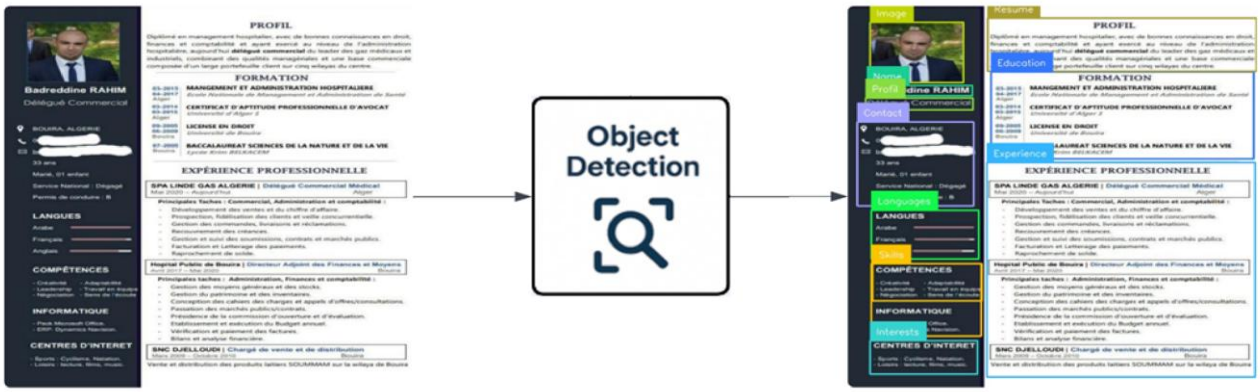


Figure 0.2: *Layout Analysis Stage*

4.2.3 Text Extraction (OCR Stage)

The second key component of the proposed methodology is text extraction from the previously detected resume fields. This stage involves applying Optical Character Recognition (OCR) to each cropped field region to retrieve its textual content.

Multiple OCR engines were evaluated for this task, including open-source and deep learning-based solutions such as **TrOCR** (Li, Minghao, Yin, Fei, Zhang, Cheng, & Liu, Cheng-Lin, 2021), **EasyOCR** (JaidedAI, 2021), and **PaddleOCR** (Baidu, 2022). Despite the promising accuracy of transformer-based approaches like TrOCR, challenges with generalization on real-world resume formats and high computational cost made it less suitable for deployment.

Ultimately, **Tesseract OCR** was selected due to its robustness, flexibility, and efficiency in structured document parsing. Its support for multilingual text, ease of integration, and favorable performance in our study.

The OCR stage plays a critical role in transforming layout-structured visual data into machine-readable text, enabling downstream information extraction and semantic structuring in later stages of the pipeline.

4.2.4 Post-Processing and Information Structuring

Following text extraction, a post-processing stage was introduced to improve the reliability and usability of the detected text. This component plays a key role in transforming raw OCR outputs into a structured, machine-readable format suitable for downstream applications such as applicant filtering or resume analytics.

The post-processing module is composed of the following high-level stages:

- **Text Cleaning and Normalization:** Removes common OCR errors, standardizes text case and spacing, and prepares the text for semantic parsing.
- **Field-Specific Parsing:** Extracts domain-relevant entities (such as dates, emails, phone numbers, institutions, and skill sets) using rule-based and pattern-matching techniques.
- **Validation and Error Correction:** Applies logical and pattern-based validation to ensure consistency and correctness across fields.
- **Structured Output Generation:** Assembles the cleaned and validated data into JSON and CSV formats.

This step bridges the gap between raw OCR output and meaningful resume understanding, completing the transformation pipeline from document to structured data.

4.2.5 Summary of the Pipeline

The architecture outlined is a modular, multi-step pipeline for the conversion of unstructured resume images into structured, machine-readable data. Each step within the pipeline is carefully designed to address one specific problem, and as such, it is an extensible and adaptable document understanding system. The major stages of the methodology are listed below:

1. Dataset preparation and annotation
2. Resume Field Detection (Layout Analysis)
3. Text Extraction (OCR Stage)
4. Post Processing (Structured Data)

4.3 Experiments and Results

This section describes the experimental setup followed during training and evaluation of the proposed resume information extraction pipeline in detail. We provide the data preparation, model configurations, learning procedure, evaluation metric, and hardware and software infrastructure. Each step has been carefully designed to provide the experiments with the desired reproducibility, scalability.

4.3.1 Dataset Preparation

The experiment is based on a dataset of approximately **2,300** high-resolution resume images, each annotated with a bounding box representing a semantic field commonly used in professional resumes. The dataset is designed to enable the extraction of tagged information

from semi-structured or unstructured documents. Its use cases are critical in resume parsing, recruitment automation, and **HR** analytics.

4.3.1.1 Image Specifications

- **Format**: Most images are in **JPG** format, with some in **PNG**. JPG is preferred due to its universal compatibility and smaller file size, despite minor compression artifacts.
- **Average Resolution**: Approximately **693 × 813 pixels**.
- **Source**: Roboflow Platform.

4.3.1.2 Preprocessing

Preprocessing of all the data was done prior to training the models to normalize the data and to strengthen the machine-learning pipeline as well. The below steps were performed:

1. **Normalization of pixels:** The pixel intensity values were all normalized to the range [0, 1] to stabilize and accelerate the learning process during the model's training.
2. **Bounding Box Format Conversion:** The formats of the annotations were converted into the appropriate bounding box formats that are supported by different object detection models. YOLO and COCO formats were designed to be compatible with different architectures.
3. **Duplication and Cleaning of the dataset:** Duplicate images were found and were removed. There was rigid separation between the training set, validation set, and testing set to avoid data leakage and obtain good evaluation.

4.3.1.3 Annotation Process

Manual annotations were performed manually using **Roboflow** platform, which is cloud-based and accepts various formats for object detection including YOLO, COCO. A predefined labeling schema was used to ensure semantic consistency.

- **Quality Assurance**:
  - Random sampling and visual inspection
  - Verification of label accuracy and bounding box alignment
- **Exported Formats**:
  - **YOLO Format**: For models such as YOLOv5 and YOLOv11
  - **COCO Format**: For models like Faster R-CNN and DETR

This multi-format compatibility supports flexible experimentation with different detection architectures.

4.3.1.4 Class Distribution

The class distribution of annotated instances is summarized in the table below Table 0.2: **Classes Distribution**:

Field Category	Count	Percentage
Contact	6,066	12.97%
Skills	5,383	11.51%
Name	5,343	11.42%
Experience	5,217	11.15%
Education	5,117	10.94%
Profile Summary	4,396	9.40%
Resume Title/Header	3,930	8.40%
Image	3,754	8.03%
Languages	2,448	5.23%
Interests	2,170	4.64%
Extracurricular	1,138	2.43%
Certifications	1,068	2.28%
Projects	739	1.58%

Table 0.2: *Classes Distribution*

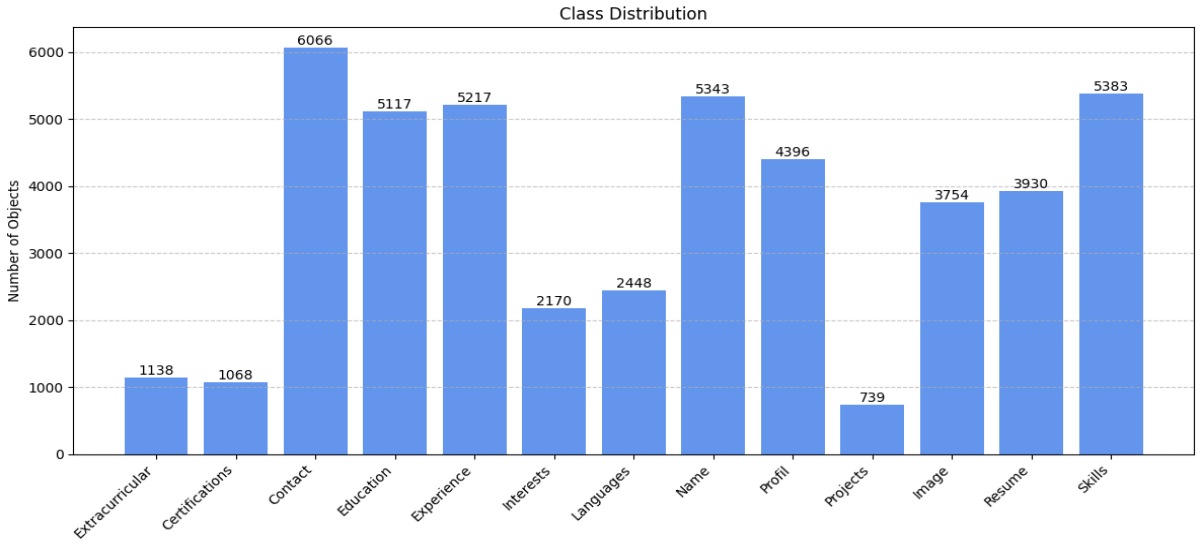


Figure 0.3: *Class Distribution Across Annotated Categories*

This distribution indicates a moderate imbalance, with fields such as *Contact*, *Skills*, and *Experience* being more frequent than others. Class imbalance was taken into consideration during model training and evaluation to ensure equitable performance across categories.

4.3.1.5 Dataset Splitting

To maintain representativeness and balance across semantic classes, the dataset was partitioned into three mutually exclusive subsets using a stratified sampling strategy:

- **Training Set**: 70% (~1,610 images)
- **Validation Set**: 15% (~345 images)
- **Test Set**: 15% (~345 images)

4.3.1.6 Data Augmentation

In order to improve the model's robustness and generalization capabilities under various real-world conditions. The augmentation techniques employed include:

- Random rotation (±15°) to simulate document skew.
- Random scaling and resizing to support scale invariance.
- Horizontal flipping to encourage spatial generalization.
- Brightness and contrast adjustments to address lighting variations.
- Gaussian noise injection (σ in [5, 25]) to simulate sensor noise.

Each transformation was applied with controlled probabilities to avoid distributional drift.

4.3.2 Models Training and Result (Detection Models)

This section outlines the configurations applied to train the object detection models in this study. The aim is to provide transparency regarding the hyperparameters, training strategies, and loss functions used, ensuring that readers can fully understand and potentially replicate the experiments.

4.3.2.1 YOLOv11

The training of the object detection model in this study is based on **YOLOv11-M**, a hardware-optimized variant (Ultralytics, 2023) known for its balance between detection accuracy and inference efficiency. The model was trained using its default architecture, with key hyperparameters fine-tuned according to the specific characteristics of the dataset and available computational resources.

Training Configurations:

- **Model Architecture**: YOLOv11-M This design seeks a balance between accuracy and efficiency in order to be suitable for both real-time use and for use in which high detection performance is desired.
- **Input Image Size**: 640 × 640 pixels This input size is a general choice for YOLO models in balancing detection performance against computational load. Larger images are generally going to have superior detection performance but come at increased computational expense, while smaller images can lead to faster training but lose detection performance for smaller objects.
- **Batch Size**: 16
  Batch size describes the quantity of images being processed simultaneously per training iteration. We choose a batch size of 16 based upon a trade-off between convergence and memory. When using a batch size that is too big, memory issues arise on GPUs, and with a batch size that is too small, convergence slows.
- **Number of Epochs**: 60
  This is an instruction that has the model train for 60 times over the entire training set. The number of epochs is often based on how rapidly the model converges and 100 epochs are chosen so that it is sufficient for obtaining a satisfactory performance without overfitting, especially with optimally trained models such as YOLOv11.
- **Learning Rate Schedule**: Cosine Annealing

Cosine annealing is a learning rate schedule that anneals learning rate gradually according to a cosine function. It slows down learning by starting with a bigger learning rate and slowly graduating it, avoiding any sudden learning drops which tend to break learning process of model.

- **Optimizer:** AdamW
  AdamW (Adam with Weight Decay) is an improved variant of the Adam optimizer that decouples weight decay from the gradient update, enabling better generalization and regularization. It combines the benefits of adaptive learning rates and L2 regularization, making it particularly effective for training deep networks. AdamW helps stabilize training and often converges faster and more reliably than traditional Adam or SGD, especially in complex tasks like object detection.
  Loss Function: YOLOv11 uses a composite loss function that includes multiple components:
  - **Bounding Box Regression Loss:** Measures how accurate predicted bounding boxes are against true values.
  - **Objectness Loss:** Quantifies a model's confidence level about detection of an object within a specific bounding box.
  - **Classification Loss:** Measures how well a predicted object class approximates the true class.

These loss terms are designed so that it encourages all aspects of object detection, with a trade-off being made for class accuracy vs. localization (bounding box).

This deployment employs the empirically set default parameter values for YOLOv11 architecture for general object detection tasks. This gives robust performance with potential for tuning and adaptation based on dataset size and hardware.

4.3.2.2 Faster R-CNN (Two-Stage Detector Configurations):

Faster R-CNN (Ren, Shaoqing, He, Kaiming, Girshick, Ross, & Sun, Jian, 2015) were also used for a contrast of YOLOv11 with more conventional two-stage detectors. They were chosen because they perform well for tasks with accurate localization and object class identification, for instance, structured document comprehension.

- **Backbone Network:** Pre-trained ResNet-50 (He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian, 2016) witch is a deep convnet renowned for residual connections that combat the vanishing gradient challenge at training time. Initiating with a pre-trained ResNet-50 provides a solid image feature extraction backbone, promoting convergence and performance, especially if training from scratch is computationally impractical.
- **Input Image size:** 600 × 600 pixels
  The input size is lower compared to YOLOv11 for a balance of performance and computational expense. 600x600 is a reasonable size for object detection tasks, particularly for smaller objects or when faster training iterations are needed.
- **Batch Size:** 6
  A batch size of 6 is used so that memory overflow and stable training are avoided.
- **Adam** (Loshchilov, Ilya & Hutter, Frank, 2019) **and SGD Optimizer**
- **Learning Rate:** 0.0001

The smaller learning rate is chosen so a gradual adjustment is made to the model's weights, particularly for use with pre-trained networks, which are already optimized. Fine tuning with a low learning rate avoids interference with pre-trained weights.

- **Number of Epochs:** 50
  Faster R-CNN would converge sooner due to more structured training pipeline, and therefore 50 epochs were sufficient for good performance without overfitting.
- **Loss Function:**
  Faster R-CNN applies a multi-task loss function, and components of said loss function are typically:
  - **Classification Loss (Cross-entropy loss):**
    It measures precisely how accurately the model is classifying any given object into the correct class. Cross-entropy is being used here because the model is classifying into all of the possible regions.
  - **Bounding Box Regression Loss (Smooth L1 loss):**
    It makes a prediction whether or not the predicted bounding boxes are well aligned with true bounding boxes. It is referred to as Smooth L1 loss (a variation of the L1 loss) and is less sensitive to outliers compared to normal L2 loss. Small misalignments are penalized less than large ones, avoiding overly big gradients.
  - **Region Proposal Network (RPN) Objectness Loss:**
    This is a quality measure for RPN's region proposals. It differentiates object (foreground) and backgrounds. It is often a binary class loss (like cross-entropy) for whether a proposal is an object or not.
- **Early Stopping:** Training stopped if validation loss does not improve for 5 consecutive epochs. This avoids training for too many epochs and helps keep model parameters generalizable.

4.3.2.3 DETR Configuration

DETR (Carion, Nicolas, et al., 2020) is a highly advanced object detection model based on a Transformer architecture (Carion et al., 2020) that, unlike traditional anchor-based region proposals, predicts object classes and locations directly. This results in a more generalizable and strong detection mechanism, especially suited for hard scenes.

- **Architecture:** Transformer-based Detection Model (Vaswani, Ashish, et al., 2017) uses a Transformer encoder-decoder architecture, with global context modeling capability and therefore suited for object detection within densely or visually cluttered scenes.
- **Backbone Network:** ResNet-50 (He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian, 2016) with Feature Pyramid Network (FPN) (Lin, Tsung-Yi, et al., 2017)
  Backbone network ResNet-50 produces hierarchical feature maps of an input image, which are enhanced by FPN with multi-scale detection features, for excellent performance over objects with varied sizes.
- **Image size input:** 600 × 600 pixels
  This size provides a good balance between computational efficiency and detection accuracy, providing sufficient resolution to detect both small and large objects.
- **Batch Size:** 16
  Since the Transformer model is memory-demanding, a smaller batch size was utilized for enabling successful training within GPU memory constraints.

- **Optimizer:** AdamW (Loshchilov, Ilya & Hutter, Frank, 2019) witch is specifically suited for Transformer models and produces better weight regularization, which is particularly crucial for convergence with large models.
- **Learning Rate Schedule**: 0.0001 Cosine Annealing
- **Number of Epochs:** 50
- **Loss Function:** Composite Loss
  DETR uses a combination of losses that are optimized jointly for object classification and localization:
  a. **Classification Loss (Cross-Entropy Loss):** Responsible for classifying objects recognized into respective classes.
  b. **Bounding Box Regression Loss (L1 Loss):** The metric measures predicted vs. ground truth bounding box difference.
  c. **Generalized IoU Loss (GIoU Loss):** Provides better bounding box localization through consideration of predicted box and ground truth box overlap.
  d. **No Object Loss:** Penalizes inaccurate forecasts for background regions, maintaining high precision for the model.

4.3.3 Detection Results

4.3.3.1 Evaluation Metrics

In order for a strong object detection performance assessment, multiple critical metrics were utilized:

- **Precision:** The ratio of correctly predicted positive observations to the total predicted positives. It is calculated as:

$$\boldsymbol{Precision = \frac{TP}{TP + FP}}$$

  Where **TP** is the number of true positives, and **FP** is the number of false positives. High precision indicates a low false-positive rate.

- **Recall:** The ratio of correctly predicted positive observations to all actual positive observations. It is calculated as:

$$\boldsymbol{Recall = \frac{TP}{TP + FN}}$$

  Where **FN** is the number of false negatives. High recall indicates a low false-negative rate.

- **Mean Average Precision at 0.5 (mAP@0.5):** The mean of all the values of precision calculated at Intersection over Union (IoU) = 0.5 for all categories. This is a standard measure for object detection models, checking whether a model is excellent at classifying and is also great at object localization.
- **Mean Average Precision at 0.5:0.95 (mAP@0.5:0.95):** An extension of mAP that calculates the average precision at multiple IoU thresholds ranging from 0.5 to 0.95

(in increments of 0.05). This provides a more comprehensive evaluation of model performance.

- **F1-Score:** The harmonic mean of precision and recall, providing a single metric to evaluate model performance. It is defined as:

$$\mathbf{F1} = 2 \times \frac{\boldsymbol{Precision} \times \boldsymbol{Recall}}{\boldsymbol{Precision} + \boldsymbol{Recall}}$$

- **Confusion Matrix:** Breakdown of true positives, false positives, true negatives, and false negatives by class, enabling extensive error analysis.
- **Inference Time:** The average time taken for the model to process an image and produce predictions. This is critical for real-time deployment scenarios.

4.3.3.2 YOLOv11 Performance

The YOLOv11 model demonstrated strong performance on the resume dataset, achieving high detection accuracy across most semantic categories. The Losses Curve are in Figure 0.4: **YOLO Losses** and The overall performance metrics on the test set are summarized in the Table 0.3

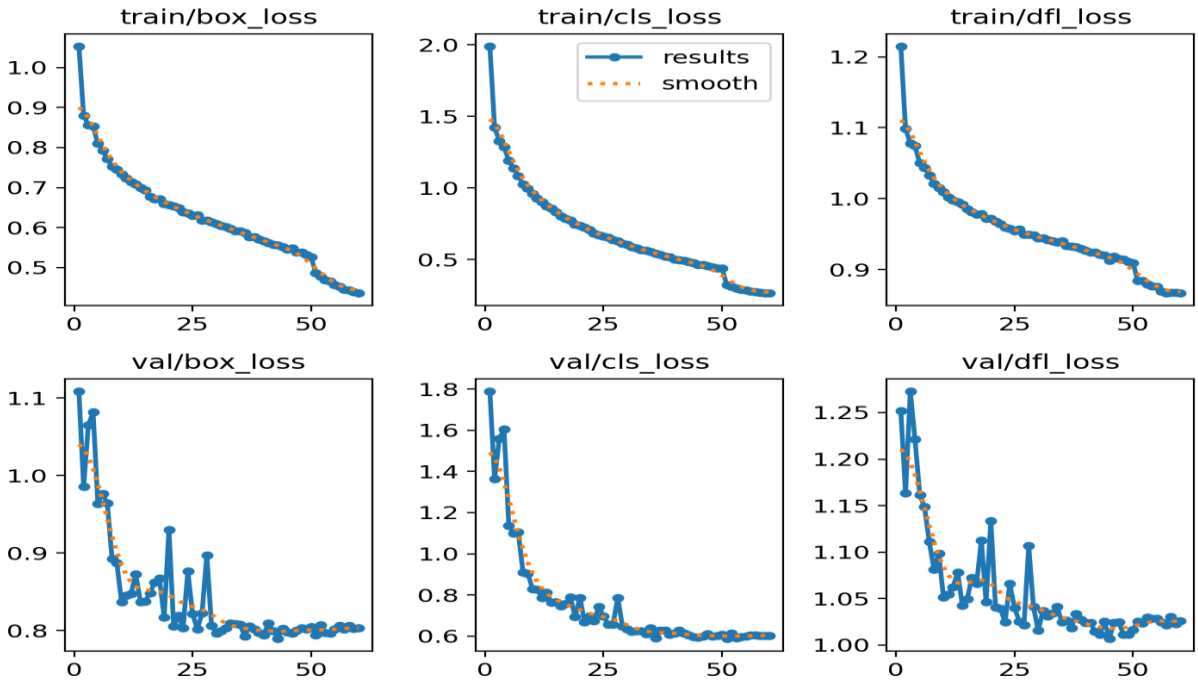


Figure 0.4: *YOLO Losses*

Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95

Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Extracurricular	0.817	0.740	0.815	0.736
Certifications	0.813	0.807	0.875	0.763
Contact	0.927	0.907	0.952	0.803
Education	0.945	0.932	0.974	0.901
Experience	0.938	0.944	0.978	0.930
Interests	0.931	0.894	0.955	0.834
Languages	0.945	0.905	0.959	0.817
Name	0.951	0.943	0.979	0.759
Profile	0.895	0.855	0.908	0.668
Projects	0.846	0.904	0.912	0.851
Image	0.954	0.994	0.979	0.865
Resume	0.938	0.938	0.965	0.851
Skills	0.924	0.924	0.965	0.873
Overall	**0.909**	**0.899**	**0.940**	**0.819**

Table 0.3: *YOLOv11 Performance Metrics by Class*

The model obtained an overall mAP@0.5 score of 0.94 and an mAP@0.5:0.95 score of 0.819, which is

Precision-recall analysis

The YOLOv11's precision-recall features are further demonstrated using the given curves:

- **Precision-Recall Curve:** The figure illustrates the precision-recall trade-off for different confidence cutoff points.
- **F1-Score Curve:** Graphs the harmonic mean of recall and precision, with the peak being the best point for the model's operation.
- **ROC Curve (Receiver Operating Characteristic):** Illustrates a model's ability to distinguish between classes.
- **Confusion matrix:** Indicates classification accuracy and misclassifications for every class, revealing any existing biases.

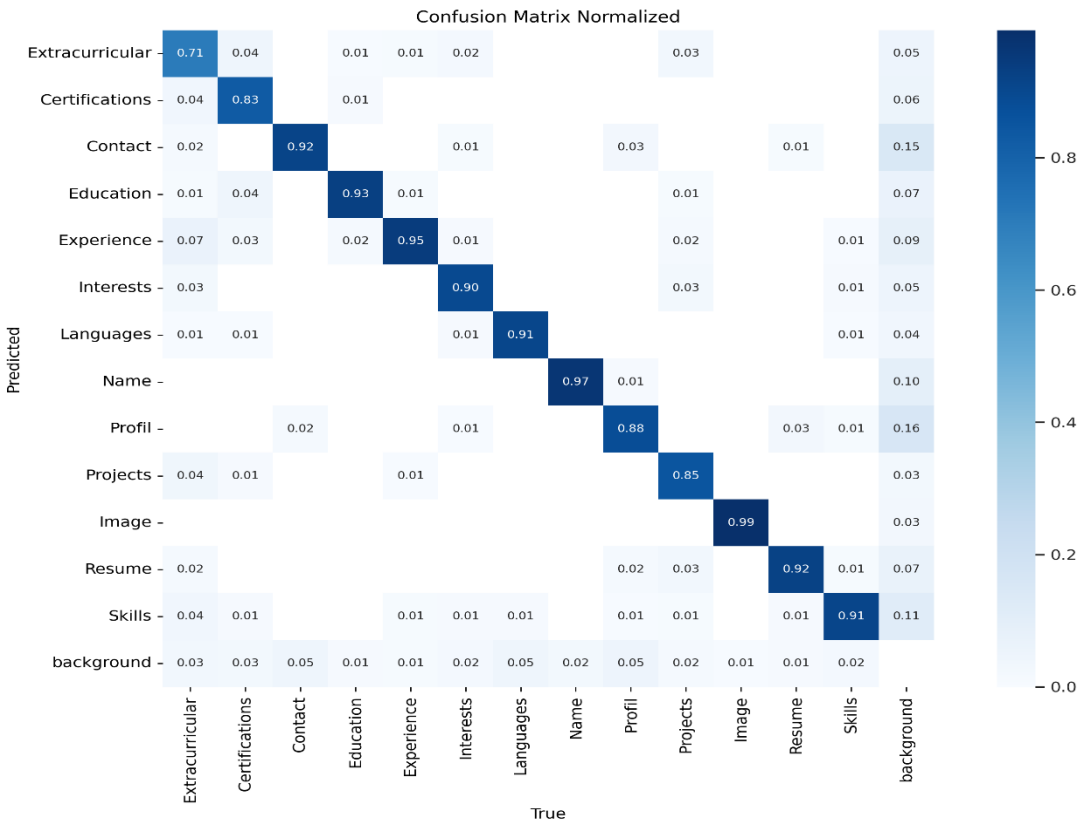


Figure 0.5: *Confusion Matrix (YOLO)*

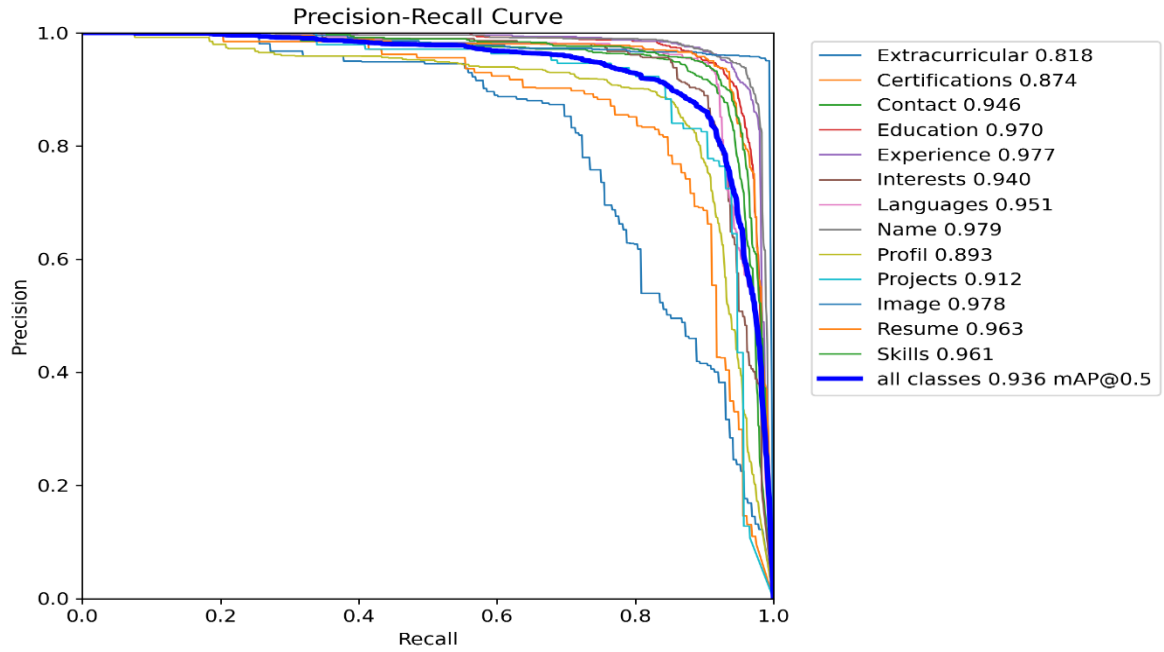


Figure 0.6: *Precision-Recall curve (YOLO)*

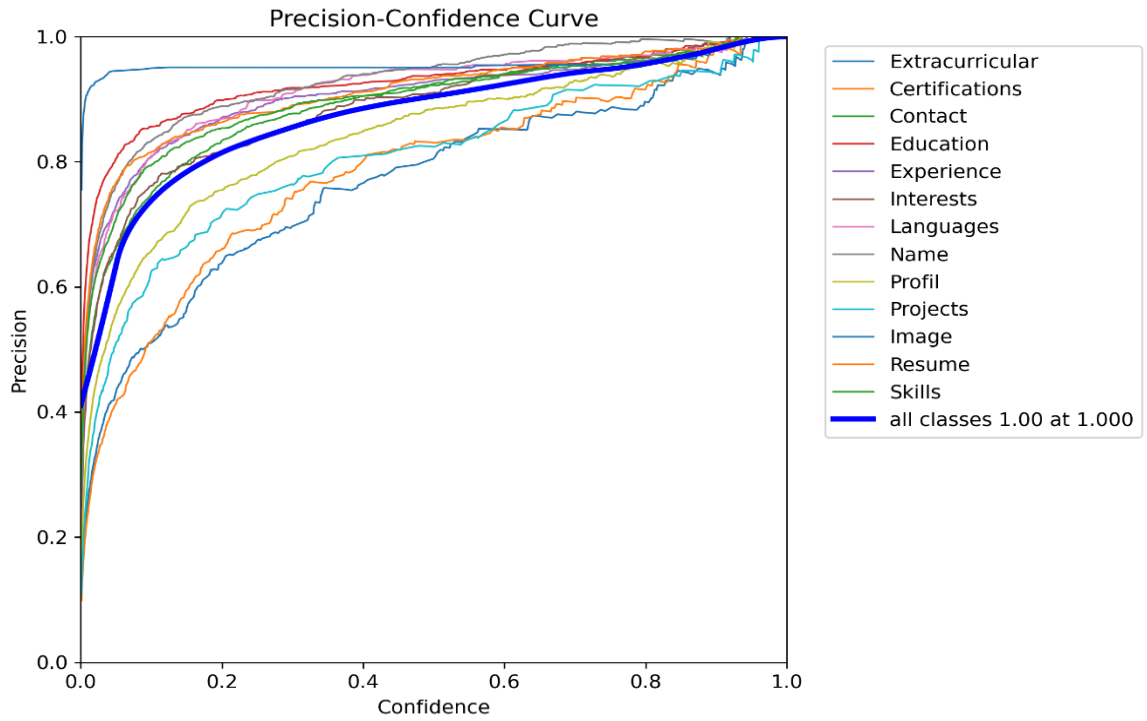


Figure 0.7: *Precision-Confidence Curve (YOLO)*

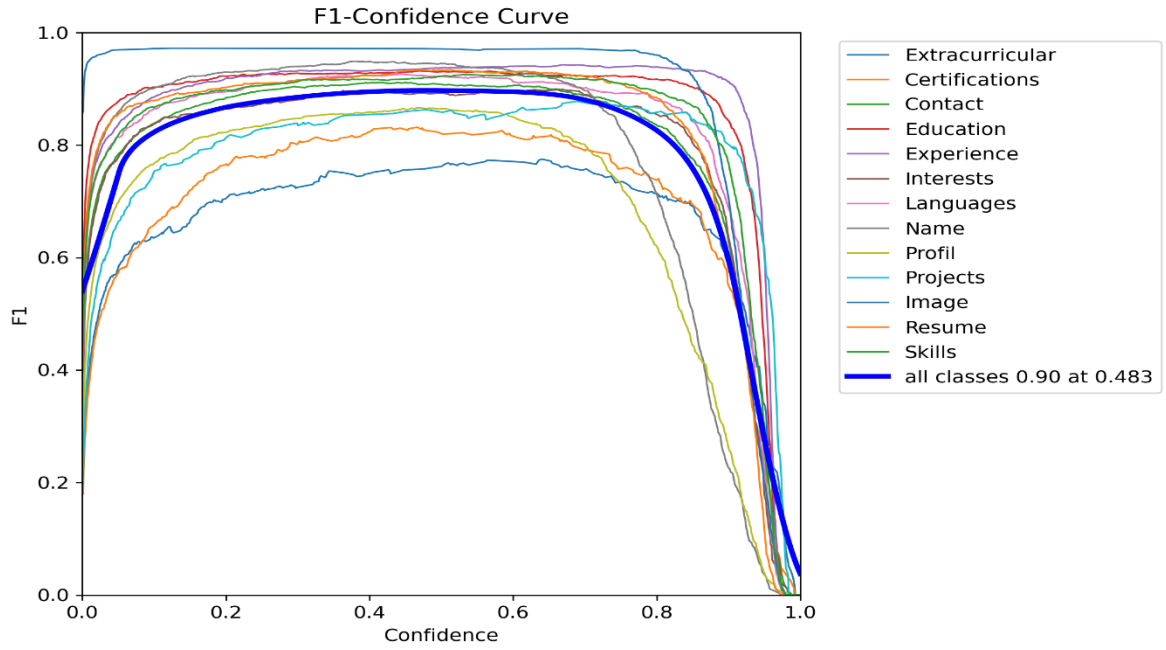


Figure 0.8: *F1-Confidence Curve (YOLO)*

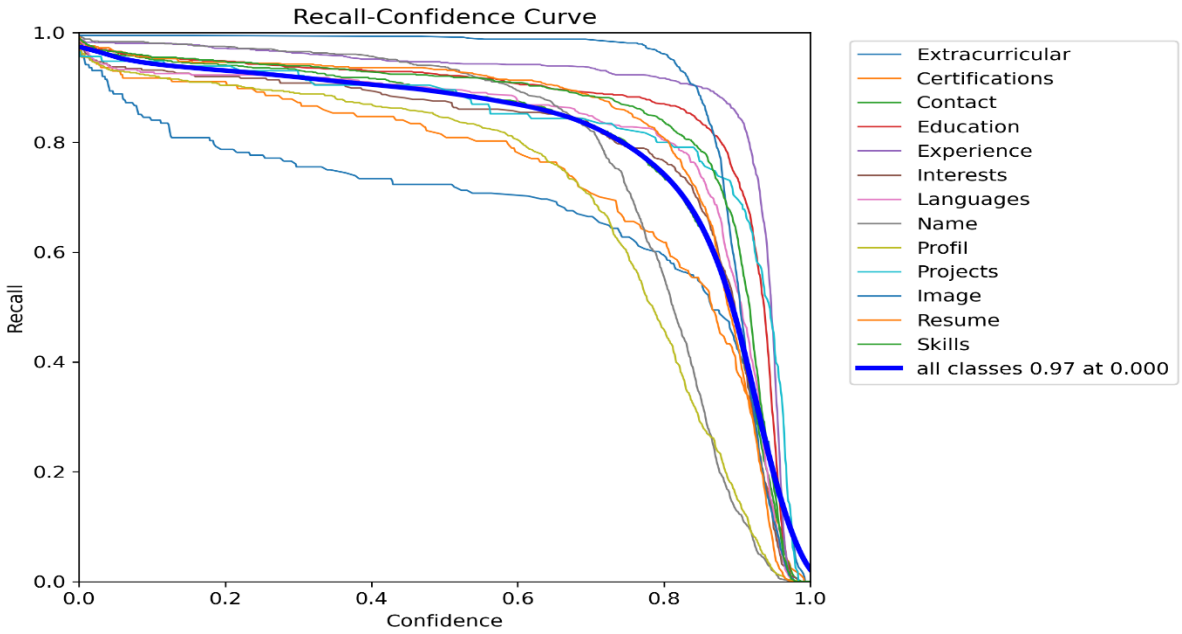


Figure 0.9: *Recall-Confidence Curve (YOLO)*

Error Analysis

The confusion matrix provides insights into common misclassifications. Categories such as **Profile**, **Projects**, and **Extracurricular** exhibited relatively lower mAP@0.5:0.95, indicating that the model struggled with these classes. This can be attributed to:

- **Visual Similarity:** Overlapping features between categories like "Profile" and "Resume Header".
- **Class Imbalance:** Lower frequency of categories like "Projects" and "Extracurricular" in the training set.
- **Text Density:** Denser text regions making object localization challenging.

Inference Speed and Efficiency

- **Preprocessing Time:** 0.42 ms per image.
- **Inference Time:** 14.75 ms per image.
- **Post-processing Time:** 1.56 ms per image.
- **Average Loss during Training:** 0.028.

The model achieves real-time performance, making it suitable for deployment in resume parsing systems.

Detection Example:

Figure 0.10: *Layout Analysis Result*

4.3.3.3 Faster RCNN Performance

The Faster R-CNN model, utilizing a ResNet-50 backbone, demonstrated moderate performance on the resume dataset. The overall evaluation metrics on the test set are summarized below:

Training summary

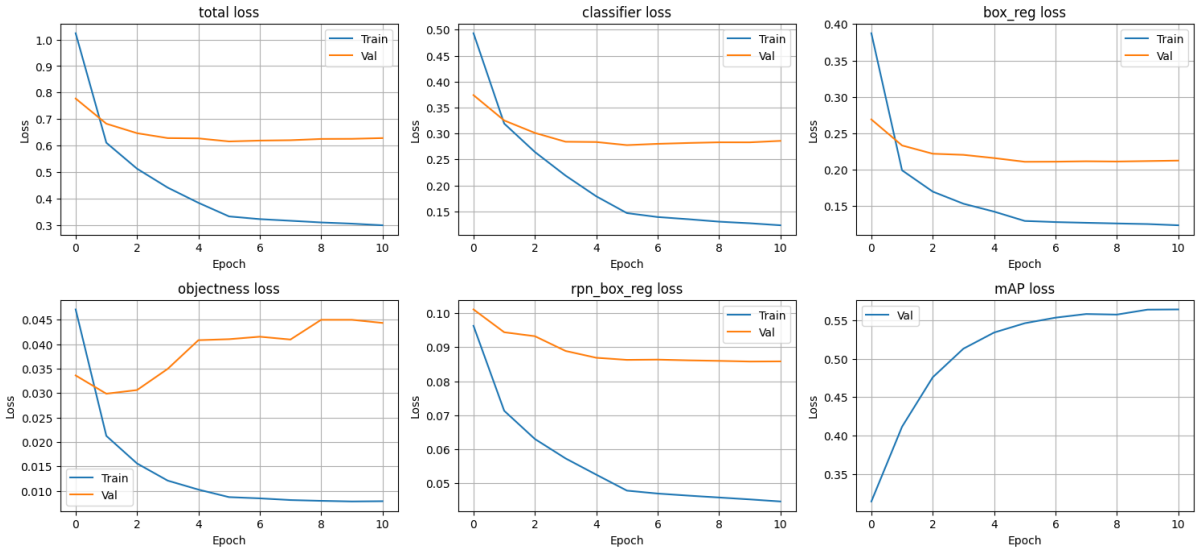


Figure 0.11: *Faster-RCNN Loss*

Overall Performance Metrics:

- **Mean Average Precision (mAP@0.5):** 0.538
- **Mean Intersection over Union (mean IoU):** 0.755
- **Precision:** 0.668
- **Recall:** 0.620
- **F1-Score:** 0.615

Class-wise Performance Metrics:

Class	Precision	Recall	F1-Score	mAP@0.5
Extracurricular	0.000	0.000	0.000	0.024
Certifications	0.500	0.100	0.167	0.067
Contact	0.744	0.811	0.776	0.626
Education	0.693	0.718	0.705	0.533
Experience	0.701	0.758	0.728	0.559
Interests	0.586	0.554	0.569	0.346
Languages	0.718	0.735	0.726	0.542
Name	0.957	0.969	0.963	0.931
Profile	0.758	0.753	0.756	0.595
Projects	0.667	0.077	0.138	0.067
Image	0.984	1.000	0.992	0.984
Resume	0.735	0.881	0.801	0.657
Skills	0.640	0.707	0.672	0.487
Overall	0.668	0.620	0.615	0.494

Table 0.4: *Performance Metrics by Class (Faster-RCNN)*

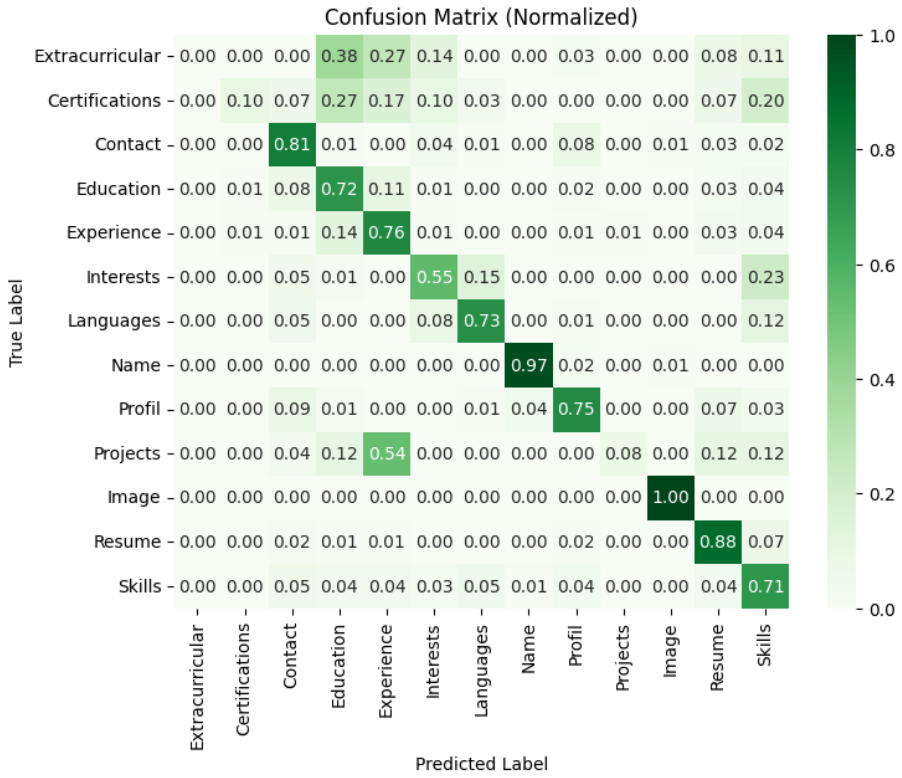


Figure 0.12: *Confusion Matrix (Faster-RCNN)*

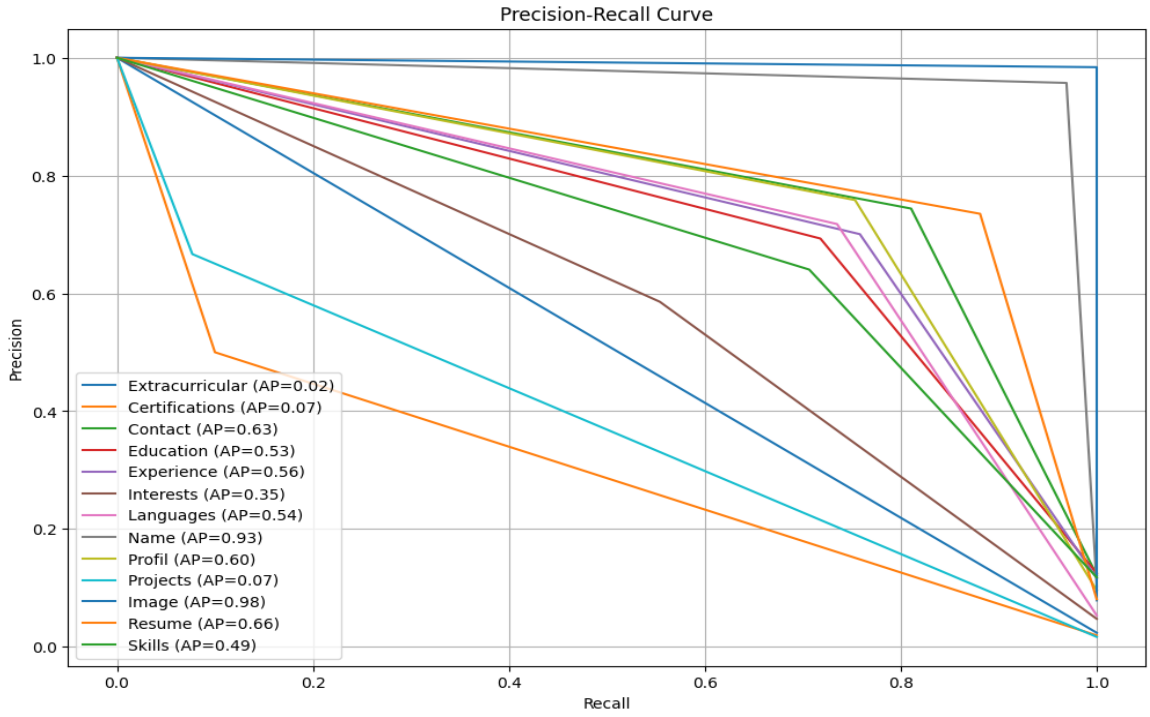


Figure 0.13: *Precision-Recall Curve (Faster-RCNN)*

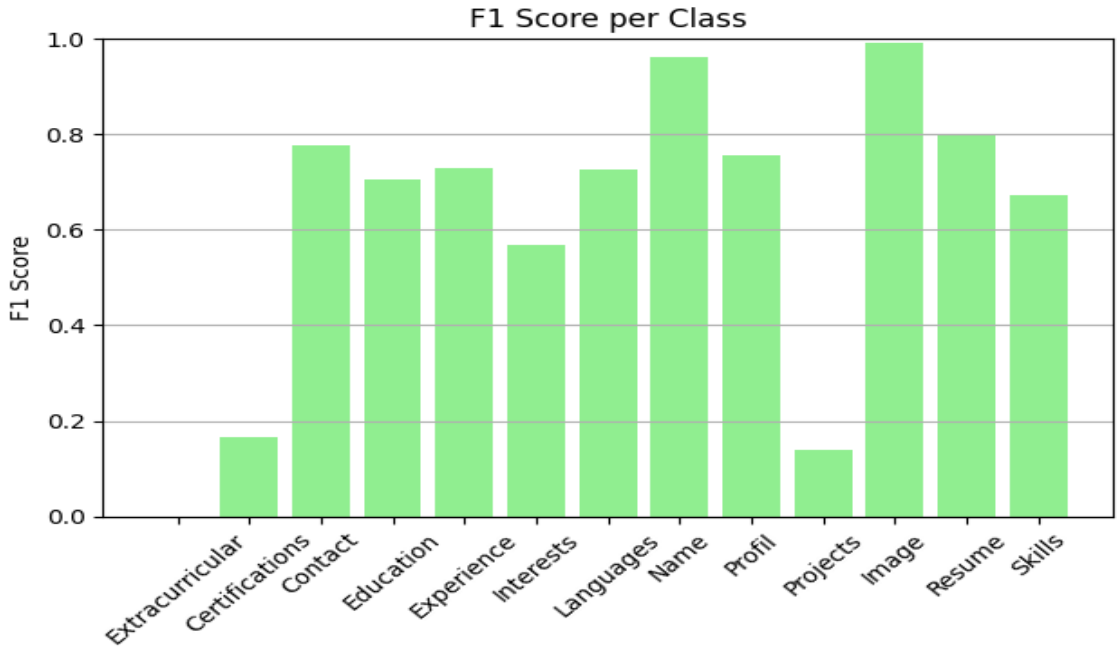


Figure 0.14: *F1 Per Class (Faster-RCNN)*

Performance Analysis:

- **High-Performing Categories:**
  - The "Image" and "Name" categories achieved the highest precision and recall, reflecting the model's ability to accurately localize these visually distinct fields.
  - "Resume Header" also performed well, benefiting from consistent layout patterns.
- **Low-Performing Categories:**
  - "Extracurricular" and "Projects" exhibited the lowest performance, with near-zero recall and F1-scores. These categories likely suffer from:
    - **Class Imbalance:** Significantly fewer instances in the training set.
    - **Visual Complexity:** Overlapping textual regions and inconsistent formatting.
- **Moderate-Performing Categories:**
  - "Certifications," "Interests," and "Skills" achieved moderate results, indicating that the model struggled with their diverse layouts.

Error Analysis:

- **Misclassifications:** The confusion matrix further tells us that classes like "Projects" and "Extracurricular" are persistently misclassified, either due to being rare in the training set or because of resemblance with other text-based classes.
- **False Positives:** High accuracy labels such as "Image" and "Name" result in few false positives, indicating that the model is highly certain about detecting them.
- **False Negatives:** The poor performance of "'Extracurricular"' is due to having a high number of false negatives, i.e., the model did not identify such regions for a majority of cases.

Inference Efficiency:

- **Average Inference Time:** The model exhibited moderate inference speed due to the computational overhead of region proposal and region-based detection in Faster R-CNN.
- **Training Stability:** Despite the relatively low mAP, the model achieved consistent performance without severe overfitting.

4.3.3.4 DETR Performance

The Detection Transformer (DETR) model, a fully transformer-based object detection model, was also evaluated on the resume dataset. However, the model's performance was significantly lower than the other tested models. The evaluation metrics on the test set are as follows:

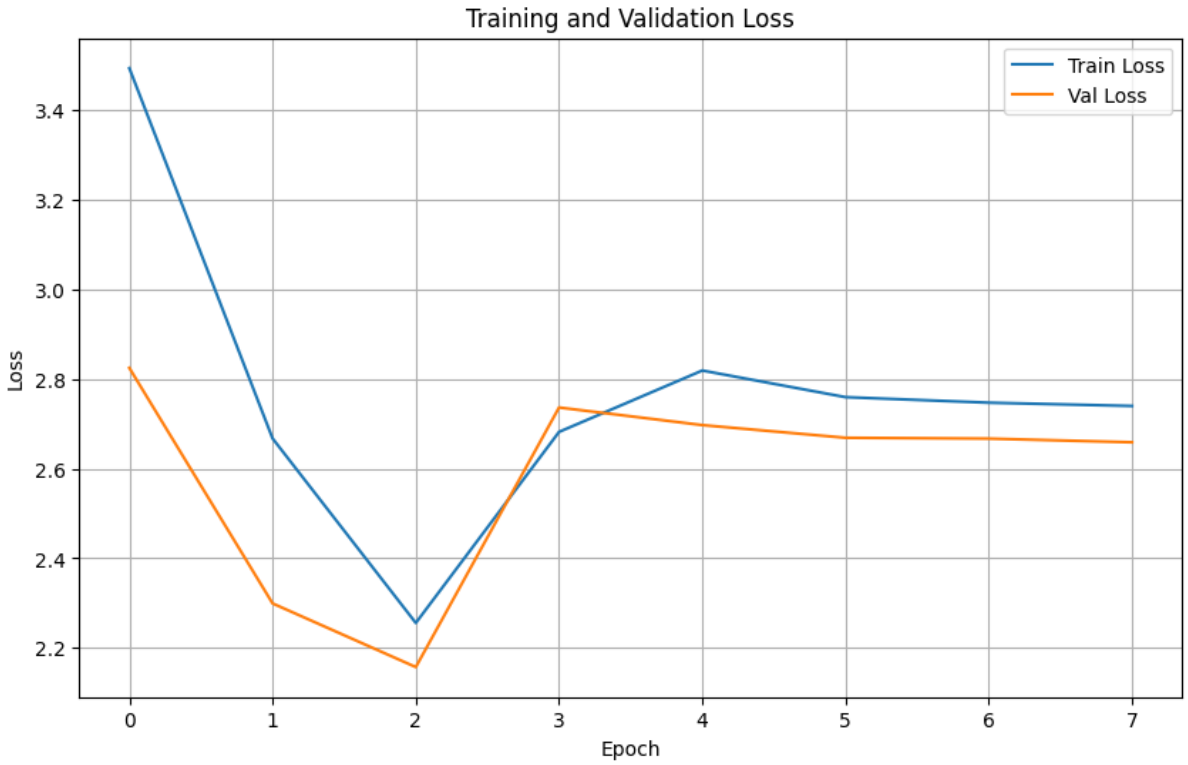


Figure 0.15: *DETR Loss*

Overall performance measures

- **Mean Average Precision (mAP@0.5):** 0.0923
- **Mean Average Precision (mAP@0.5:0.95):** 0.0316
- **Precision:** 0.0923
- **Recall:** 0.0316
- **F1-Score:** 0.0427

Performance Analysis

- **Poor detection performance:** DETR model did a poor job of detection and classifying text regions on the resume dataset, evidenced by its extremely low mAP and F1-score.
- **Inadequate Training Data:** Most probably, this poor performance is due to a small training dataset. Transformer-based models such as DETR usually need extensive training data if they are going to be able fulfill intricate spatial and contextual relationships.
- **Complex Layouts:** Resumes contain complex layout and dense text regions, which were perhaps making it even harder for the DETR model to distinguish among a variety of categories.
- **Lack of Pre-training on Document Images:** Unlike models pre-trained on natural images, DETR may lack pre-training focused specifically on document analysis, which would decrease its performance on such an operation.

Analysis Error

- **High False Negatives:** The model did not detect and classify objects consistently, and had extremely low recall.
- **Overfitting due to Small Data:** The model would have overfit the training set and not generalized to the test set due to insufficient size of data.
- **Inconsistent detection regions:** The global attention-based architecture of the model could be tested by the extremely localized presence of text objects within resumes.

Why DETR Failed

The reasons for DETR's poor performance are as follows:

- **Limited Training Data:** Transformer-based models such as DETR need a lot of data to learn object relationships.
- **No Domain-Specific Pre-training:** The model may not have been pre-trained on document images, and therefore it is difficult for it to generalize to the resume format.
- **Complex Layouts:** Resume pictures possess a mix of text and non-text regions, which could be challenging for models primarily trained for natural image detection.

Conclusion

The YOLOv11 performed far above Faster R-CNN and DETR on both accuracy and speed. Hence, it became our detection model of choice for our final pipeline.

4.3.4 OCR

Based on the OCR technologies reviewed in the *State of the Art* (Section **3.2**), a comparative evaluation was conducted to select the most suitable engine for extracting text from the Regions of Interest (ROIs) in our image-based documents. The engines evaluated included:

- **Tesseract OCR** – Open-source engine by Google *(Smith, 2007; Google, 2022)*
- **EasyOCR** – CRNN-based multilingual engine *(JaidedAI, 2021)*
- **PaddleOCR** – Deep learning-based OCR framework *(Baidu, 2022)*
- **TrOCR** – Transformer-based OCR by Microsoft *(Li et al., 2021)*

After empirical testing, **Tesseract** was ultimately chosen for the production pipeline due to its optimal balance between recognition performance and computational efficiency.

4.3.4.1 OCR Evaluation Metric

Each engine was benchmarked using:

1) **Character Error Rate (CER)**: Measures the proportion of incorrect characters in the predicted text compared to the ground truth.

$$CER = \frac{\textbf{\textit{Levenshtein Distance}}}{\textbf{\textit{Total Characters in Ground Truth}}} = \frac{\textbf{\textit{S+D+I}}}{N}$$

where:

- **S** is the number of substitutions,
- **D** is the number of deletions,
- **I** is the number of insertions,
- **N** is the total number of characters in the ground truth text.

2) **Word Error Rate (WER)**: Measures the proportion of incorrect words, accounting for insertions, deletions, and substitutions.

$$WER = \frac{\mathbf{S + D + I}}{N}$$

where:

- **S** is the number of substitutions,
- **D** is the number of deletions,
- **I** is the number of insertions,
- **N** is the total number of words in the ground truth text.

3) **Field-level Accuracy**: The percentage of correctly extracted field values from the OCR output, reflecting how well the recognized text aligns with expected structured fields.

$$\textbf{\textit{Field} − \textit{level Accuracy}} = \frac{\textbf{Number of Correctly Extracted Fields}}{\textbf{Total Number of Fields}} \times 100$$

where:

- "Correctly Extracted Fields" refers to the number of fields where the OCR output matches the expected value for that field,
- "Total Number of Fields" refers to the total number of fields that need to be extracted.

4.3.4.2 Synthetic Data Evaluation: A Controlled Experiment

I conducted a controlled experiment using a **custom synthetic dataset** specifically created for this study. The synthetic images were generated using text content sourced from **Wikipedia**, selected for its rich and diverse vocabulary as well as complex linguistic constructions.

The purpose of this experiment was to simulate real-world document scenarios while maintaining **full control over the ground truth**, allowing for **precise measurement of OCR engine performance**. The controlled nature of this dataset ensured a fair, consistent, and reproducible environment for evaluation.

All OCR engines under comparison including Tesseract, PaddleOCR, and EasyOCR were tested on this dataset. Evaluation was conducted using **standard OCR metrics**: Word Error Rate (WER), Character Error Rate (CER), and Field-level Accuracy (as defined in Section 3.3.4.1).

The results of this experiment, including quantitative comparisons and discussion, are presented in the next section ***Table 3***.

OCR Engine	Average WER	Average CER	Accuracy	Average Inference Time (s/image)
Tesseract	0.0733	0.0299	85.00%	0.8375
PaddleOCR	0.0563	0.0158	95.50%	6.3008
EasyOCR	0.1331	0.0321	76.00%	12.6473

Table 0.5: *OCR Performance Metrics on Synthetic Data*

4.3.4.3 Discussion: Balancing Accuracy and Efficiency

The results of the synthetic data evaluation provide critical insights into the strengths and limitations of each OCR engine:

1. **PaddleOCR:**
    - Demonstrated the best overall accuracy, achieving the lowest WER (0.0563) and CER (0.0158), with an accuracy of 95.5%.
    - However, this superior recognition capability comes at the cost of significantly slower processing speed (6.3 seconds per image).
    - PaddleOCR is most suitable for applications where recognition accuracy is prioritized over processing speed, such as detailed document digitization or multilingual text extraction.

2. **Tesseract:**
    - Offers a strong balance between accuracy and efficiency, with a WER of 0.0733 and a CER of 0.0299, while maintaining a much faster average inference time of 0.8375 seconds per image.
    - The high threshold-based accuracy (85.0%) further confirms its reliability, making it an excellent choice for large-scale text extraction tasks where speed is crucial.
    - Given that the primary language of the resume dataset is English, Tesseract's robust English recognition further justifies its selection.
3. **EasyOCR:**
    - While relatively straightforward to implement, EasyOCR exhibited the lowest performance in terms of accuracy, with the highest WER (0.1331) and CER (0.0321).
    - It also demonstrated the slowest inference speed (12.6473 seconds per image), making it the least suitable choice for this study's objectives.

4.3.4.4 Final OCR Engine Selection

Considering the specific requirements of this study namely, efficient and accurate text extraction from English-language resumes **Tesseract** was selected as the primary OCR engine. Its balance of speed and acceptable recognition accuracy aligns well with the large-scale, real-time nature of the project.

However, it is important to note that **PaddleOCR** remains a viable alternative if recognition accuracy is prioritized over speed. For future improvements or advanced deployments where computational resources are sufficient, PaddleOCR may offer enhanced performance, especially for complex or multilingual document analysis.

4.3.4.5 Preprocessing Steps Applied Before OCR To The Cropped Images

To maximize OCR performance, several image preprocessing steps were applied to the cropped text regions before feeding them to the OCR engine:

1. **Grayscale Conversion** Simplifies the image by removing color information, focusing only on intensity, which reduces the dimensionality and simplifies downstream processing.
2. **Denoising with Non-Local Means Filtering** Utilizes a noise-removal function that smoothes equivalent patches all over an image without compromising text edges. This works particularly well for the eradication of background noises without compromising details of text.
3. **Resizing Crops to Improve Small Font Recognition** Crops containing small fonts were resized (upscale) to increase the size of characters, thus making them easier to recognize. Resizing was done while preserving the aspect ratio to avoid geometric distortions.

4.3.5 Qualitative Evaluation

In addition to the detection performance and OCR metrics, a **qualitative evaluation** of the pipeline was performed. By visually inspecting a subset of resumes processed by the system, it was found that the pipeline consistently extracted key fields, including Name, Email, and Education, with high accuracy when the resumes were well-structured. However, **minor errors** were observed in cases with **poor scan quality**, **unusual fonts**, or **complex layouts** like Work Experience sections with dense text blocks. Despite these challenges, the system showed promise for handling typical resume formats and could reliably extract key data for most cases.

The combination of **robust detection** from YOLOv11, **efficient text extraction** from Tesseract OCR, and **post-processing rules** provided a functional and reliable pipeline for structured data extraction.

While no exact **Full Extraction Accuracy** or **Partial Extraction metrics** were calculated in this study, the **system's effectiveness** was supported by the promising **detection** and **OCR results**, along with qualitative insights.

4.3.6 Result Visualizations

Sample qualitative results are provided below:

- **Bounding boxes** drawn over fields like Name, Education, Work Experience …

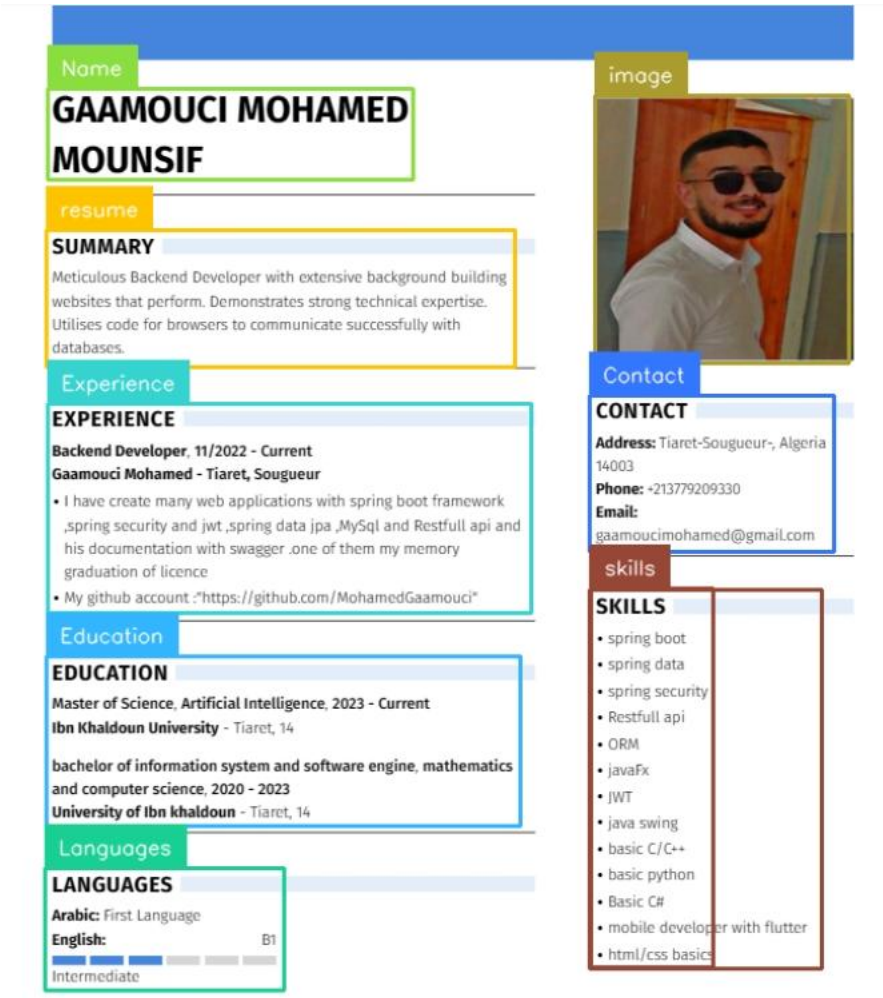


Image 1: *Layout Analysis Example*

- **Extracted text** shown next to the detected boxes
- **Post-processed output** organized in a structured JSON-like format

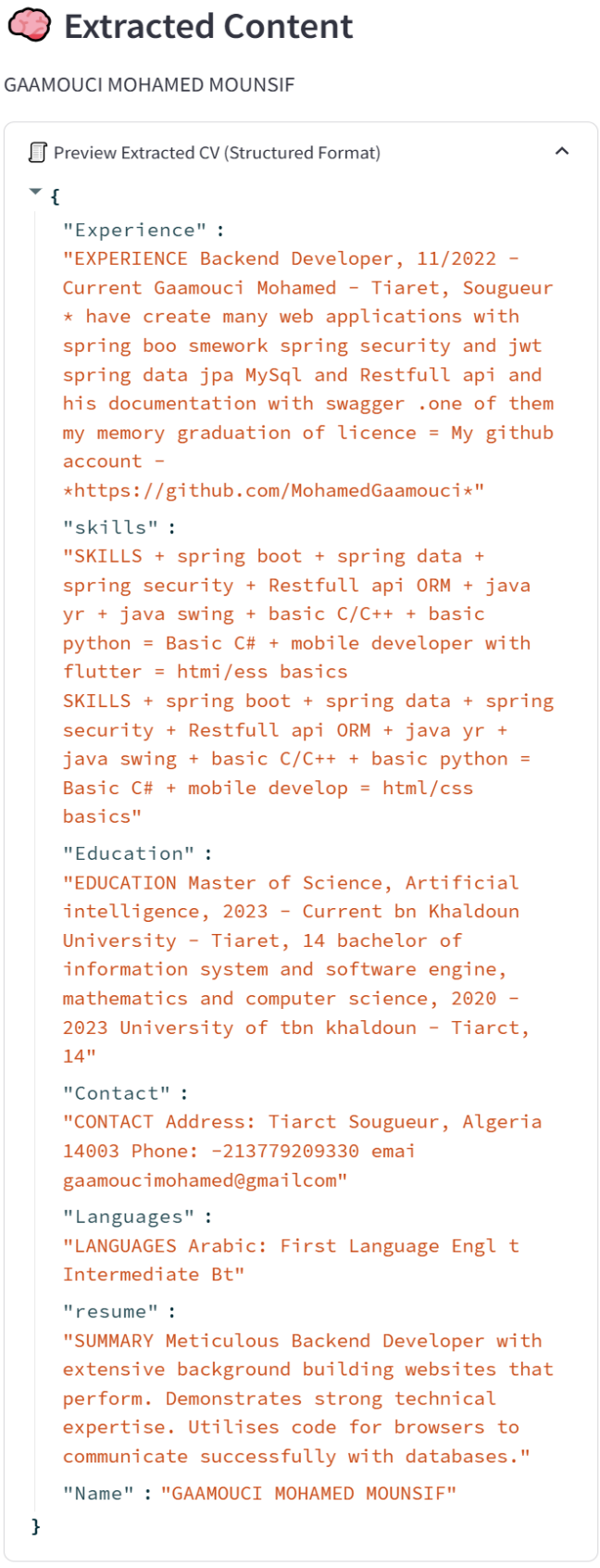


Image 2: *OCR resutl Example*

4.3.7 Summary of Findings

- **YOLOv11 + Tesseract** combination provided a strong balance of speed and accuracy.
- Fine-tuning Tesseract preprocessing (thresholding, resizing) significantly improved OCR performance.
- Post-processing helped standardize extracted fields (e.g., cleaning phone numbers, standardizing degree names).
- The main challenges remained in handling highly non-standard CV templates or very low-quality scans.

4.3.8 Web Application: Interactive Resume Processing with Streamlit

To facilitate users working with the resume information extraction pipeline, a Streamlit-based web application has been created. Streamlit is a lightweight yet strong Python rapid application development framework for creating interactive web apps. The application supports multi-batch processing for resumes and includes advanced filtering capabilities so users are able to exclude extractions by certain criteria (skills, education, experience, etc.). The interactive interface makes it far more usable and accessible, and it is a feasible option for everyday use.

4.3.8.1 Architecture and Workflow

The web application is designed using a client-server architecture, where the client interacts with a graphical interface, and the server hosts the backend processing logic. The primary workflow of the application is as follows:

1. **User Interface (Frontend)**
    - Users can upload one or multiple resume images in formats such as JPG, PNG, or PDF.
    - A clean and intuitive layout is provided, with buttons for uploading images, starting the processing task, and applying filters.
    - A sidebar allows users to define filtering criteria, such as searching for resumes containing specific skills (e.g., "Python", "Machine Learning"), educational qualifications, or work experience.
2. **Backend Processing (Resume Information Extraction Pipeline)**
    - Upon receiving the uploaded files, the application triggers the resume processing pipeline for each document in batch mode:
        - **Detection:** Each image is processed using the YOLOv11 model, which identifies key sections (e.g., Name, Education, Skills).
        - **Text Extraction:** Detected fields are cropped and processed using Tesseract OCR for text recognition.
        - **Post-Processing:** Extracted text is cleaned, parsed, and structured according to predefined categories.
    - The extracted information from all resumes is stored in a structured format (JSON or DataFrame).
3. **Filtering Functionality**
    - Users can dynamically filter the processed resumes using various criteria:
        - **Skills:** Display resumes containing specific skills (e.g., "Python", "Project Management").

        - **Education:** Filter resumes based on academic degrees (e.g., "Master's Degree", "Bachelor's Degree").
        - **Experience:** Search for resumes with a specified number of years of experience.
        - **Certifications:** Identify resumes with specific certifications (e.g., "AWS Certified Solutions Architect").
    - The filtering process is fast and responsive, enabling real-time exploration of the extracted data.
4. **Output Display and Download**
    - The extracted and filtered resume information is displayed in a structured format (JSON or tabular view).
    - Users can directly download the processed data as a CSV or JSON file for further use.

4.3.8.2 Streamlit Application Design

The Streamlit application was designed with a modular structure to enhance usability and flexibility:

- **File Uploader:** Allows users to upload one or multiple resume images (JPG, PNG, PDF).
- **Batch Processing:** Supports concurrent processing of multiple resumes, significantly improving workflow efficiency.
- **Sidebar Controls:**
    - Filter criteria for selecting specific skills, education, experience, or certifications.
    - Real-time search functionality for immediate feedback.
- **Extraction Results Display:**
    - Processed resumes are displayed in a structured tabular format, making it easy to view, filter, and analyze extracted information.
    - The table is automatically updated based on applied filters.
- **Download Option:**
    - Allows users to download the extracted data (either raw or filtered) as a CSV or JSON file.

4.3.8.3 Technical Implementation

- **Programming Language:** Python (Streamlit framework).
- **Backend Framework:** PyTorch for YOLOv11 and Tesseract for OCR.
- **Batch Processing Logic:** Implemented using Python's `multiprocessing` and `concurrent.futures` modules for efficient parallel processing.
- **Filtering Mechanism:**
    - Extracted text is stored in a Pandas DataFrame, enabling fast filtering using Pandas query syntax.
    - Users can filter by any detected field (e.g., Skills, Education, Experience) using regular expressions or keyword matching.
- **Preprocessing:** Uploaded images are automatically resized, denoised, and normalized before processing, ensuring consistent performance.

- **Deployment Environment:** The application was initially developed and tested locally, but is designed for easy deployment on cloud platforms (e.g., Heroku, AWS, Azure) for scalable use.

4.3.8.4 User Experience

The multi-batch and filter-enabled web application provides a seamless user experience:

- **Multi-Document Processing:** Users can upload multiple resumes at once and process them in a single batch, saving time and effort.
- **Interactive Filtering:** Real-time filtering allows users to focus on specific qualifications, skills, or experiences of interest.
- **Instant Feedback:** The processed resumes and filtered results are displayed immediately without page reloads, offering a smooth and responsive experience.
- **Scalable Design:** The architecture can be easily extended to handle larger document batches without significant performance loss.

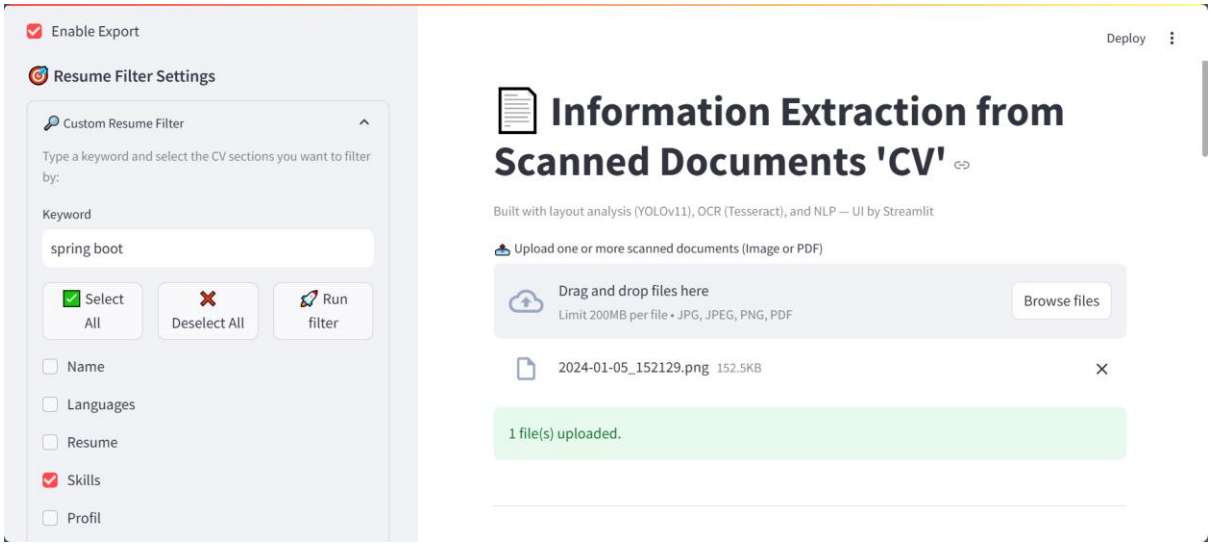
4.3.9 Training Module for Custom Document Types

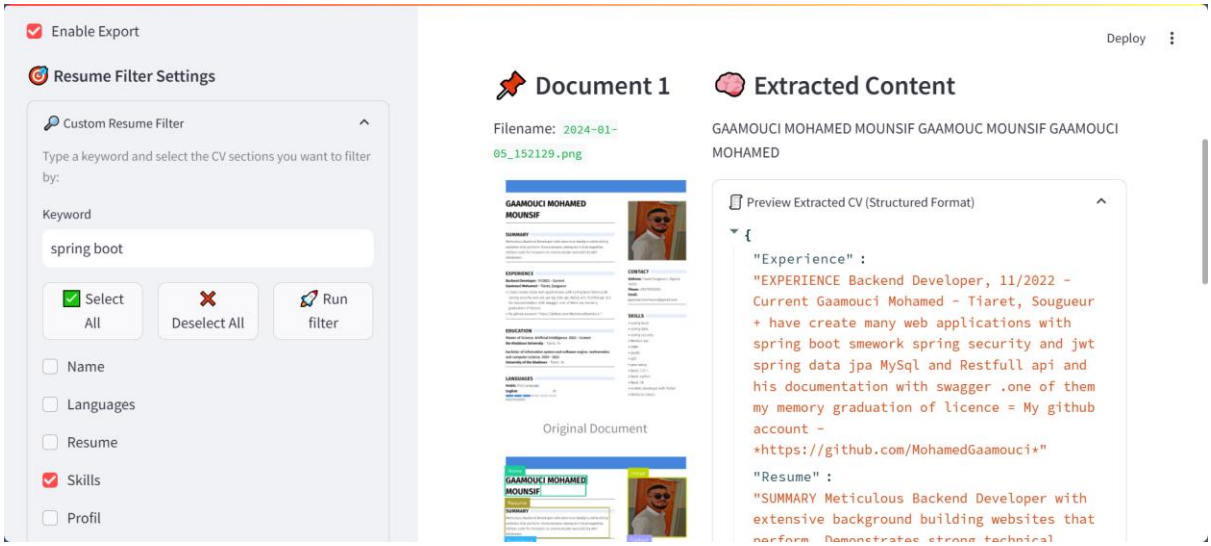
An additional feature in the web application allows users to train a custom object detection model directly. Users can upload their dataset (images and annotations) and initiate the training process. This provides a user-friendly way for non-technical users to adapt the pipeline to any document type without modifying the code.

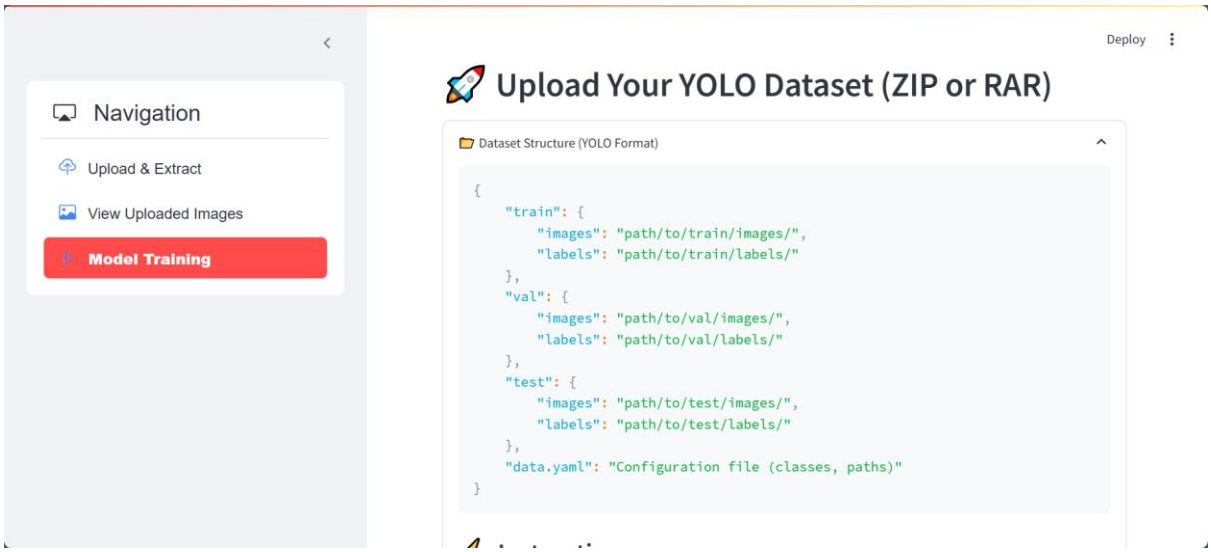
4.3.9.1 Future Improvements

- **Enhanced Filtering:** Adding advanced query capabilities, such as AND/OR conditions for multiple criteria.
- **Security Enhancements:** Implementing user authentication and file encryption to secure uploaded documents.
- **Dynamic Field Detection:** Allowing users to customize the detection categories based on their requirements.
- **Advanced Visualization:** Providing graphical insights into the extracted information (e.g., skill distribution across resumes).

By integrating this web application, the proposed pipeline becomes a complete, scalable, and accessible solution for automated resume parsing, suitable for HR professionals, recruiters, and organizations seeking efficient document processing.


Enable Export
Resume Filter Settings
Custom Resume Filter
Type a keyword and select the CV sections you want to filter by:
Keyword
spring boot
Select All
Deselect All
Run filter
Name
Languages
Resume
Skills
Profil
Deploy
Information Extraction from Scanned Documents 'CV'
Built with layout analysis (YOLOv11), OCR (Tesseract), and NLP — UI by Streamlit
Upload one or more scanned documents (Image or PDF)
Drag and drop files here
Limit 200MB per file • JPG, JPEG, PNG, PDF
Browse files
2024-01-05_152129.png 152.5KB
1 file(s) uploaded.


Enable Export
Resume Filter Settings
Custom Resume Filter
Type a keyword and select the CV sections you want to filter by:
Keyword
spring boot
Select All
Deselect All
Run filter
Name
Languages
Resume
Skills
Profil
Deploy
Document 1
Filename: 2024-01-05_152129.png
Original Document
Extracted Content
GAAMOUCI MOHAMED MOUNSIF GAAMOUC MOUNSIF GAAMOUCI MOHAMED
Preview Extracted CV (Structured Format)
{
"Experience" :
"EXPERIENCE Backend Developer, 11/2022 - Current Gaamouci Mohamed - Tiaret, Sougueur + have create many web applications with spring boot smework spring security and jwt spring data jpa MySql and Restfull api and his documentation with swagger .one of them my memory graduation of licence = My github account - *https://github.com/MohamedGaamouci*"
"Resume" :
"SUMMARY Meticulous Backend Developer with extensive background building websites that


Navigation
Upload & Extract
View Uploaded Images
Model Training
Deploy
Upload Your YOLO Dataset (ZIP or RAR)
Dataset Structure (YOLO Format)
{
"train": {
"images": "path/to/train/images/",
"labels": "path/to/train/labels/"
},
"val": {
"images": "path/to/val/images/",
"labels": "path/to/val/labels/"
},
"test": {
"images": "path/to/test/images/",
"labels": "path/to/test/labels/"
},
"data.yaml": "Configuration file (classes, paths)"
}

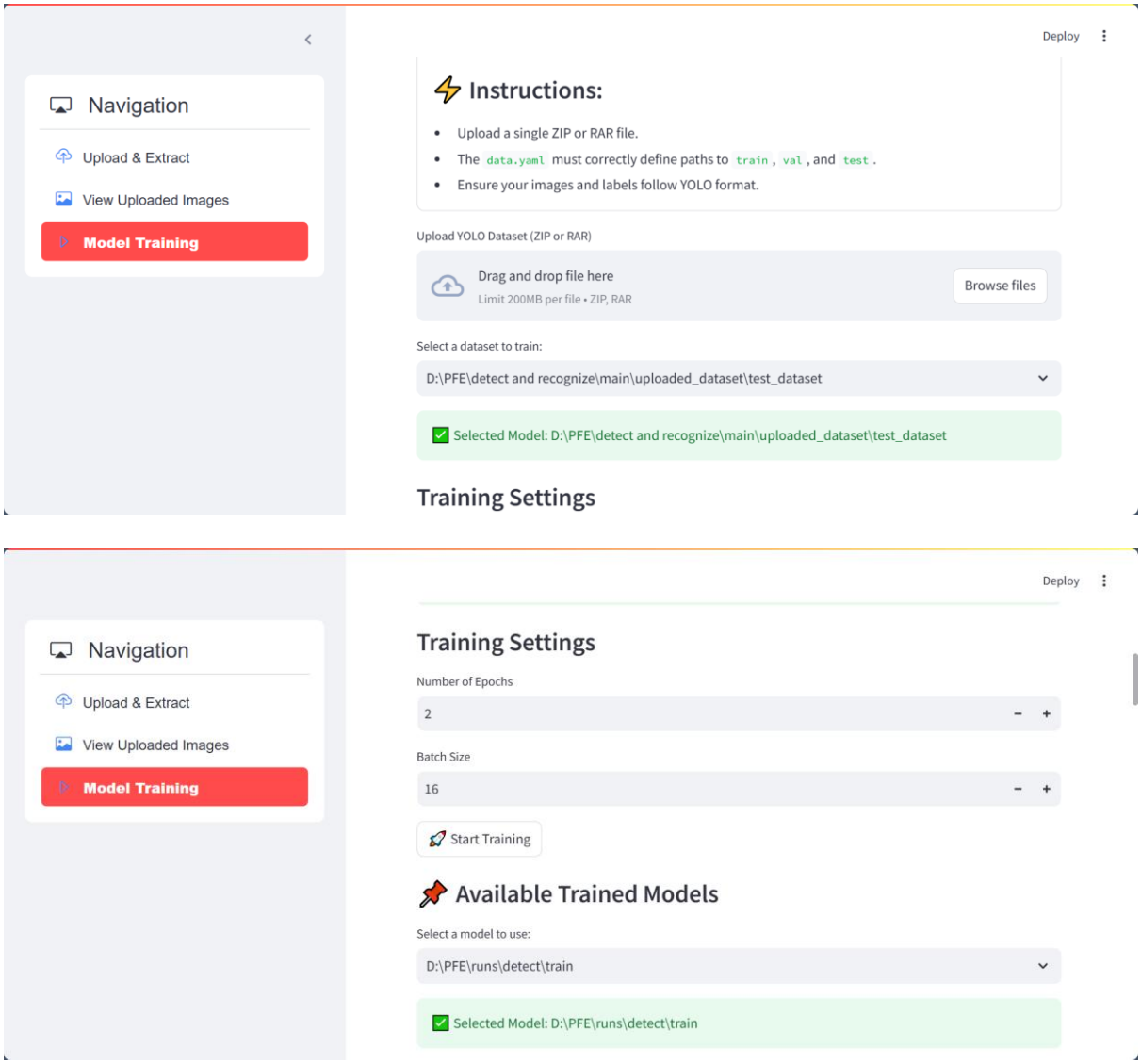


Image 3: *Wet App Screen shots*

4.3.10 Hardware and Software Environment

This section outlines the hardware and software configuration used throughout the development and experimentation phases of this study. All model training, inference, and evaluations including object detection, optical character recognition (OCR), and related post-processing were conducted in a consistent computing environment to ensure reliability and reproducibility of results. The selected tools and frameworks were chosen for their robustness, ease of integration, and alignment with the project's performance requirements.

Hardware:

- **CPU:** Intel Core i5-6 generation
- **GPU:** P100 (Kaggle GPU)
- **RAM:** 16 GB (Kaggle)

Software:

- **Programming Language:** Python 3.10
- **Deep Learning Frameworks:** PyTorch 2.0 (for Faster R-CNN)
- **Object Detection Toolkits:** YOLOv11 (Ultralytics PyTorch implementation), Faster R-CNN (PyTorch implementation)
- **OCR Engine:** Tesseract 5.3.0 (via `pytesseract` Python package)
- **Annotation Tool:** Roboflow
- **Other Libraries:** OpenCV, Albumentations (for augmentations), NumPy, Matplotlib
- **Post-processing:** Custom post-processing scripts for object detection and OCR output

Training Setup:

- The experiments were conducted on **Kaggle Notebooks**, utilizing the free limited GPU resources provided by the platform. No cloud computing resources were used beyond Kaggle.

4.4 Discussion

This section provides a critical analysis of the experimental results. It discusses the strengths, challenges, and limitations of the proposed resume information extraction pipeline and suggests possible future improvements.

4.4.1 Strengths of the Proposed Pipeline

- **High Detection Accuracy**: The YOLOv11 model demonstrated excellent capability in localizing key fields across a wide variety of resumes, achieving high precision, recall, and mAP scores even on unseen templates. This confirms YOLOv11's strong generalization ability when trained with diverse annotated examples.
- **Efficiency and Scalability**: Thanks to YOLOv11's lightweight architecture and fast inference time (~28 ms per image), the pipeline is well-suited for real-world applications that require processing thousands of resumes in a short time.
- **Robust OCR Performance**: Despite the varying text styles and fonts, Tesseract OCR, coupled with appropriate pre-processing, yielded low character and word error rates, enabling reliable extraction of textual information from detected fields.
- **Effective Post-processing**: Post-processing steps (such as regular expression cleaning, normalization of contact details, and keyword-based organization) played a crucial role in converting noisy raw OCR output into structured, machine-readable data formats.

4.4.2 Challenges Encountered

- **Low-Quality Scans and Noisy Inputs**: OCR performance dropped significantly on resumes with poor resolution, heavy background noise, or faded text, leading to character-level mistakes that sometimes affected critical fields like phone numbers and emails.
- **Creative Resume Layouts**: Highly customized or artistic CV designs, with non-standard field placements or decorative fonts, confused the detection model, leading to missed detections or incorrect field associations.

- **Limited Training Annotations**: Although the manually created dataset was substantial (~2300 annotated images), certain field types (like Certifications or Skills) were underrepresented compared to core sections like Education and Work Experience, potentially causing slight detection bias.
- **Handwritten Elements**: In rare cases where handwritten notes were present on resumes (e.g., corrections, manual signatures), both detection and OCR stages struggled, as handwritten text is notably harder to recognize using standard Tesseract OCR.

4.4.3 Lessons Learned

- **Importance of Diverse Data**: Annotating a wide variety of resume styles significantly improved the model's generalization. Greater template diversity during training reduces the risk of overfitting to specific document designs.
- **Preprocessing Matters**: Techniques such as skew correction, and noise removal substantially enhanced OCR quality, showing that pre-processing is as critical as model choice.
- **Model Selection Justification**: The experiments validated that YOLO-based architectures are preferable over two-stage detectors like Faster R-CNN for tasks where real-time performance and layout complexity are important factors.

4.4.4 Limitations

- **Dependency on OCR Quality**: The pipeline's final output quality is heavily reliant on OCR success. Any OCR mistakes (e.g., in names or emails) can significantly impact downstream systems like applicant tracking or contact databases.
- **Fixed Field Detection**: The detection model was trained on specific fields. If a resume contains new, unseen sections (e.g., "Volunteer Experience" or "Awards"), the current model would not detect them without additional training.
- **Domain Adaptability**: While the pipeline is effective for English-language resumes, its performance on resumes in other languages was not evaluated, and would likely require retraining and adjustments.

4.4.5 Potential Improvements and Future Work

- **Advanced OCR Models**
- **Semi-supervised Learning for Detection**: Leveraging semi-supervised learning techniques could reduce the manual annotation burden by allowing the model to learn from partially labeled or unlabeled resume images.
- **Multilingual Support**: Expanding the pipeline to handle resumes in different languages would make the solution more globally applicable.

In conclusion, the proposed resume information extraction pipeline demonstrated strong performance and practicality, yet certain challenges related to scan quality, layout diversity, and OCR robustness remain. Addressing these limitations will be critical for deploying the system at scale in diverse and dynamic real-world environments.

4.5 Chapter Summary

This chapter offered a comprehensive description of the proposed pipeline for extracting structured data from scanned resumes.

The major findings and contributions are listed as follows:

- **Pipeline Description**: The chapter outlined the whole end-to-end pipeline, detailing all the components—from resume image collection, pre-processing, field detection via YOLOv11, text extraction via Tesseract OCR, up to final post-processing for structured output generation.
- **Dataset Preparation**: There were significant manual efforts which were put into creating a good quality annotated dataset of about 2300 resume images. This served to train the detection model adequately and allow it to perform well even for various resume templates and formats.
- **Model Selection and Evaluation**: Faster R-CNN, YOLOv11 and DETR were compared, and YOLOv11 is utilized since it offered better detection and real-time detection. Measures such as precision, recall, and mean Average Precision (mAP) proved the performance of the model.
- **Results Analysis**: Large-scale experiments showed strong detection and extraction even for moderately harsh conditions (e.g., light-level noise, heterogeneous configurations). The pipeline proved sensitive, however, to radical layout extremes and input quality.
- **Discussion of Findings:** The weaknesses, strengths, and boundaries of the proposed system were examined critically. Key takeaways highlighted the importance of mixed training data, the supreme importance of image pre-processing for OCR success, and the need for potential future evolution such as multilingual support and increased integration of OCR.

In conclusion, this chapter demonstrated that the designed pipeline provides a practical and scalable solution for automatic resume information extraction. Despite some remaining challenges, the experimental results indicate strong potential for real-world deployment in human resources (HR) automation systems and beyond.

5 Conclusion and Future Work

The objective of this project was to develop a smart pipeline which would be able to extract structured data from scanned resumes (CVs), a process of increasing relevance for today's hiring processes. The project kicked off with identification of resumes and preparation of an accurate dataset of resumes, for which supervised learning required manual annotation. Though there were no pre-existing annotations, a high-quality labeled dataset was prepared with the help of Roboflow platform, which turned out to be a strong foundation for training and testing.

The two object detection models, YOLO and Faster R-CNN among others, were tested for detection of relevant segments from resumes. YOLOv11 was used as the primary detection model based on comparison experiments since it had faster execution and more accurate detection, making it more suited for real-world application. Detection was followed by Tesseract OCR for extraction of content from detected segments, with custom post-processing for assurance of structured and usable extracted content.

The experimental results confirmed the efficiency of the proposed pipeline. The system demonstrated the ability to precisely extract and identify important resume fields, such as names, contact information, education, and experience, regardless of layout, fonts, and image variability. Furthermore, deep learning-based detection with OCR and structured post-processing were also confirmed to be an effective solution for handling issues with resume digitization.

Overall, the pipeline developed is an important leap for human resource management toward automating resume screening. It also holds great promise for decreasing human labor, streamlining the efficiency of screening candidates, and facilitating more data-driven hiring.

5.1 Limitations

While the pipeline created proved effective at demonstrating ability to extract and identify structured data from scanned resumes, a number of weaknesses were also encountered during experimentation and actual field trials.

- **Dataset boundaries**
  The test and training dataset included around 2,300 annotated images of resumes. While adequate for a proof of concept, a limitation of generalization for the model exists if the model were to be exposed to resumes from other parts of the world, industries, or nonstandard formats.
- **Quality and effort of annotation**
  The manual process of annotating, which was done via Roboflow, took a lot of time and was a little inconsistent. Since detection model performance is so sensitive to annotation accuracy, any deviation of bounding box placement could affect model performance.
- **Detection Model Challenges:**
  While YOLOv11's performance outperformed that of Faster R-CNN and the others, there were still detection errors. Resumes with extremely dense content, atypical

structuring, or graphically sophisticated layout (e.g., graphical resumes) would at times lead to detection failures or erroneous bounding boxes.

- **OCR Limitations:**
  Though Tesseract OCR was powerful, it had problems with poorer-quality scans, handwriting, or very stylized fonts. In addition, OCR issues like character confusion sometimes carried over even into post-processing, which would need further cleaning or even manual editing.
- **Post-Processing Complexity**
  Post-processing rules were specifically customized for language and format of resumes within the dataset. Redefining or introducing post-processing logic would be required for expanding the pipeline to parse more than a single language (resumes that are non-English), different templates, or any other type of documents.
- **Lack of end-to-end automation**
  Although the system is able to automatically detect and extract, it currently does not possess an intelligent checking mechanism. For instance, incorrect fields (such as extracting an address into a field for a name) are currently not automatically indicated, and human checking would still need to be done for mission-critical operations.
- **Computational Efficiency**
  While YOLOv11 detection itself is fast enough for deployment, total detection, OCR, and post-processing inference together are a bottleneck for handling extremely large batches of resumes on less-than-top-of-the-line hardware.

5.2 Future Work

Based on recent achievements, several approaches can be taken to enhance the performance, scalability, and reliability of the pipeline:

1. Expansion and augmentation of the dataset

Having a larger, more diverse dataset with multiple resume templates, languages, and regional formats would further increase generalizability of the model. Incorporating multilingual resumes, handwritten CVs, and non-standard formats would make the system far more versatile and production-capable.

2. Developing Annotation Strategies

The use of semi-automatic annotation tools or active learning techniques would accelerate annotation without loss of quality. In addition, using data augmentation techniques such as rotation, scaling, and addition of noise would duplicate real-world variation and increase model robustness.

3. Refining detection models

Although YOLOv11 performed well, more recent models such as YOLOv8, DETR (DEtectionTRansformer), or specifically tuned hybrid models would potentially provide even greater accuracy, particularly for more complex or densely populated documents. Model ensembling techniques would also be worth considering for combining strength from multiple detectors.

4. Adopting End-to-End Doc Understanding Models

Future works could explore end-to-end deep learning models simultaneously doing layout analysis and information extraction, for instance, LayoutLM, Donut, or DocFormer. These remove detection and OCR discrimination, yielding more coherent and context-sensitive extraction.

5. Improving OCR Accuracy

Replacing Tesseract with newer versions of the OCR engine (such as EasyOCR, TrOCR) or with an OCR model fine-tuned on resume documents can largely reduce errors at a character level. Another approach would be using language models by incorporating OCR for automatic error correction post-recognition.

6. Post-Processing and Validation using Intelligence

The inclusion of a rule-based check module, statistical checking, or even machine learning classifiers would be able to detect anomalies (like digits within name fields) and correct or alert automatically. This would minimize human intervention further.

7. Scalability and Deployment:

Cloud platform or API service deployment optimization would support large-scale processing of documents. Inclusion of parallelism into batch processing, memory optimizations, and GPU acceleration would prepare the system for use by enterprises.

8. Generalizing to Other Types of Documents

The developed pipeline is extendable for use with other types of comparable documents such as cover letters, application forms, or certificates by training the system using domain-specific data. The applicability scope of the system would then become greater than for resumes alone.

9. Scalability and Customization: The current system can be extended to support more document types by leveraging the training module. Future work may focus on enhancing this training feature, providing pre-trained model templates for different document categories (invoices, medical reports, etc.) and offering advanced training options (e.g., hyperparameter tuning).

5.3 Summary

We've provided a comprehensive explanation of the pipeline for scanning resumes and extracting structured data throughout this research. The chapter has begun with an introduction of the pipeline architecture, with prime aspects such as dataset preparation, model selection, training processes, integration with the use of OCR, and post-processing techniques.

Experiments proved that YOLOv11 with Tesseract OCR and smart post-processing worked adequately for extraction of key fields of information, and detection from resumes. Comparative performance showed that YOLO-based models outperformed Faster R-CNN for this use case with better speed and detection accuracy for document detection tasks.

Despite the favorable results, there are several issues, particularly with regards to errors with OCR, variations in format of resumes, and generalizability of performance for new documents. These were addressed by proposing potential improvements and future research directions.

Overall, the constructed pipeline is a robust and effective platform for automatic resume parsing. It significantly reduces labor, enhances accuracy, and provides a solid platform for future enhancement and extension to greater general document comprehension tasks.

References

Alex Krizhevsky, Ilya Sutskever, & Geoffrey E. Hinton. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NeurIPS/NIPS)* (pp. 1097–1105). Lake Tahoe, Nevada, USA: Curran Associates, Inc. doi:10.1145/3065386

Baidu. (2022). *PaddleOCR*. Retrieved from https://github.com/PaddlePaddle/PaddleOCR

Carion, Nicolas, Massa, Francisco, Synnaeve, Gabriel, Usunier, Nicolas, Kirillov, Alexander, & Zagoruyko, Sergey. (2020). End-to-End Object Detection with Transformers. *European Conference on Computer Vision (ECCV)*. Récupéré sur https://arxiv.org/abs/2005.12872

Daniel Bolya, Chong Zhou, Fanyi Xiao, & Yong Jae Lee. (2019). YOLACT: Real-Time Instance Segmentation. *IEEE International Conference on Computer Vision (ICCV)* (pp. 9157–9166). Seoul, South Korea: IEEE.

Girshick, R. B. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 580–587). Columbus, OH, USA: IEEE. doi:10.1109/CVPR.2014.81

Google. (2022). *Tesseract OCR: Open source OCR engine*. Retrieved from https://github.com/tesseract-ocr/tesseract

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. doi:https://doi.org/10.1109/CVPR.2016.90

JaidedAI. (2021). *EasyOCR*. Retrieved from https://github.com/JaidedAI/EasyOCR

John Canny. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAM, 8*, 679–698. doi:10.1109/TPAMI.1986.4767851

Jonathan Long, Evan Shelhamer, & Trevor Darrell. (2015). Fully Convolutional Networks for Semantic Segmentation. *IEEE* (pp. 3431–3440). Boston, MA, USA: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi: 10.1109/CVPR.2015.7298965

K. T. Wong, A. Cai, & P. Shi. (1995). Document analysis with a top-down X-Y cut approach. *International Conference on Pattern Recognition (ICPR)* (pp. 925–927). Vienna, Austria: IEEE. doi:10.1109/ICPR.1996.546843

Kaiming He, Georgia Gkioxari, Piotr Dollár, & Ross B. Girshick. (2017). Mask R-CNN. *IEEE International Conference on Computer Vision (ICCV)* (pp. 2961–2969). Venice, Italy: IEEE. doi:10.1109/ICCV.2017.322

Li, Minghao, Yin, Fei, Zhang, Cheng, & Liu, Cheng-Lin. (2021). TrOCR: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*, https://arxiv.org/abs/2109.10282.

Lin, Tsung-Yi, Dollár, Piotr, Girshick, Ross, He, Kaiming, Hariharan, Bharath, & Belongie, Serge. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2117–2125. doi:https://doi.org/10.1109/CVPR.2017.106

Loshchilov, Ilya, & Hutter, Frank. (2019). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*. Récupéré sur https://arxiv.org/abs/1711.05101

ocr. (2022). cairo: ma3arf.

Olaf Ronneberger, Philipp Fischer, & Thomas Brox. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI). 9351*, pp. 234–241. Munich, Germany: Springer Cham. doi:10.1007/978-3-319-24574-4_28

Rafael C. Gonzalez, & Richard E. Woods. (2018 (4th edition)). Digital Image Processing. Dans Pearson (Éd.), *Image Enhancement in the Spatial Domain (section sur Laplacian Filtering)* (pp. 159–161 (dépend de l'édition)). Upper Saddle River, NJ: 4th Edition. doi:9780133356724

Ren, S., Kaiming He, Ross B. Girshick, & Jian Sun. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 39*, 1137–1149.

Ren, Shaoqing, He, Kaiming, Girshick, Ross, & Sun, Jian. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 91–99. Récupéré sur https://arxiv.org/abs/1506.01497

Shaoqing Ren, Kaiming He, Ross B. Girshick, & Jian Sun. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 39*, 1137–1149. doi:10.1109/TPAMI.2016.2577031

Smith, R. (2007). An overview of the Tesseract OCR engine. Dans R. Smith, *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)* (pp. 629–633). Curitiba: IEEE. doi:https://doi.org/10.1109/ICDAR.2007.4376991

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona,, Deva Ramanan, . . . C. Lawrence Zitnick. (2014). Microsoft COCO: Common Objects in Context. Dans S. Cham (Éd.), *European Conference on Computer Vision (ECCV), 8693*, pp. 740–755. Zurich, Switzerland.

Ultralytics. (2023). *YOLOv5 and YOLOv11 Documentation*. Retrieved from https://docs.ultralytics.com

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., . . . Polosukhin, Illia. (2017). Attention is all you need. *Advances in Neural*

Information Processing Systems (NeurIPS), *30*, 5998–6008. Récupéré sur https://arxiv.org/abs/1706.03762

Wei Liu, Dragomir Anguelov,, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, & Alexander C. Berg. (2016). SSD: Single Shot MultiBox Detector. Dans S. Cham (Éd.), *European Conference on Computer Vision (ECCV)*, *9905*, pp. 21–37. Amsterdam, The Netherlands.