Intelligent OCR Application for Text Extraction and Structuring on Online Platforms and Newspapers

Paulo Roberto Machado Silva Junior - a65416

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master's Degree in Electrical and Computers Engineering within the scope of the double degree program with the Federal University of Technology – Paraná.

Supervisors:

Prof. Dr. Paulo Alexandre Vara Alves

Prof. Dr. José Eduardo Fernandes

Prof. Dr. Márcio Rodrigues da Cunha

Bragança

February 2026

Intelligent OCR Application for Text Extraction and Structuring on Online Platforms and Newspapers

Paulo Roberto Machado Silva Junior - a65416

Dissertation presented to the School of Technology and Management in the scope of the Master in Electrical and Computers Engineering.

Supervisors:

Prof. Dr. Paulo Alexandre Vara Alves

Prof. Dr. José Eduardo Fernandes

Prof. Dr. Márcio Rodrigues da Cunha

Bragança

February 2026

Dedication

To my family, thank you for being my foundation. To my mother, Verônica Nascimento, thank you for your unconditional love, strength, and sacrifices. To my grandmother, Olicea Barroso, and my grandfather, Carlos Alberto, thank you for your care, wisdom, and for the values that shaped who I am. To my sister, Ana Julia, thank you for your advice and for always being there when I needed support. To my aunts, Angelica and Erli, thank you for your encouragement and support throughout this journey.

Acknowledgment

I would like to express my sincere gratitude to my supervisors in Portugal and Brazil, Professor Doutor Paulo Alexandre Vara Alves, Professor Doutor José Eduardo Fernandes, MSc. David Dias, MSc. Bruno Silva and Professor Doutor Márcio Rodrigues da Cunha, for their guidance and support throughout this thesis. Thank you for your availability and constructive feedback, which helped me. I am also grateful to the School of Technology and Management of Bragança (ESTiG) and the Polytechnic Institute of Bragança (IPB) for providing the academic environment that made this research possible. Additionally, I would like to thank Federal University of Technology – Paraná (UTFPR) – Campus Campo Mourão for the academic foundation that supported my development and helped prepare me for this journey.

I also want to thank my friends from Brazil who have been part of my path since graduation, especially the "Mansão dos Amigos" group, Renato Garcia and Lucas Pawelski, for the loyalty, laughter, and long distance support. A special thank you to Vinicius Sasaki for accepting to come with me to this new experience in Portugal, your presence made the transition easier and turned many difficult days into good memories.

I am thankful for the friends that Bragança gave me, especially Lucas Paula, for the support and shared routines that made this experience in Portugal feel lighter and more meaningful.

Finally, I extend my gratitude to everyone who contributed in any way, through encouragement, conversations, or support behind the scenes. This thesis carries a part of each person who stood by me, and I am truly thankful for all of you.

Abstract

The monitoring of print media is a important function for the advertising industry, enabling the identification of advertisements in newspapers and magazines for market analysis. However, automating this extraction is challenging due to the complex layouts of these publications. Conventional Optical Character Recognition (OCR) systems, capable of transcribing individual characters, often fail to retain structural organization and logical reading order.

To address these issues, the proposed process integrates Document Layout Analysis (DLA) with OCR in a multi-stage process. YOLOv10 and YOLOv12 models detect and segment document elements, and the resulting regions are then passed to PaddleOCR for text extraction.

Experimental results show that the first pre-trained model achieved a mAP@50 of 0.728 on a 2,000 images sample from DocLayNet. The second pre-trained model achieved a mAP@50 of 0.519 on a custom dataset. The fusion strategy reduced detection redundancy, and comparative evaluation against a production baseline indicates competitive performance. The final workflow produces a semi-structured JSON output that preserves the association between bounding box coordinates and extracted text. Future work will assess Vision Language Models (VLMs) to improve reading order reconstruction in more complex layouts.

Resumo

O acompanhamento de texto impressos é uma função importante para a indústria da publicidade, permitindo a identificação de anúncios em jornais e revistas para análise de mercado. No entanto, a automatização desta extração é desafiante devido aos layouts complexos destas publicações. Os sistemas convencionais de Reconhecimento Ótico de Caracteres (OCR), embora capazes de transcrever caracteres individuais, falham em preservar a organização estrutural e a ordem lógica de leitura.

Para contornar estes desafios, o processo proposto integra a Análise de Layout de Documentos (DLA) com OCR num processo de multietapas. Os modelos YOLOv10 e YOLOv12 detetam e segmentam elementos do documento, e as regiões resultantes são depois encaminhadas para o PaddleOCR para extração de texto.

Os resultados experimentais mostram que o primeiro modelo pré-treinado alcançou um mAP@50 de 0,728 numa amostra de 2.000 imagens do DocLayNet. O segundo modelo pré-treinado obteve um mAP@50 de 0,519 num conjunto de dados personalizado. A estratégia de fusão entre os resultados dos modelos reduziu a redundância de deteção e uma avaliação comparativa com uma abordagem atual em produção indica desempenho competitivo. O fluxo de trabalho final produz uma saída JSON semiestruturada que preserva a associação entre as coordenadas das caixas delimitadoras e o texto extraído. Trabalhos futuros sugerem avaliar Modelos de Visão e Linguagem (VLMs) para melhorar a reconstrução da ordem de leitura em layouts mais complexos.

Contents

List of Figures

Acronyms

AI Artificial Inteligence.

AP Average Precision.

API Application Programming Interface.

C2PSA Convolutional block with Parallel Spatial Attention.

CER Character Error Rate.

CNN Convolutional Neural Network.

COCO Common Objects in Context.

CRNN Convolutional Recurrent Neural Network.

CSPDarknet53 Cross Stage Partial Darknet-53.

CSPNet Cross Stage Partial Network.

CTC Connectionist Temporal Classification.

DB Database.

DBMS Database Management System.

DI Document Intelligence.

DLA Document Layout Analysis.

DNN Deep Neural Network.

ELAN Efficient Layer Aggregation Networks.

ESTiG Escola Superior de Tecnologia e Gestão.

FN False Negative.

FP False Positive.

GELAN Gradient-Efficient Layer Aggregation Network.

GPU Graphics Processing Units.

HTTP HyperText Transfer Protocol.

IoU Intersection over Union.

IPB Instituto Politécnico de Bragança.

LLM Large Language Model.

LSTM Long Short-Term Memory.

mAP Mean Average Precision.

NMS Non-Maximum Suppression.

OCR Optical Character Recognition.

PAN Path Aggregation Network.

PAN-FPN Path Aggregation Network - Feature Pyramid Network.

PANet Path Aggregation Network.

PGI Programmable Gradient Information.

PMCOA PubMed Central Open Access.

PSA Partial Self-Attention.

QR code Quick Response code.

R-CNN Region-based Convolutional Neural Network.

R-ELAN Residual Efficient Layer Aggregation Network.

RDBMS Relational Database Management System.

ReLU Rectified Linear Unit.

RNN Recurrent Neural Network.

RPN Region Proposal Network.

TP True Positive.

VLM Vision Language Model.

WER Word Error Rate.

YOLO You Only Look Once.

Chapter 1

Introduction

Digitalization has transformed how information is archived and accessed. While computers can easily read digital files, a lot of information is still in pictures of documents, such as scanned newspapers, magazines, and old records. To make such content searchable and use this information, it must be converted into a structured representation that machines can process and Optical Character Recognition (OCR) has been used for this purpose. The OCR alone cannot provide sufficient performance for complex layouts, OCR can transcribe characters, but it typically does not model document structure, and the resulting output may appear as disordered text that fails to preserve the organization of the original page.

This dissertation addresses the extraction of information from digital documents with diverse layouts, with a focus on newspapers and online magazines characterized by differents designs. The proposed approach plain a process by integrating deep learning, DLA, with OCR. By treating layout elements as visual objects to detect and segment, and then associating each region with its recognized text, the system can produce an output that combines structural organization with textual content.

1.1 Motivation

The digitalization presents challenges due to their complex design. These publications utilize multi-column layouts. Traditional approaches to information extraction often treat documents as stream of data. When applied to these complex layouts, some text extraction systems, OCR, process pages in a simple top-to-bottom and left-to-right direction. Consequently, sentences from adjacent columns become mixed together, creating an unformatted stream of data that compromises context.

To preserve document readability and make market analysis practicable, character transcription must be combined with layout analysis. Historically, this required manual segmentation or rule based on heuristics, which are often difficult to generalize across different document layout.

Recent developments in the field of Artificial Intelligence, specifically in Computer Vision and Deep Learning, offer new ways to address these limitations. Object detection architectures, such as the YOLO family, have demonstrated the ability to identify and segment visual elements within an image, while OCR models have improved in recognizing text.

However, the integration of these technologies for document structuring keep the challenge. The detection and recognition models is insufficient without post-processing method to the logical reading order of newspapers. This project proposes a solution that understand the document structure from image inputs, associating each detected visual region with its corresponding text content.

1.2 Objectives

The objective of this dissertation is to develop a OCR application capable of extracting, segmenting, and matching document structure with the corresponding text from newspapers with diverse layouts. The system outputs a semi-structured representation that preserves the logical reading order and semantic organization of the document. To

achieve this, the project is divided into the following specific goals:

- **Study and analysis of state-of-the-art technologies:** Review deep learning approaches for DLA and OCR.
- **Development of a layout detection module:** Implement an object detection module capable of identifying and segmenting document elements.
- **Integration of text extraction with structural analysis:** Integrate the layout detection module with the OCR engine to extract the corresponding text for each detected element.
- **Development of reading order heuristics:** Design post-processing algorithms that approximate human reading order.
- **Structured output generation:** Generate an output format that preserves the relationship between textual content and visual class labels.

1.3 Structure of the Dissertation

This dissertation is organized in five chapters that detail the research, development, and evaluation of the proposed solution. The structure is defined as follows:

Chapter 1 - Introduction: Shows the context of the research, identifying the challenges with extracting information from documents with complex layouts.

Chapter 2 - State of the Art: Gives the theoretical foundation and a review of existing literature. This section bring the evolution of DLA from traditional heuristic methods to modern deep learning approaches.

Chapter 3 - System Architecture and Development: Describes the technical environment used, technologies for the project and explains the proposed solution.

Chapter 4 - Results: Presents the experimental evaluation and analysis of the system. It compares training metrics across datasets and shows a comparative against an existing production system.

Chapter 5 - Conclusion and Future Work: Makes a summarization of the contributions of the dissertation and proposes future works.

Chapter 2

State of the Art

This chapter contextualizes the project's state of the art by providing an overview of how related projects have been developed. It reviews computer vision tools, the application of layout detection in different domains, and techniques for OCR.

2.1 Document Layout Analysis (DLA)

The DLA is a important component of Document Intelligence (DI), with focus in analyzing the spatial arrangement of elements in a document image, such as tables, text, and images. It can address both structured and unstructured document formats [1].

The scope of DLA comprises two dimensions: Physical Layout Analysis, which detects and classifies page elements such as text, images, and tables, and Logical Layout Analysis, which identifies semantic components including titles, headers, and paragraphs [1].

To support and streamline a range of downstream tasks, the DLA is an essential approach for several reasons: it enhances the efficiency of document processing, particularly in content extraction and comprehension [2]; it transforms documents into structured or semi-structured formats necessary for identifying key information; and it supports applications such as document digitization and content summarization [1].

The digitalization of documents has traditionally depended on OCR to convert images into text readable by machine [3]. However, OCR alone faces limitations when characters

appear noisy or fragmented, when layouts are complex, or when the source material is of poor quality [4].

The OCR can generated erros and it can propagate through subsequent processing stages, reducing overall system performance. To identify and address this issue is requires more advanced methods for text recognition. To solve this need, DLA has emerged as a valuable approach for mitigating errors related to OCR [1].

Over past the years, various strategies have been proposed, including heuristic, based in rule algorithms (top-down, bottom-up, and hybrid). With the advent of CNN, document analysis saw marked improvements [5]. More recently, architectures based in transformer have become prevalent, owing to their global attention mechanisms and integration of Non-Maximum Suppression (NMS) [6].

2.1.1 Evolution of Approaches: From Classical Methods to Deep Learning

For converting scanned documents into searchable, interpretable formats and for streamlining information retrieval and data extraction, DLA plays a central role [6]. It provides the foundation for document processing and content categorization.

The evolution of DLA, particularly in layout analysis methods, can be categorized into two primary approaches: traditional and deep learning based.

2.1.1.1 Traditional Approaches

Before deep learning gained importance, heuristic algorithms formed the foundation of layout analysis. Accordingly, various based rule approaches can be classified as bottom-up, top-down, or hybrid, depending on their sorting criteria [7].

The bottom-up approach involves fundamental processes such as pixel grouping and clustering to form uniform regions for similar objects while separating dissimilar ones [1]. The method utilizes connected component analysis, which groups pixels or low level features into regions according to their connectivity [4]. As shown in Figure 2.1, the

examples in [8] illustrate an input image and its bottom-up segmentations at four scales (from fine to coarse), with each segment rendered in a distinct color and outlined in black.

Figure 2.1: Bottom-up segmentation at multiple scales [8], p. 7.

In contrast, the top-down approach recursively partitions a document into regions until homogeneous areas are isolated [1]. For instance, the authors [7] implement this by iteratively cutting the image along whitespace or boundary lines. Although this method is straightforward and fast to implement, its applicability is limited to documents with regular layouts.

The Figure 2.2 summarizes page segmentation techniques, where Hybrid methods combine bottom-up and top-down strategies to balance accuracy and adaptability.

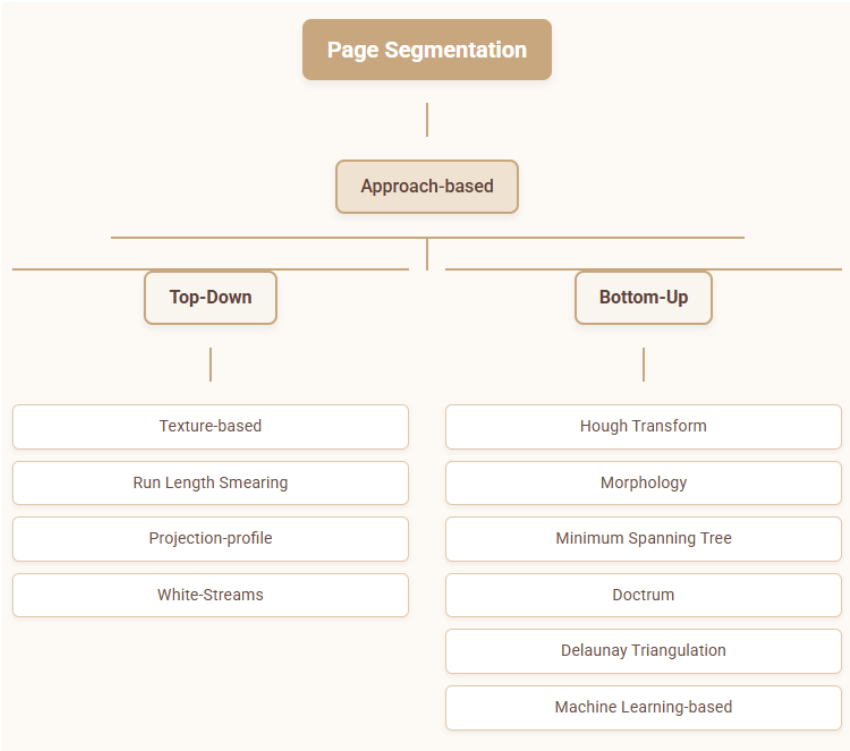


Figure 2.2: Comparison of page-segmentation methods [9], p. 11.

The Traditional heuristic approaches often produce limited results. Because they cannot accommodate intricate layouts, their use remains restricted to simpler document types, leading to weak performance on heterogeneous datasets. With the advent of deep learning, however, new methods have emerged that can handle complex contexts [10].

2.1.1.2 Deep Learning Based Approaches

In the work of [2], the authors report an increase in document digitalization. Consequently, emphasis has shifted from physical paper to digital formats, creating a growing demand for approaches based in deep learning.

Modern algorithmic methods provide techniques for extracting insights from documents. Understanding document content is important for converting data into value in applications such as machine learning and image processing [2].

The document layout analysis field has undergone a transformation with the rise of deep learning techniques [11]. This transformation was driven by:

- advancements in computational power and technology: Deep Neural Network (DNN), particularly CNN, achieved higher accuracy and faster processing than traditional methods in capturing complex document layouts [12] .
- the availability of large datasets: deep learning models require substantial training data to perform effectively [11]. Notable contributions include PubLayNet [13], DocLayNet [14], and DocBank [15].

The PubLayNet dataset consists of 1M images from PubMed Central Open Access (PMCOA), where it was annotated with five classes, such as: Text, Title, List, Table, and Figure. Being sourced from academic articles, the dataset is destinated to a specific domain and shows limited diversity in page structure and design [13]. An example image from the dataset is shown in Figure 2.3.

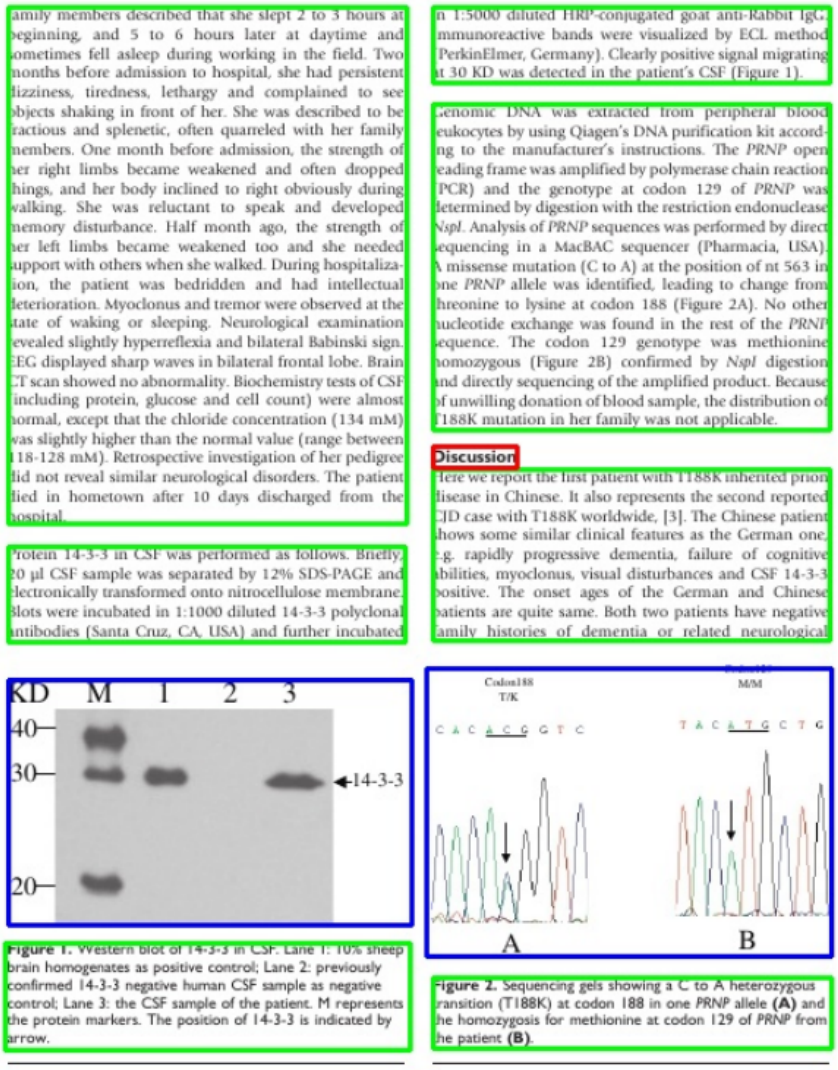


Figure 2.3: Example page with bounding-box annotations by PubLayNet [13], p. 6.

DocLayNet, a example of this dataset is shown in Figure 2.4, is a manually annotated document layout dataset in Common Objects in Context (COCO) format, containing data from diverse sources. It defines eleven classes: Caption, Footnote, Formula, List-item, Page-footer, Page, Picture, Section-header, Table, Text, and Title [14].

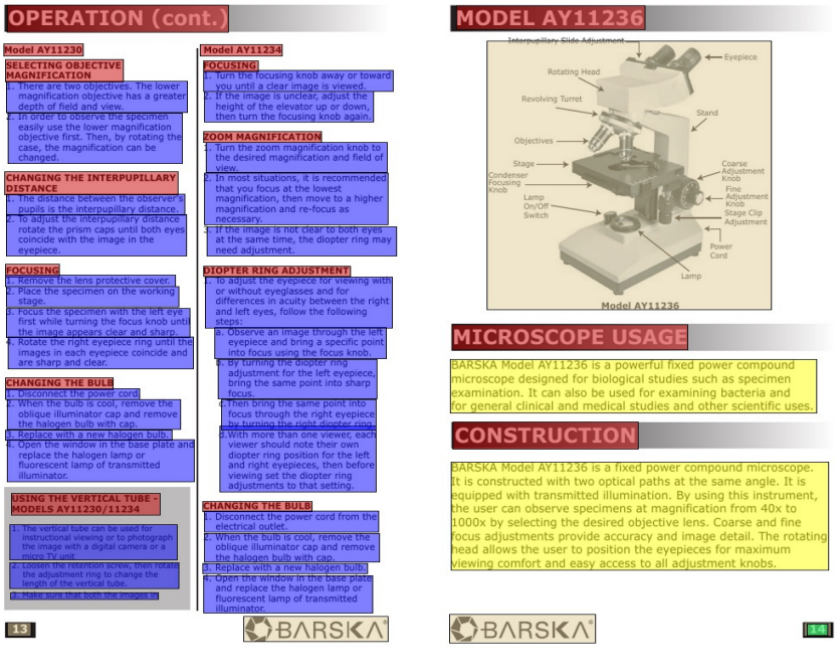


Figure 2.4: Example page with bounding-box annotations by DocLayNet [14], p. 9.B.

The authors of [15] introduce DocBank, a dataset of 500K English document images. This dataset, shown in Figure 2.5, is based on automatically annotated documents, which limits its generalizability.

Figure 2.5: Example page with bounding-box annotations by DocBank [15], p. 2.

The DocBank comprises thirteen classes: Abstract, Author, Caption, Date, Equation, Figure, Footer, List, Paragraph, Reference, Section, Table, and Title [15].

2.1.2 Layout Detection Models

In the domain of DLA, detection models function by treating distinct document components, such as headings, paragraphs, and images, as visual objects to be localized and classified. This approach leverages architectures originally designed for general object detection, adapting them to recognize the specific structural elements of a page. This approach involve architectures originally designed for general object detection, CNN.

Designed for processing data with a grid like topology, CNNs distinguish themselves from traditional neural networks by automatically learning spatial hierarchies of features. This makes them a good choice for the pattern acknowledgment and computer vision tasks required to recognize specific structural elements on a page [16].

The following subsections examine the evolution of these architectures, from foundational CNN based models to the real-time performance of the YOLO family.

2.1.2.1 Models based on CNN

The CNN have a foundational role in the context of DLA, especially as the field transitions into deep learning approaches [1]. With advancements in computational power, the availability of large datasets, and the widespread adoption of the Rectified Linear Unit (ReLU) activation function, CNNs have become the principal method for working within the DLA domain, significantly improving both precision and efficiency [11].

Originally designed for generic object detection, models based in CNN have been adapted to the DLA context by treating document layout elements as visual objects within an image. In this way, DLA involves localizing and detecting components such as headings, text blocks, and images, much like a CNN would detect cars, animals, and other objects [17].

CNNs are a class of DNNs distinguished by their capacity to automatically learn

hierarchical features. They were originally developed for image processing tasks [18]. The core architecture of a CNN comprises convolutional, pooling, and activation layers that progressively convert the input into higher level representations [19]. The Figure 2.6 and Figure 2.7 illustrate the main components of a CNN.

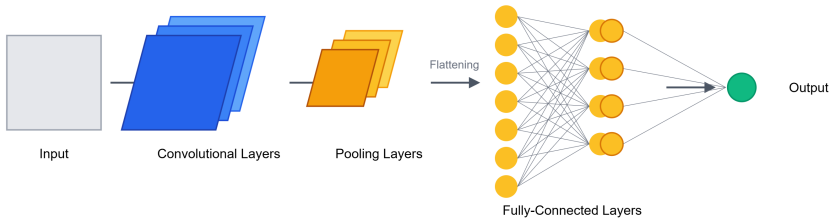


Figure 2.6: The general structure of a CNN. Adapted from [19], p. 355.

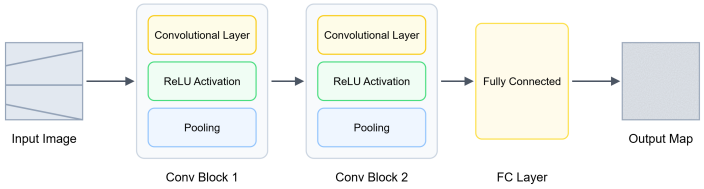


Figure 2.7: A example of proposed abstract architecture. Adapted from [19], p. 358.

An important design aspect of a CNN is the definition of the number of kernels and their respective spatial sizes. During the convolution process, each kernel is element-wise multiplied with a corresponding patch of the input, followed by summation [19]. Mathematically, this feature extraction process is described by Equation 2.1:

$$\text{Output}[i,j] = \sum_{u=0}^{k_h-1} \sum_{v=0}^{k_w-1} \text{Input}[i+u, j+v] \cdot \text{Kernel}[u,v] \tag{2.1}$$

In Equation 2.1, the variables are defined as follows: $Output[i,j]$, the element at position (i,j) in the output feature map; $Input[i+u, j+v]$, the input element at position

$(i+u, j+v)$; $Kernel[u,v]$ is the kernel weight at position (u,v); k_h is kernel height; k_w is kernel width.

According to [20], many pooling strategies can be applied during convolution, such as average pooling, sum pooling, and most commonly, max pooling. The max pooling operation for a one dimensional input is defined by Equation 2.2:

$$a_x^l = \max\left(a_{(x-y)}^{(l-1)}, a_{(x+y)}^{(l-1)}\right) \tag{2.2}$$

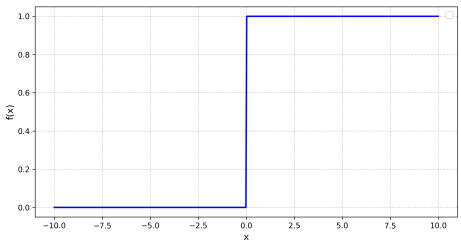
Within the CNN layer, the ReLU activation function Equation 2.3 has become the standard due to its computational simplicity: it outputs $\max(0,x)$, making it faster to compute than alternatives like the sigmoid and tanh functions, shown in Equation 2.4 and Equation 2.5, respectively:

$$f(x) = \max(0,x) \tag{2.3}$$

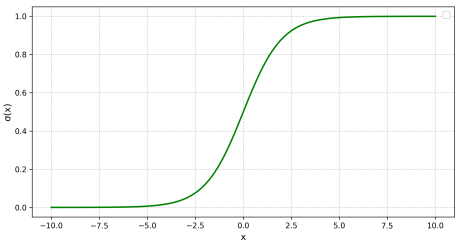
$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

$$\tanh(x) = 2\sigma(2x) - 1 \tag{2.5}$$

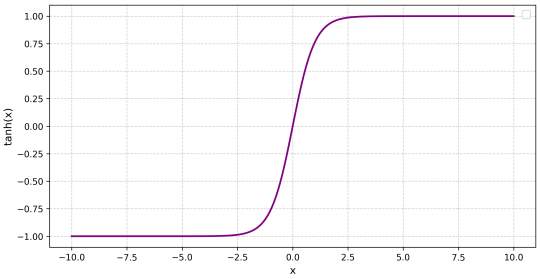
The Step, Sigmoid, and Tanh functions are also commonly used for activation, which serve as the foundation for numerous other variants [21]. The Figure 2.8 shows, respectively, the graphics of those activation functions.



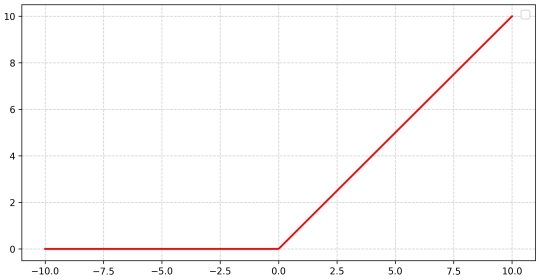
(a) Step activation function.



(b) Sigmoid activation function.



(c) Tanh activation function.



(d) ReLU activation function.

Figure 2.8: Activation functions. Adapted from [22].

After many convolutional and pooling stages, the extracted features are flattened into a one dimensional vector and passed to fully connected layers. These layers support the classification process by combining and refining the learned features. Each neuron in one layer is connected to every neuron in the next, what enable a integration of features for effective classification [3].

Responsible for extracting multiple features from input document images, the backbone network, shown in Figure 2.9, is the initial part of a deep learning model. These features are then passed on to the rest of the network, which uses them to perform tasks like object detection or classification [1], [19].

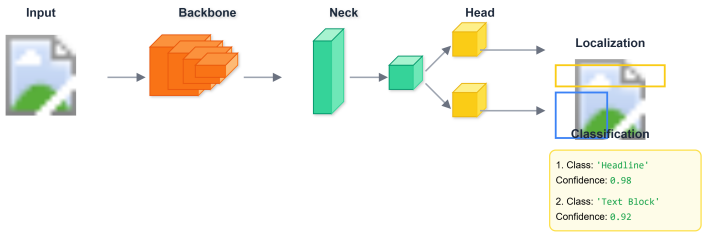


Figure 2.9: A standard architecture of object detectors. Adapted from [19], p. 357.

The main purpose of using a backbone network is to extract information from input images. It processes the input to generate a sequence of feature maps at a reduced resolution. These feature maps are refined for each instance using 1×1 convolution layers, which helps manage the number of trainable parameters [23].

The performance of DLA depends on its backbone. While various architectures can be used for this purpose, the ResNet family, such as ResNet-18, ResNet-50, and ResNet-101 has been the most commonly chosen due to its proven deep residual learning framework that produces feature hierarchies [2].

Backbone architectures based on Vision Transformers have gained attention. In particular, the Swin Transformer is frequently used due to its performance in handling complex dependencies and hierarchical features [24].

Models such as R-CNN, Fast R-CNN, and Faster R-CNN, referred to as two-stage detectors, are used to first propose candidate regions in an image, then classify and refine the bounding boxes within these regions [25]. The output of each model is typically defined by bounding boxes that delineate the detected layout elements [18].

By integrating a Region Proposal Network (RPN), Faster Region-based Convolutional Neural Network (R-CNN) significantly improved upon its predecessors by generating region proposals that are then classified and refined through Fast R-CNN's detector and bounding box regression [18].

These object detection based in CNN models present some fundamental limitations

when applied to DLA. Although they are only visual, which meaning, they operate on pixel level information, they do not incorporate the processing or understanding of actual text content [12] [26]. Additionally, they often perform poorly in accurately localizing small scale text regions [27]. Detectors with two stage are typically demand more computational resources than single stage alternatives, which may restrict their applicability [17].

2.1.2.2 The YOLO Family

One of the most famous deep learning methods is the YOLO approach, in which object detection models have been adapted for DLA by treating different document components as visual objects [12]. First introduced in 2015, as show in Figure 2.10, this approach is considered a real-time object detection method and has demonstrated superior performance compared to other object detection approaches [28].

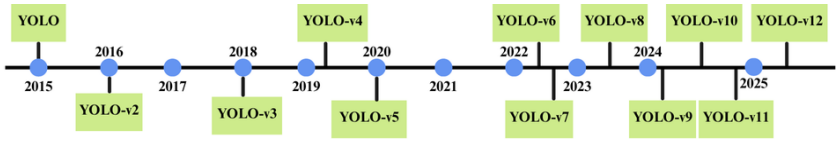


Figure 2.10: Evolution of the YOLO family. Adapted from [25], p. 1367.

The YOLO approach is known as a single stage detector, which means that object identification requires only one iteration to predict bounding box locations and assign the corresponding labels [25]. This method established a new state of the art by achieving relevant gains in accuracy and computational efficiency when evaluated on public available benchmark datasets [29].

The YOLO series has evolved considerably, marked by changes in backbone architectures, loss functions, and detection mechanisms. The YOLOv1, The first model, employed a 24-layer network [28]. The YOLOv2 introduced some innovations, including batch normalization and anchor boxes sized through dimension clustering [30]. YOLOv3 additionally improved performance by adopting the Darknet-53 backbone with residual connections [31].

The evolution continued with YOLOv4, which adopted the Cross Stage Partial Darknet-53 (CSPDarknet53) backbone and a feature aggregation neck combining Path Aggregation Network (PANet) with Spatial Pyramid Pooling [32]. YOLOv5 enhanced inference speed and usability through its PyTorch implementation and a new generation of across stage partial connections. YOLOv6 preserved these gains while aiming to reduce computational overhead. YOLOv7 advanced the series with novel training strategies, including auxiliary and lead heads, to reach state-of-the-art accuracy [33] [34].

The YOLO family has focused on solving the bottlenecks in deep learning and real time detection. YOLOv8, for instance, established a versatile foundation with its multi-task platform for detection, segmentation, and tracking, which uses a CSPDarknet53 backbone and a Path Aggregation Network - Feature Pyramid Network (PAN-FPN) neck. YOLOv9 improved accuracy by addressing information loss through techniques such as Programmable Gradient Information (PGI) and Gradient-Efficient Layer Aggregation Network (GELAN). The YOLOv10 introduced an NMS-free approach to deal with the post-processing bottleneck that limits the speed of inference, which eliminated a key computational constraint and improving the object detection [35].

YOLOv11 advanced the series by introducing the C3k2 block and the Convolutional block with Parallel Spatial Attention (C2PSA) module, achieving higher accuracy and efficiency with a more favorable performance speed trade-off than YOLOv9 and YOLOv10. Its successor, YOLOv12, adopted an attention centric design with Area Attention and Residual Efficient Layer Aggregation Network (R-ELAN) that, despite its intent, resulted in improvements and increased computational overhead, making it difficult to balance speed and accuracy [36].

2.1.2.3 The YOLO Architecture

As mentioned in 2.1.2.2, the YOLOv10 was developed to improve real-time object detection by eliminate the need for NMS, thereby reducing post-processing latency. YOLOv12 introduces an attention centric design (Area Attention, R-ELAN) and a unified detection framework for multi-task learning, which together improve speed and accuracy.

The core idea of YOLOv10's architecture is the Consistent Dual Assignments training strategy, which provides effective supervision while removing the computationally costly NMS post-processing step [37].

YOLOv10 achieves its performance, shown in Figure 2.11, through a combination of training strategies, architectural changes, and multiple model variants. However, this design may have slow convergence and reduce accuracy, as it can lead to weak supervision[38]. A one-to-many assignment scheme, illustrated in Figure 2.12, can alleviate this limitation by compensating for the reduced supervision [39].

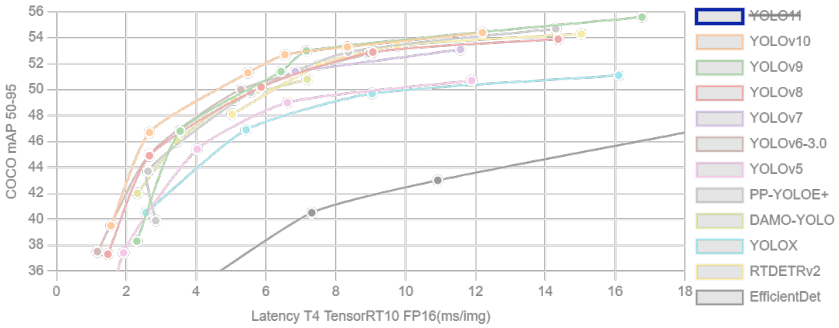


Figure 2.11: YOLOv10 Performance [37].

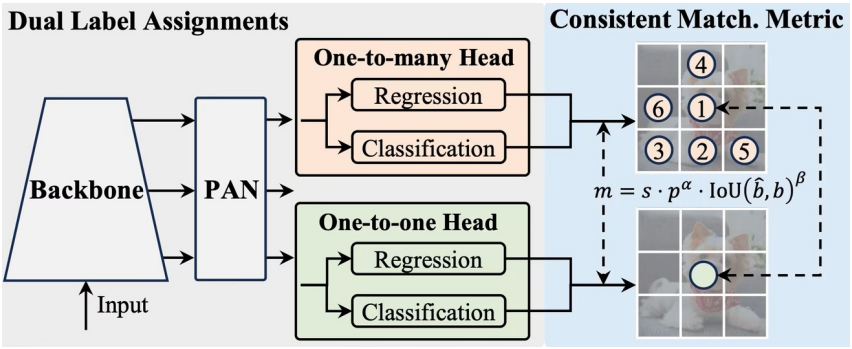


Figure 2.12: Consistent dual assignments [37].

To enable end-to-end deployment, a secondary head is introduced, identical in structure to the one-to-many branch but using one-to-one matching for label assignments. The two heads are trained together, which allows the backbone to receive supervision from the one-to-many branch. During inference, however, the one-to-many head is discarded, and only the one-to-one head is used, resulting in no additional inference cost [37].

The YOLOv10 architecture is organized into three principal components: the backbone, the neck, and the detection head. The backbone is constructed using an enhanced Cross Stage Partial Network (CSPNet) that is designed to improve gradient flow and minimize computational redundancy, while the neck integrates Path Aggregation Network (PAN) layers to make multi-scale feature fusion easier [36].

YOLOv10 integrates a series of design trade-offs focused on maximizing performance while reducing computational overhead:

- **Lightweight Classification Head:** This component consists of two depthwise separable convolutions (3Ö3 kernel size) followed by a 1Ö1 convolution, reducing computational cost without significantly compromising accuracy [40].
- **Spatial Channel Decoupled Downsampling:** This technique reduces model size while maintaining accuracy, making it an efficient choice for resource constrained environments [37].
- **Rank Guided Block Design:** YOLOv10 employs a rank guided block design to create a compact structural configuration [25].

With a balanced between speed and precision, YOLOv10 has been benchmarked on standard datasets such as COCO. The model achieved accuracy while reducing latency, outperforming its predecessors and other modern detectors, as shown in Figure 2.13.

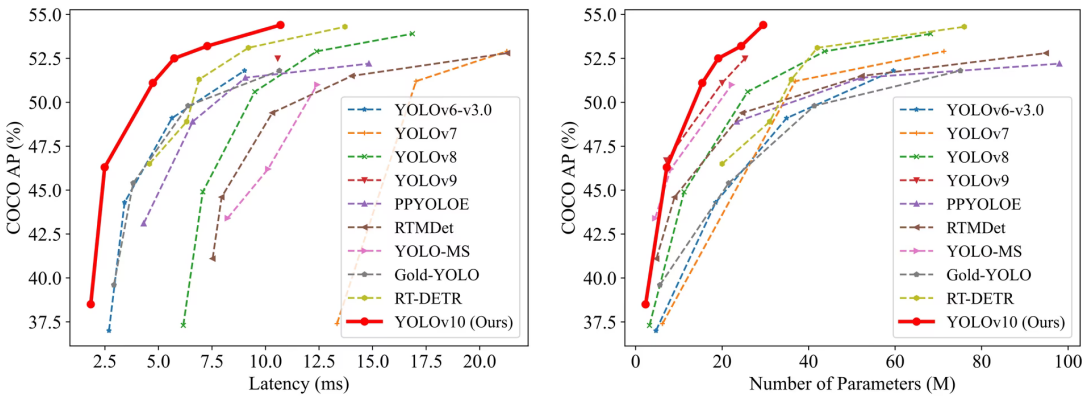


Figure 2.13: YOLOv10 latency and number of parameters [37].

YOLOv10's advancements are evident when measured against other top models. Throught the Table 2.1, compared to RT-DETR-R18, YOLOv10-S is faster for the same accuracy.

Model	**Param. (M)**	**FLOPs (G)**	**AP^{val}**(%)	**Latency (ms)**	**Latency^{f} (ms)**
YOLOv6-3.0-N	4.7	11.4	37.0	2.69	1.76
Gold-YOLO-N	5.6	12.1	39.6	2.92	1.82
YOLOv8-N	3.2	8.7	37.3	6.16	1.77
YOLOv10-N	**2.3**	**6.7**	**39.5**	**1.84**	**1.79**
YOLOv6-3.0-S	18.5	45.3	44.3	3.42	2.35
Gold-YOLO-S	21.5	46.0	45.4	3.82	2.73
YOLO-MS-XS	4.5	17.4	43.4	2.83	1.45
YOLO-MS-S	8.1	32.2	46.2	3.42	1.78
YOLOv8-S	11.2	28.6	44.9	7.07	2.33
YOLOv9-S	7.1	26.4	46.7	-	-
RT-DETR-R18	20.0	60.0	46.5	4.58	4.49
YOLOv10-S	**7.2**	**21.6**	**46.8**	**2.49**	**2.39**
YOLOv6-3.0-M	34.9	85.8	49.1	5.63	4.56
Gold-YOLO-M	41.3	87.5	49.8	6.38	5.45
YOLO-MS	22.2	70.2	51.0	12.41	7.30
YOLOv8-M	25.9	78.9	50.6	9.50	5.09
YOLOv9-M	20.0	70.6	51.1	-	-
RT-DETR-R34	31.0	92.0	48.9	6.32	6.21
RT-DETR-R50m	36.0	113.1	51.3	6.90	6.84
YOLOv10-M	**15.4**	**59.1**	**51.3**	**4.74**	**4.63**
YOLOv6-3.0-L	59.6	150.7	51.8	9.02	7.90
Gold-YOLO-L	75.1	151.7	51.8	10.65	9.78
YOLOv9-C	25.3	96.2	51.8	10.57	6.13
YOLOv10-B	**19.1**	**92.0**	**52.7**	**5.74**	**5.67**
YOLOv8-L	43.7	165.2	52.9	12.39	8.06
RT-DETR-R50	42.0	136.0	53.1	9.20	9.07
YOLOv10-L	**24.4**	**120.3**	**53.4**	**7.28**	**7.21**
YOLOv8-X	68.2	257.8	53.9	16.86	12.83
RT-DETR-R101	76.0	259.0	54.3	13.71	13.58
YOLOv10-X	**29.5**	**160.4**	**54.4**	**10.70**	**10.60**

Table 2.1: Comparison of YOLO variants and baseline object detectors. Adapted from [37].

Although YOLOv10 demonstrated good performance, the more recent YOLOv12 was introduced as an attention-centric real-time object detector. This version uses a different modeling capacity of attention mechanisms while preserving the speed typical of models based in CNN [41]. Its anchor free design removes the complexity of traditional anchor

boxes, leading to shorter inference times [42].

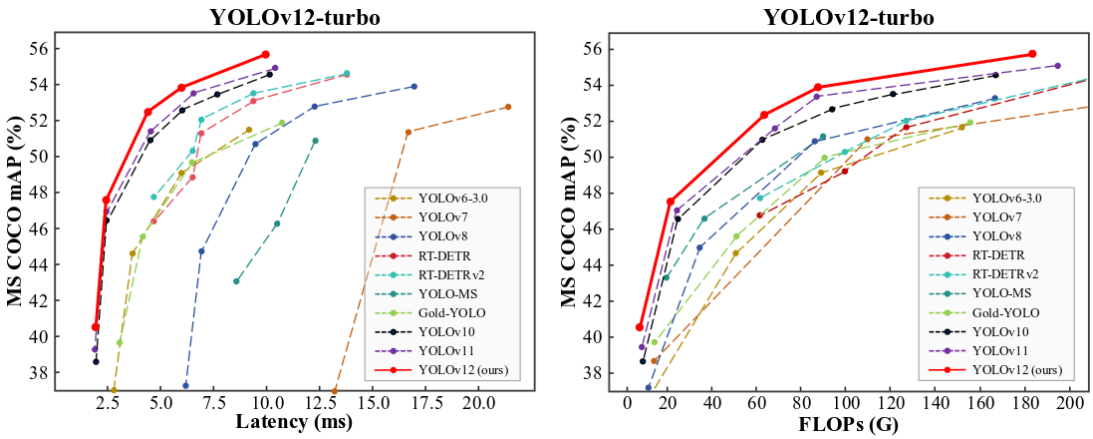


Figure 2.14: Comparison with popular methods in terms of latency accuracy and FLOPs-accuracy [41].

Within the YOLO framework, the YOLOv12 adopts an integrated architecture that departs from conventional designs based in CNN. As detailed in [41], this integration introduces the following innovations:

- A simple, efficient, and attention-centric architecture that moves beyond the conventional CNN structure used in previous YOLO models.
- State-of-the-art performance in both detection accuracy and inference speed, as show in Table 2.2.

Method	FLOPs (G)	#Param. (M)	$AP^{val}_{50:95}$ (%)	AP^{val}_{50} (%)	AP^{val}_{75} (%)	Latency (ms)
YOLOv6-3.0-N	11.4	4.7	37.0	52.7	-	2.69
Gold-YOLO-N	12.1	5.6	39.6	55.7	-	2.47
YOLOv8-N	8.7	3.2	37.6	52.6	40.5	1.77
YOLOv10-N	6.5	2.1	38.5	53.8	43.1	1.84
YOLO11-N	6.5	2.6	39.4	55.2	-	1.74
YOLOv12-N	**6.5**	**2.6**	**40.6**	**56.7**	**43.8**	**1.64**
YOLOv6-3.0-S	45.3	18.5	44.3	61.2	-	3.42
Gold-YOLO-S	46.0	21.5	45.4	62.5	-	3.82
YOLOv8-S	28.6	11.2	43.6	60.2	48.7	2.33
RT-DETR-R18	60.0	20.0	47.9	64.3	-	4.58
RT-DETRv2-R18	60.0	20.0	47.9	64.3	-	4.58
YOLOv9-S	26.4	7.1	46.4	60.7	50.7	2.47
YOLOv10-S	21.6	7.2	46.3	63.0	50.4	2.49
YOLO11-S	21.5	9.2	47.5	63.9	-	2.53
YOLOv12-S	**21.4**	**9.3**	**48.0**	**65.0**	**51.8**	**2.61**
YOLOv6-3.0-M	85.8	34.9	49.1	66.1	-	5.63
Gold-YOLO-M	93.3	41.3	49.8	67.0	-	6.38
YOLOv8-M	78.9	25.9	50.3	67.2	54.7	5.09
RT-DETR-R34	100.0	36.0	51.0	66.8	-	6.32
RT-DETRv2-R34	100.0	36.0	51.0	67.5	-	6.32
YOLOv9-M	76.3	20.0	51.4	68.1	56.1	4.52
YOLOv10-M	51.5	20.1	51.6	67.6	55.6	4.74
YOLO11-M	68.0	20.1	51.5	68.5	55.7	4.76
YOLOv12-M	**67.5**	**20.2**	**52.5**	**69.6**	**57.1**	**4.86**
YOLOv6-3.0-L	150.7	59.6	51.8	69.2	-	9.02
Gold-YOLO-L	151.7	75.1	51.8	68.9	-	10.65
YOLOv8-L	132.0	43.7	53.0	69.8	57.7	8.70
RT-DETR-R50	165.2	42.0	53.1	71.3	-	6.90
RT-DETRv2-R50	165.2	42.0	53.1	71.6	-	6.90
YOLOv10-B	92.0	19.1	52.5	69.6	57.2	5.74
YOLO11-L	88.9	25.3	53.0	70.1	58.2	6.2
YOLOv12-L	**88.9**	**26.4**	**53.7**	**70.7**	**58.5**	**6.77**
YOLOv8-X	257.8	68.2	54.0	71.0	58.8	12.83
RT-DETR-R101	259.0	76.0	54.3	72.7	-	13.5
RT-DETRv2-R101	259.0	76.0	54.3	72.7	-	13.5
YOLOv10-X	160.4	36.1	54.3	72.8	59.1	10.70
YOLO11-X	199.0	56.9	54.6	71.5	-	11.79
YOLOv12-X	**199.0**	**59.1**	**55.2**	**72.0**	**60.2**	**11.79**

Table 2.2: Performance comparison of object detectors on 640×640 input images. Adapted from [41].

YOLOv12 incorporates R-ELAN to enhance feature aggregation within the network, as shown in Figure 2.15 [41]. As shown in Table 2.3, the architecture is divided into three main components.

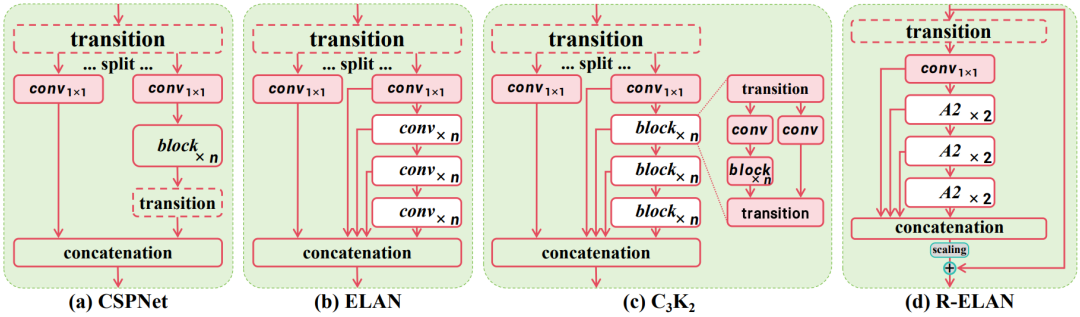


Figure 2.15: Architecture comparison with popular modules [41].

Architectures such as CSPNet and Efficient Layer Aggregation Networks (ELAN) are designed for feature aggregation by splitting feature paths, processing them independently, and concatenating the outputs. However, the standard ELAN design shown in Figure 2.15(b) causes training instability and gradient blocking when combined with the attention mechanisms used in YOLOv12. These issues have hindered the convergence of larger models. The issues were attributed to residual connections and the aggregation flow of features, which became bottlenecks at larger scales [41].

To address these limitations, the framework YOLOv12 introduces R-ELAN, shown in Figure 2.15(d), with two main modifications. First, a residual shortcut from the block's input to its output, inspired by Layer Scaling [43], stabilizes gradient flow. Second, the data flow is reorganized into a structured bottleneck, processing a single feature map instead of multiple splits. These modifications reduce training instability and improve efficiency by lowering computational cost, parameter count, and memory usage, thereby adds more sustaining to YOLOv12's balance between speed and accuracy.

Component	Functionality	Innovations in YOLOv12
Backbone	Serves as the primary feature extractor, processing input images to capture multi-scale visual information.	Employs R-ELAN to enhance residual learning connections. Uses 7×7 separable convolutions for spatial context with a reduced parameter count.
Neck	Fuses feature maps from various backbone stages.	Integrates area based attention mechanisms using FlashAttention, enabling the model prioritize salient image regions.
Head	Acts as the final output layer, responsible for predicting object bounding boxes, scores, and class probabilities.	Features improved prediction paths for object detection across multiple scales. Incorporates loss functions for real-time performance.

Table 2.3: An overview of the core architectural components in YOLOv12 and their key innovations. Adapted from [44].

The YOLO algorithm begins by dividing the input image into a uniform grid of size $S \times S$. A practical example of this is the 8×8 cell division shown in Figure 2.16a. The detection principle is that the grid cell containing the center of an object is responsible for predicting that object's bounding box coordinates [28].

Each grid cell is responsible for producing $\mathcal{B}$ bounding boxes. A single bounding box prediction includes coordinates relative to its parent cell and an associated confidence score (ranging from 0.0 to 1.0). This score reflects both the model's confidence that the box contains an object and the accuracy of the box's position and size. Visually, high confidence boxes are shown with red lines, while low confidence boxes are shown with green lines, as illustrated in Figure 2.16b [28].

Also, the model generates a class probability map for each cell, which estimates the likelihood of different object categories Figure 2.16c. Final object identification is based on the combination of bounding box confidence scores and the respective class probabilities [28]. A post-processing refinement step filters the results using a configurable confidence

threshold.

(a) (b) (c)

Figure 2.16: Example YOLO detection steps. Adapted from [22].

The YOLO models are evaluated using a metrics created to balance accuracy and speed [19]. These metrics is responsable to reflect the model's performance in detecting and classifying objects across different scenarios.

Several metrics are used to provide an understandable evaluation, including Precision (P), Recall (R), the $F1$ score, Intersection over Union (IoU), and Mean Average Precision (mAP).

Recall (R) quantifies the proportion of correctly predicted instances (True Positive (TP)) relative to all actual positive instances in the dataset [45]. It indicates the model's capacity to detect relevant objects. The formula is given in Equation 2.6, where TP denotes the number of correctly detected objects and False Negative (FN) the number of actual objects missed.

$$R = \frac{TP}{TP + FN} \tag{2.6}$$

Precision (P) represents the proportion of true positives among all predicted positive instances. It reflects the model's ability to limit false positives [45]. The formula is shown in Equation 2.7, where False Positive (FP) denotes the incorrect classification of

non-object regions as objects.

$$P = \frac{TP}{TP + FP} \tag{2.7}$$

The $F1$ score, defined in Equation 2.8, is the harmonic mean of precision and recall, providing a single indicator of overall performance. This metric is particularly useful in page object detection, where it is necessary to maintain a balance between precision and recall is critical. Its values range from 0 to 1 [23].

$$F1 = \frac{2PR}{P + R} \tag{2.8}$$

To evaluate the alignment between a predicted bounding box and the ground truth, the IoU metric, defined in Equation 2.9, is used. It quantifies the overlap between predicted and actual bounding boxes.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.9}$$

The mAP, defined in Equation 2.11, evaluates object detection models by averaging Average Precision (AP) scores across all object classes [36]. The calculation of AP is shown in Equation 2.10,

$$AP = \int_0^1 P(R)dR, \tag{2.10}$$

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N}, \tag{2.11}$$

where N is the number of object classes.

The work in [22] presents mAP variants for evaluating object detection models. The first, mAP_{50}, evaluates performance at a fixed IoU threshold of 0.5. A more comprehensive variant, $mAP_{50:95}$, averages the mAP over ten IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05.

2.2 Optical Character Recognition (OCR)

In the context of document analysis, the OCR refers to the process of converting images of printed or handwritten text into text readable by machine [3]. This conversion allow the extraction and use of textual information contained in documents for tasks such as document retrieval, text recognition, and content categorization [1].

OCR is responsible for transcribing the visual information within the delimited regions into digital text [12].

2.2.1 OCR Model Architectures

To convert visual text into digital formats, OCR systems use deep learning architectures capable of handling sequential data. This section explores the two primary neural network frameworks dominating the field: CRNN, which integrates convolutional layers with recurrent units, and Transformer based models, which utilize self-attention mechanisms to capture global dependencies within the text.

2.2.1.1 CRNN

A CRNN is designed for text recognition in images by combining CNNs for feature extraction and Recurrent Neural Networks (RNNs) for sequence modeling [46].

The CNN works the initial stage, extracting local features from the input text image. This includes convolutional layers responsible to detect visual patterns such as edges and corners, max pooling layers to reduce spatial dimensions, and batch processing to improve performance. The output is a set of visual feature maps [3].

As shown in Figure 2.17, after the feature extraction stage, the feature maps are reshaped and passed to the recurrent layers. The RNN component, such as typically implemented with Long Short-Term Memory (LSTM) networks, it captures the sequential dependencies among characters or words in the text [3].

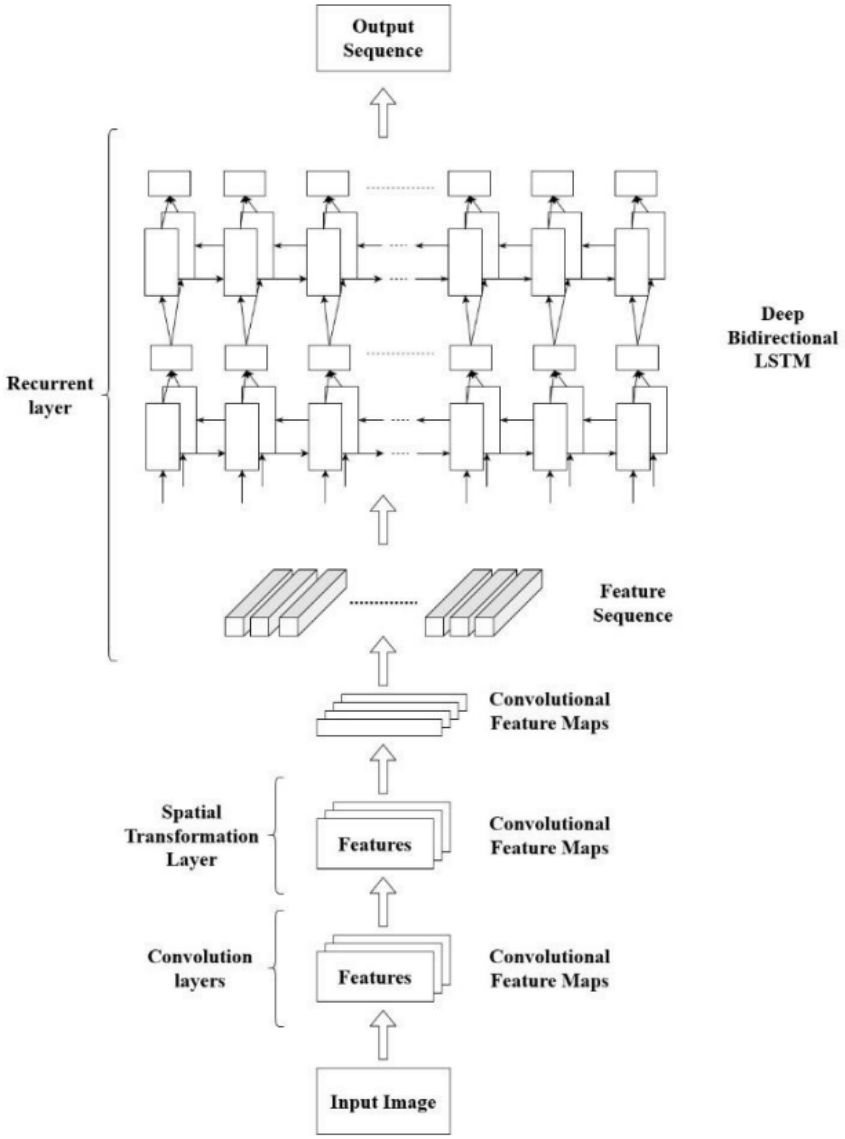


Figure 2.17: Architecture of CRNN for OCR [3].

In the context of text recognition, a key improvement in CRNNs is the use of bidirectional LSTMs, which process sequences in both forward and backward directions. With this design, word prediction performance improves by incorporating context from both preceding and succeeding characters [46]

The Connectionist Temporal Classification (CTC) aims to align the network's output with the correct text without requiring pre-segmentation of individual characters for text decoding. This approach avoids the need for manual character level annotations during

training [47].

This architecture is appropriate for recognizing text in newspapers, which present diverse formats, varying column structures, multiple font sizes and styles, and irregular advertisements [7]. The CNN extracts features independently of the text's visual appearance, while the LSTMs model sequential data of arbitrary length and capture contextual information [46]. With CTC, character level segmentation is not required [48].

2.2.1.2 Models Based on Transformers

An alternative to CRNNs for OCR is models based on Transformer, which have seen increasing adoption due to their innovative architecture, particularly the use of global attention mechanisms. Transformers provide a different paradigm for processing sequential data and capturing dependencies [1], whereas CRNNs have traditionally been a popular architecture for text recognition in images [3].

The central innovation in models based in Transformers, such as TrOCR and UltOCR, lies in the self-attention mechanism, which assigns weights to different parts of the input sequence when processing each element. This design enables the direct modeling of long-range dependencies within the text [1].

Unlike traditional CNNs, which rely on local receptive fields, or RNNs, self-attention can relate words or characters that are distant in the input sequence, modeling their interdependencies regardless of position. In text recognition, this is particularly important because a character's meaning might be influenced by factors that come significantly earlier or later in the sequence [41].

Despite the advantages of models based in Transformers, many state-of-the-art OCR systems continue to adopt CRNNs due to their practical balance between speed and accuracy [47]. This implies that many real world OCR applications still choose CRNN based designs, even though Transformers theoretically provide advantages in capturing intricate textual relationships [46].

2.2.2 OCR Engines and Tools

The implementation of OCR in applications involves engines that balance recognition with computational performance. While traditional tools often depend on rule based techniques, modern frameworks have adopted deep learning architectures to handle diverse languages and complex document structures. This section presents an overview of the available OCR tools, with a specific focus on PaddleOCR, an open-source system that integrates recognition algorithms with multilingual document analysis.

2.2.2.1 PaddleOCR

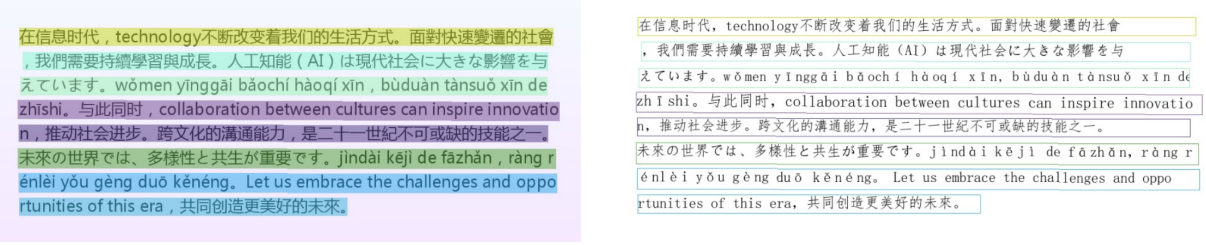
PaddleOCR's text recognition module largely uses the CRNN architecture and CTC for text decoding. CNN layers extract features from input images, identifying relevant patterns necessary for image recognition [3].

PaddleOCR is an lightweight OCR system that is open-source and released under the Apache license. It is introduced as a toolkit for OCR and document parsing [49]. Its text recognition module primarily uses the CRNN architecture with CTC for text decoding [48].

The text content is obtained by applying the recognition model based in CRNN into PaddleOCR to the detected text regions [48]. The recognition process is illustrated in Figure 2.18.

Figure 2.18: Text recognition process. Adapted from [48].

As indicated in Table 2.4, PaddleOCR's recognition models, like PP-OCRv5, aim to strike a balance between accuracy and efficiency. These models support different scripts, such as Simplified Chinese, Traditional Chinese, Chinese Pinyin, English, Portuguese, and Japanese [49]. Figure 2.19 offers an example in which Figure 2.19a presents a multilingual text and Figure 2.19b displays the matching extraction.



(a) (b)

Figure 2.19: Examples of multilingual text recognition and extraction [49].

The PP-OCRv5 model is based on a dual path framework that employs PP-HGNetV2 as its feature extraction backbone. The first path consists of a GTC-NRTR branch, which uses an attention mechanism to enhance sequence modeling during training [50]. The second path, an SVTR-HGNet branch with a CTC loss function, is optimized for inference. A guided training strategy is adopted in which the GTC-NRTR branch supervises the SVTR-HGNet branch. During inference, only the computationally lighter SVTR-HGNet branch is used, providing a balance between accuracy and efficiency [51], [52].

The PP-OCRv5 process, shown in Figure 2.20, is a multi-stage procedure that aims to optimize accuracy and efficiency. It starts with a preprocessing module that uses models like PP-LCNet and UVDoc to fix image orientation and distortion. The text identification stage makes use of a model built on the PP-HGNetV2 backbone, which is supplemented by knowledge distillation from a GOT-OCR2.0 instructor model and data augmentation techniques such as ERNIE-4.5-VL mining. After detection, a line alignment module refines individual text lines to increase recognition accuracy [49].

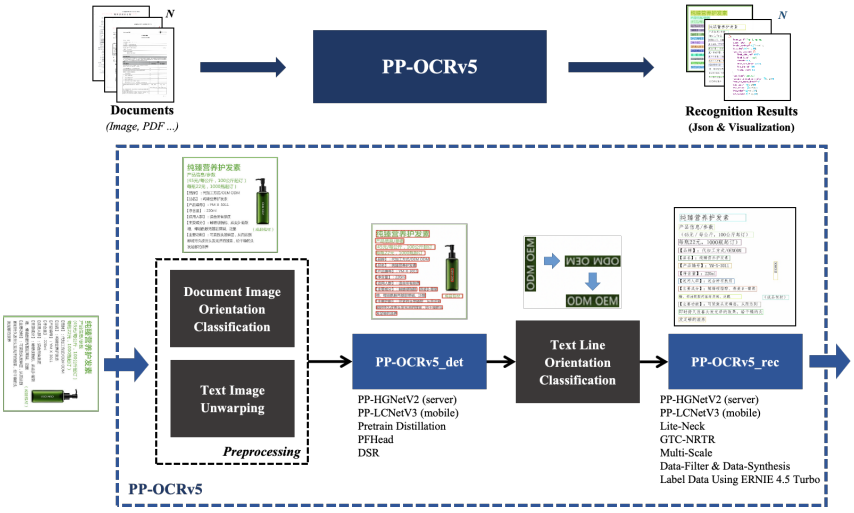


Figure 2.20: Process of PP-OCRv5 [49].

Model	Handwritten Chinese	Handwritten English	Printed Chinese	Printed English	Traditional Chinese	Ancient Text	Japanese	General Scenario	Pinyin	Rotation	Distortion	Artistic Text	Average
PP-OCRv5 server_det	**0.803**	**0.841**	**0.945**	**0.917**	**0.815**	**0.676**	**0.772**	**0.797**	**0.671**	**0.800**	**0.876**	**0.673**	**0.827**
PP-OCRv4 server_det	0.706	0.249	0.888	0.690	0.759	0.473	0.685	0.715	0.542	0.366	0.775	0.583	0.662
PP-OCRv5 mobile_det	0.744	0.777	0.905	0.910	0.823	0.581	0.727	0.721	0.575	0.647	0.827	0.525	0.770
PP-OCRv4 mobile_det	0.583	0.369	0.872	0.773	0.663	0.231	0.634	0.710	0.430	0.299	0.715	0.549	0.624

Table 2.4: Performance of different models in multiple OCR scenarios. Adapted from [49].

Performance assessments show that PaddleOCR produces consistent results across a variety of datasets. The Levenshtein Distance compares two text strings by calculating the number of single character modifications required to convert one string to the other. This metric is especially useful for assessing numbers and short words, since it captures small errors that word level measures often miss [47]. The Character Error Rate (CER) is defined in Equation 2.12:

$$\text{CER} = \frac{S + D + I}{N}, \tag{2.12}$$

where S is the number of character substitutions, D is character deletions, I is character insertions, and N is the total number of characters in the reference text, represented by the sum of $S + D + C$, where C is the number of correct characters. Lower CER values

indicate better performance [53].

The Word Error Rate (WER), as defined in [54], is presented in Equation 2.13:

$$\text{WER} = \frac{S + D + I}{N} \tag{2.13}$$

where: S is the number of word substitutions, D is word deletions, I is word insertions, and N is the total number of words in the reference text, represented by the sum of $S + D + C$, where C is the number of correct words.

Alongside PaddleOCR, many other OCR systems have been developed using different approaches, including EasyOCR, Keras-OCR, Tesseract, OpenCV, PyMuPDF, and DocTR [55]. A comparison is shown in Table 2.5.

OCR Engine	**Accuracy (%)**	**CER**	**WER**	**Notes**
PaddleOCR	**67.28**	**0.43**	**0.66**	Best overall performance; fast and accurate.
EasyOCR	27.24	0.64	1.05	Good speed, moderate accuracy.
PyTesseract	14.63	0.50	0.66	Traditional OCR; poor with handwriting or low-res input.
OpenCV OCR	14.63	0.79	0.82	Slower and less accurate with complex layouts.
Keras-OCR	3.51	0.77	1.41	High error rates; not suitable for precision tasks.
PyMuPDF	6.95	1.00	1.02	Basic OCR; limited effectiveness.
DocTR	17.21	0.81	1.12	Deep learning; less mature in handling variance.

Table 2.5: Comparison of OCR Engines Based on Accuracy and Error Rates. Adapted from [55].

Traditional engines such as Tesseract rely on techniques based in rules, whereas modern alternatives like EasyOCR and DocTR employ deep learning models. Their performance differs depending on layout complexity, language, and document type [55].

Chapter 3

System Architecture and Development

3.1 Development Tools

This chapter will describe the software frameworks, development platforms, and computational hardware used for the implementation and moldels training.

Each tool was selected based on its specific applicability in the context of DLA and OCR.

3.1.1 Programming Language and Frameworks

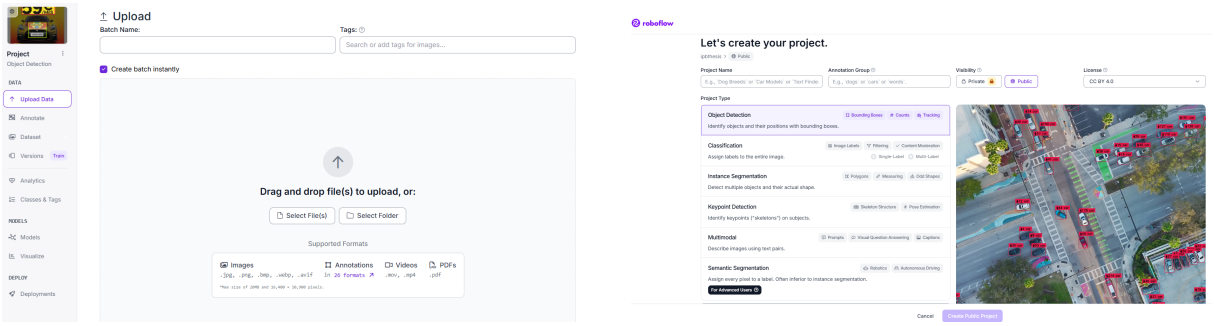
Python, the most programming language adopted in the Artificial Inteligence (AI) research community, was chosen for its clear syntax, its established and its support libraries.

The deep learning components were implemented using two frameworks. The PyTorch was used to build and train the custom layout detection model based in YOLO, which can provide the flexibility required by the project's architecture. For the text extraction stage, the PaddlePaddle framework was integrated through PaddleOCR, whose recognition model based in CRNN was applied during the recognition phase of the process.

3.1.2 Roboflow

The Roboflow is a computer vision platform designed to help the development and deployment of computer vision models [56]. Its goal is to provide tools and infrastructure that help the creation of computer vision models, making the process to label more easy.

This tool facilitates the process by allowing users to upload images using a one click upload option and working in a collaborated project, as show in Figure 3.1a. Roboflow offers tools for labeling images with different porpose, Figure 3.1b, which is a step for training computer vision models, identifing and locating objects within images.



(a) (b)

Figure 3.1: Roboflow workspace.

The platform is used to train computer vision models using deep learning frameworks such as TensorFlow, COCO, PyTorch, TFRecord, and YOLO [57].

This system works as a platform for developing computer vision by providing tools for data annotation, data augmentation, model training, validation, and deployment [58].

3.1.3 Google Colaboratory

Google Colaboratory, shown in Figure 3.2, was used for model development, training, and testing. The Colab is a Jupyter Notebook service based on cloud that offers a ready to use machine learning environment, eliminating the need for local setup and dependency management.

One of Colab's main features is free, on-demand access to hardware, especially Graphics Processing Unitss (GPUs). Training deep learning models requires substantial computational power, often expensive. Colab's platform enables the training of DLA models with high computational demands.

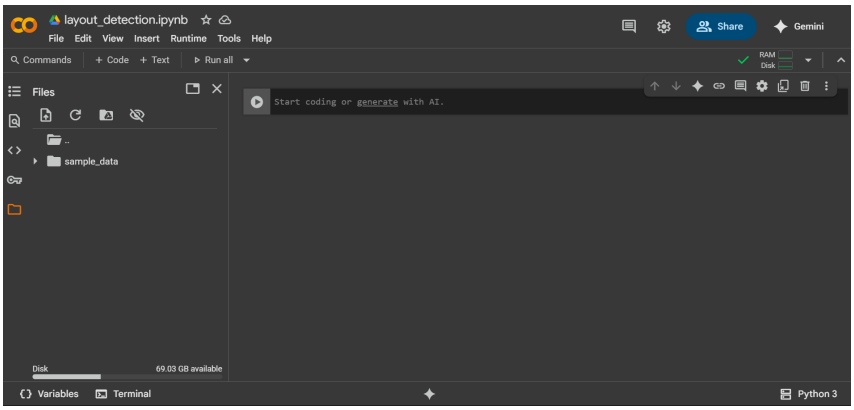


Figure 3.2: Google Colaboratory interface.

In the Colab environment, the primary freely available hardware is the NVIDIA Tesla T4 GPU, a processor built on the NVIDIA Turing architecture. The project's requirements is attempt by T4 features, and its specifications are listed in Table 3.1.

Feature	**Specification**
GPU Architecture	NVIDIA Turing (TU104)
Process Size	12 nm
NVIDIA CUDA Cores	2,560
Turing Tensor Cores	320
GPU Memory	16 GB GDDR6
Memory Interface	256-bit
Memory Bandwidth	300 GB/s
Peak FP32 Performance	8.1 TFLOPS
Peak Mixed Precision	65 TFLOPS
Peak INT8 Performance	130 TOPS
Max Power Consumption	70 W
System Interface	PCIe 3.0 x16

Table 3.1: Technical specifications of the GPU based on NVIDIA Turing (TU104) architecture. Adapted from [59].

Colab integrates with Google services, enabling datasets, model checkpoints, and experimental results to be stored in Google Drive, which can be mounted directly into a session. This integration supports data persistence and simplifies management.

The free version of Colab does not provide unlimited resources. After extended use, the runtime may be interrupted, making the process more difficult for experiments that require long execution times.

3.2 Proposed Solution

This work proposes a multi-stage process, to address the challenges of automated information extraction from documents with diverse layouts, such as newspapers and magazines, that integrates based deep learning computer vision methods for both DLA and OCR. The system converts unstructured visual data from document images into a semi-structured format.

The architecture illustrated in Figure 3.3, is organized into three stages. The first stage processes the image input. The second stage detects the document layout components and extracts textual content from the full page. The third stage integrates these outputs by linking the extracted text to its corresponding layout category.

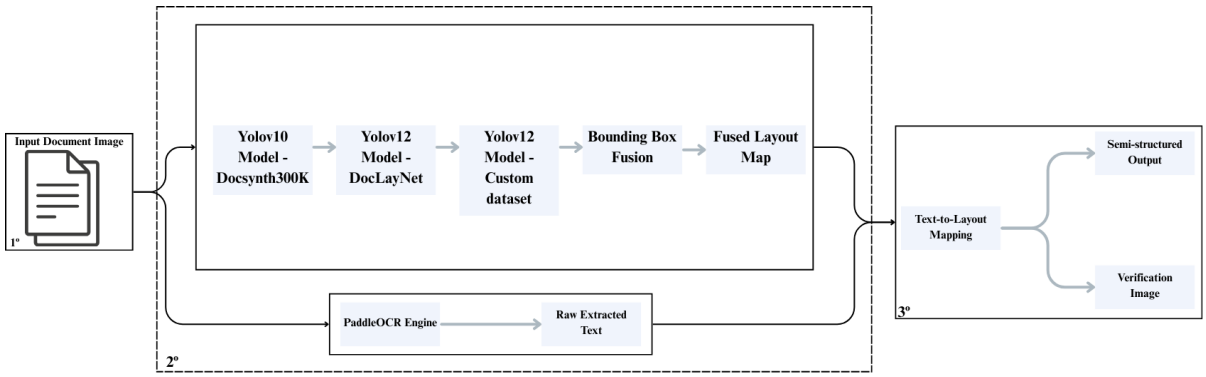


Figure 3.3: Block Diagram of the system.

3.2.1 Layout Detection Module

For the initial stage of the process, analyzes the physical structure of the input document image, an object detection model from the YOLO family is employed. These models are trained to detect distinct document components, such as headlines, paragraphs, images, and tables.

The detection module outputs a set of bounding boxes, as shown in Figure 3.4, where each box marks a region of interest on the page and is assigned a class label.

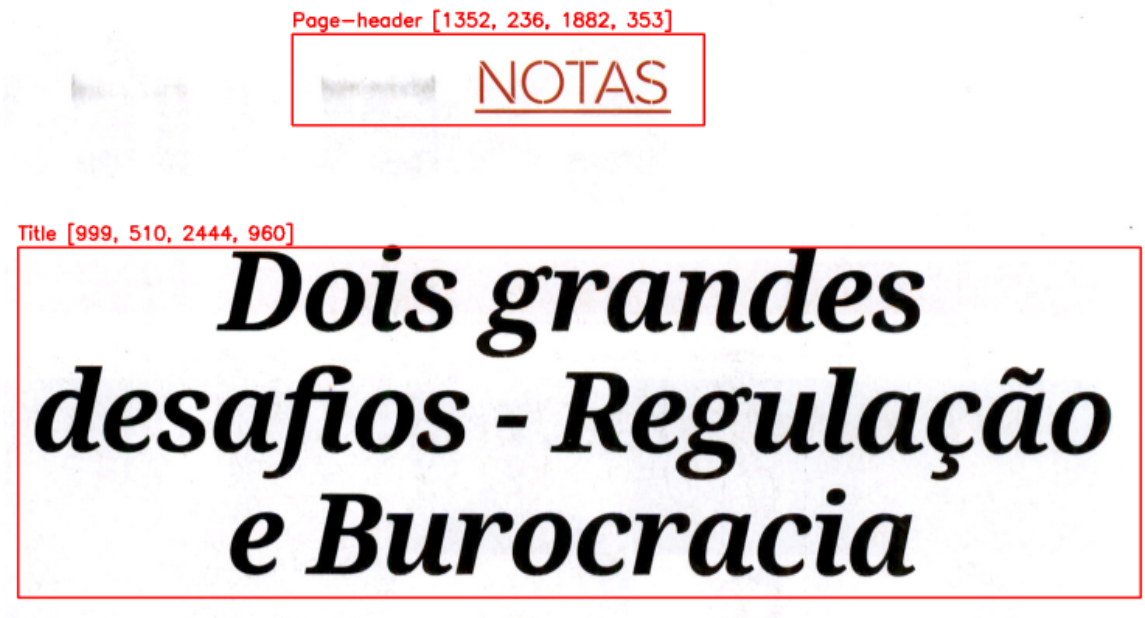


Figure 3.4: Classes and their respective bounding boxes detected by YOLO.

Training is performed using the Roboflow platform, which supports dataset annotation and versioning. After the document layout is segmented into labeled bounding boxes, the second stage of the process begins with text extraction.

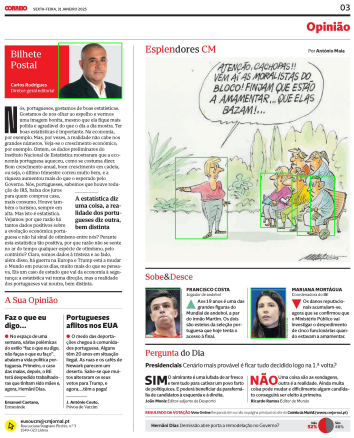
For each image, an OCR engine such as PaddleOCR extracts the text, producing both the recognized content and the associated bounding boxes.

The choice of PaddleOCR is because its accuracy, multilingual support, and balance between performance and computational efficiency.

3.2.2 Fusion Models Layout Detections

When applied to documents, object detection models, can sometimes produce fragmented results. For example, a single paragraph of text might be detected as several distinct but overlapping bounding boxes. This is often due to the model's sensitivity to line breaks, font changes, or labelling during process to traing the model.

When running models detectors in an image to predict it, each model may output overlapping or shifted boxes for the same object. This problem is show in Figure 3.5.



(a) Model [37].

(b) Model [17].

(c) Model [60].

(d) Model [61].

Figure 3.5: YOLO segmentation applying different models in same image.

This representation bring the same image applied to different models and with different output, where both may fail to detect certain layout elements or may produce imprecise boundaries.

To unify these fragmented and redundant detections into a coherent layout, the complete fusion described in Algorithm 4 is applied. This procedure begins by detecting regions using the pretrained YOLOv10 model, followed by the YOLOv12m model trained on 2k images and finally the custom YOLOv12m model.

At each stage, redundant boxes are removed using the strategy defined in Algorithm 5. When different models detect similar regions with partial overlap, the system performs a fusion of bounding boxes as defined in Algorithm 6.

After merging detections from all models, remaining geometric conflicts between boxes are resolved through the intersection and subtraction procedure described in Algorithm 7. The geometric cutting necessary to preserve the largest meaningful region is performed by the cutting operator detailed in Algorithm 8.

3.2.3 Heuristic text order

The output provided by YOLO usually is not ordered. The sequence in which the model returns detected regions is arbitrary, often reflecting the model's internal processing rather than the document's logical flow. A list of detected elements, however, is essential for understanding the document's content and sequence.

To transform unordered data into a logical structure that follows the natural human reading order, a hierarchical process is applied. This process is organized into two levels, as shown in Algorithm 1 and Algorithm 2.

Algorithm 1 Region Sorting

Input: Unordered regions $R = \{r_1, \ldots, r_n\}$
Output: Regions sorted in reading order
Step 1: Compute center coordinates for each region
    $x_c(r) \leftarrow (x_{\min}(r) + x_{\max}(r))/2$
    $y_c(r) \leftarrow (y_{\min}(r) + y_{\max}(r))/2$
Step 2: Sort by (y_c, x_c) (top-to-bottom, left-to-right)
return Sorted regions

Algorithm 2 Text Fragment Sorting

Input: Unordered fragments $F = \{f_1, \ldots, f_m\}$ within one region
Output: Fragments sorted in reading order
Step 1: Sort fragments roughly by vertical position y_c
Step 2: Group fragments into text lines
    foreach *fragment* f **do**
        Find existing line where f aligns vertically
        if *alignment found* **then**
            Add f to that line
        else
            Create new line with f
Step 3: Sort lines top-to-bottom
Step 4: Within each line, sort fragments left-to-right
Step 5: Concatenate all sorted lines
return Sorted fragments

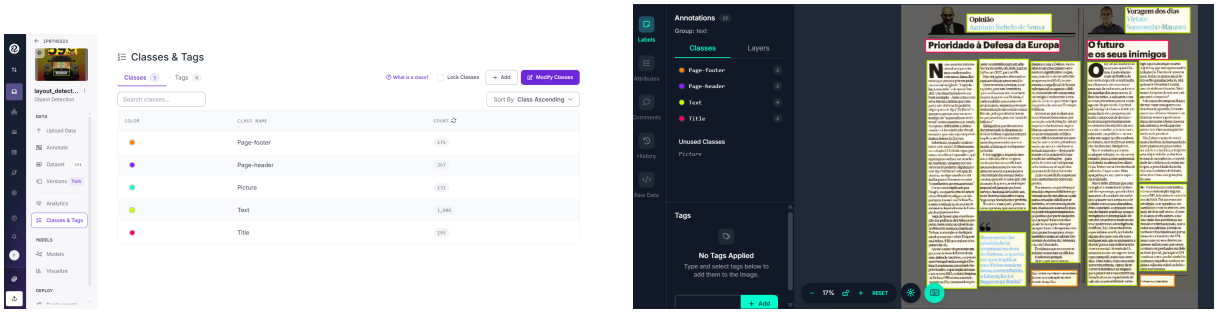
The process is constituted by implementing a sorting process that first arranges the document structure and then refines the text sequence within each structural element. This approach is like a human reader interprets a page: first identifying major sections (such as columns or paragraphs) and then reading the text within those sections line by line.

3.2.4 Dataset Creation

The results of the YOLO object detection model depend on the dataset used for training.

The image collection was provided by different newspapers and magazines. The objective was to include a variety of layouts to ensure that the model could handle diverse

document structures. The class schema created to categorize the key structural elements of a document is shown in Figure 3.6a. An example of a labeled image is presented in Figure 3.6b.



(a) (b)

Figure 3.6: Classes and labels in Roboflow for model training.

The dataset annotation process had to follow criteria, ensuring that the same regions were labeled with the same classes from beginning to end. This procedure avoid ambiguities and inconsistencies in the model's learning.

The annotated dataset was partitioned into three subsets. A randomized split was applied with the following allocation: for a dataset of 151 images, 70% was assigned to training, 20% to validation, and 10% to testing. This resolution was adopted because, in [17], the authors used high resolution images to better capture complex layouts and small objects.

3.2.5 Model training

The model used will determine how well the project performs. The YOLOv12 model was chosen for training as, at the time of this study, it was the most advanced model in the YOLO series. Benchmark results that demonstrated better performance than previous iterations served as the basis for this choice. The architectural improvements of YOLOv12 are particularly advantageous for document analysis, enhancing its capacity to detect objects across multiple scales and complex spatial arrangements.

The model was not trained from randomly initialized weights but initialized with

weights pre-trained on the COCO dataset. This approach enables to have a performance without demanding a large dataset and extensive training time. Based on this prior knowledge, the model fine-tunes its weights to enhance accuracy in document layout detection.

Using these pre-trained weights as a foundation, the model was further trained on a annotated custom dataset and the public DocLayNet[62] dataset. During this stage, the model adjusted its learned features to match the specific visual characteristics of the defined layout classes.

Hyperparameter selection also influenced performance. The initial training employed the default settings of the YOLOv12 framework. The final configuration of hyperparameters is reported in chapter 4.

3.2.6 Integration of Layout Analysis and OCR

The layout detection model provided by YOLO identifies the regions of a document and its respective layout and position, while the OCR model PaddleOCR extracts raw text fragments from the image. This section details the association algorithm developed to map each text fragment to its corresponding semantic layout region.

The association logic is based on spatial proximity. The text fragment extracted by the OCR model belongs to a layout region if its bounding box has a significant geometric overlap with the bounding box of that region.

The Algorithm 3 evaluates each text fragment against all detected layout regions, calculates a quantitative measure of their overlap, and assigns the text based on a criterion.

Algorithm 3 Text-to-Region Association

Input: Document image I; layout regions R

Output: Regions with associated text

Step 1: Run OCR on entire image to extract text fragments F

Step 2: Filter fragments by confidence score (≥ 0.1)

Step 3: Assign each fragment to best-matching region

    foreach *text fragment* f **do**

        Compute overlap ratio with each region Find region r with maximum overlap

        if *overlap* ≥ 0.3 **then**

            Assign f to region r

return Regions with text

The final step of this process of regions, is a creation of a JSON file. This format is used to be readable by machine for automated processing and interpretable by human for analysis. As shown in the Code 1, the structure is as follows:

Code 1 Exemple of JSON structure

```
{
  "1": { "cls": "Page-header", "bbox": [683,159,2012,442], "text": "FIDELIDADE
  ↪ SEGUROS DESDE 1808" },
  "2": { "cls": "Title", "bbox": [179,1608,1620,1718], "text": "SEGURO DE VIDA
  ↪ INDIVIDUAL" }
}
```

where `"cls"` denotes the category of the region (e.g., `"Title"`, `"Text"`, `"Picture"`) predicted by the YOLO model; `"bbox"` is an array of four integers specifying the pixel coordinates $[x_{\min}, y_{\min}, x_{\max}, y_{\max}]$ of the region's bounding box; and `"text"` contains the final text string assigned to that region.

Chapter 4

Results

This chapter presents the experimental results obtained with the layout detectors based in YOLO, the PaddleOCR module, and the heuristics designed for reading order sorting in newspapers.

The chapter assesses how the proposed approach addresses the goals defined in chapter 1, as well as discuss limitations and the areas of development remaining to be explored.

4.1 Experimental Setup

This section describes the experimental context in which all results were obtained, including the hardware and software environment, the datasets employed, the evaluation metrics, and the principal training hyperparameters.

4.1.1 Hardware and Software Environment

The training of the object detection models and the execution of the OCR module were conducted on a Virtual Machine based in a cloud instance. The main specifications were:

- **Hardware**:
  - **GPU**: NVIDIA Tesla T4

- **GPU Memory**: 16 GB GDDR6
- **CPU**: Intel Xeon (2–4 vCPUs)
- **System RAM**: 13 GB
- **Environment**: Google Colaboratory Pro+

- **Software**:
  - **Operating System**: Ubuntu 22.04.4 LTS
  - **Python**: Version 3.12.12
  - **Deep Learning Framework**: PyTorch 2.1.0
  - **Object Detection**: Ultralytics library v8.3.228 with the pre-trained YOLOv12m model
  - **OCR**: PaddleOCR v3.2.0, employed for text extraction from detected regions

4.1.2 Datasets for Layout Detection

Two datasets were chosen to train the model. The first is custom newspaper and magazine images and the second one, DocLayNet, was included because it provides a large, pre-annotated set of over 80k document images.

4.1.2.1 Custom Dataset

The custom dataset comprises 151 images annotated in Roboflow. The defined classes include: Page-footer, Page-header, Picture, Text, and Title. The dataset was divided into 70% for training, 20% for validation, and 10% for testing. An example of these images appears in Figure 4.1.

(a) (b)

Figure 4.1: Classes and labels in Roboflow for model training.

The defined annotated classes:

- **Page-header** and **Page-footer**: These regions were annotated to isolate page numbers, and publication dates from the main content.
- **Title**: This class labels the headlines and sub-headlines, distinguishing them from the body text.
- **Picture**: This category represent all visual media, illustrations and graphic advertisements.
- **Text**: This class covers the body content.

4.1.2.2 Public Datasets

The public dataset used in this investigation was DocLayNet, a human annotated resource designed for document layout segmentation. The complete dataset comprises 80,863 pages from different types of document sources. The layout variation of several source categories and the following classes are presented: Caption, Footnote, Formula,

List-item, Page-footer, Page-header, Picture, Section-header, Table, Text, and Title. A case is illustrated in Figure 4.2.

(a) (b)

Figure 4.2: Classes and labels in DocLayNet dataset.

For this work, the model was trained using a subset of 2,000 images sampled from the complete DocLayNet collection.

4.1.2.3 Pre-trained YOLOv10 Dataset

The pre-trained DocLayout-YOLO model was initially trained on the synthetic dataset DocSynth-300K, which comprises 300,000 document pages. This dataset is produced through the Mesh candidate BestFit algorithm, which frames document synthesis as a two dimensional bin-packing task. The visual and structural components assembled during generation originate from the *M'Doc* test dataset, which supplies the raw elements for layout construction[60].

After this stage, the model is adapted using three real world layout datasets: D4LA, DocLayNet, and DocStructBench. The Figure 4.3 illustrates the dataset structure.

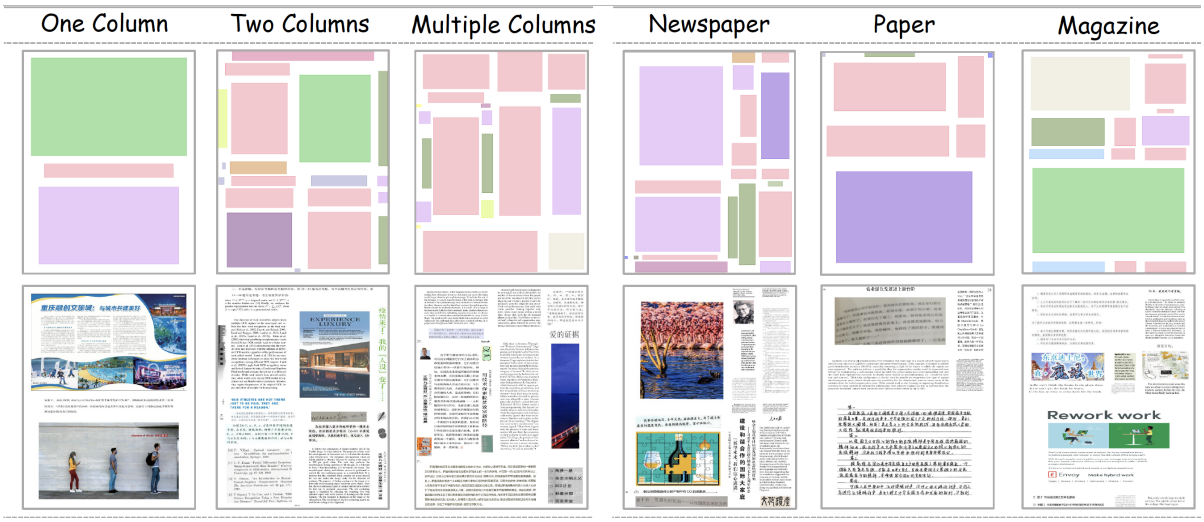


Figure 4.3: Schema of DocSynth300K Dataset.

During fine-tuning, the model learns 10 categories: Title, Plain Text, Abandoned Text, Figure, Figure Caption, Table, Table Caption, Table Footnote, Isolated Formula, and Formula Caption.

4.1.3 Training Hyperparameters and Evaluation Metrics

The experiments utilized the pre-trained YOLOv12m variant, selected because it balance between computational demands and the risk of overfitting. The model was applied to both the custom dataset and the already annotated DocLayNet dataset. The number of epochs and the batch size were constrained by the available computational resources.

The model was trained for 20 epochs with a batch size of 16 images. In contrast, the model proposed by [60] was trained for 30 epochs with a batch size of 128. The Table 4.1 presents the main evaluation metrics. The pre-trained YOLOv10 [60] document only show mAP@50 and mAP@50–95; Precision, Recall, and F1 Score are absent and therefore marked as n/a.

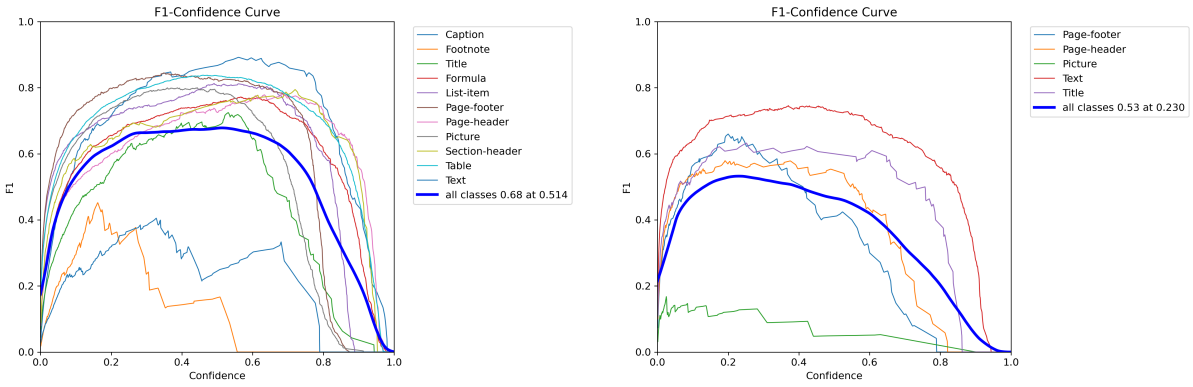
Dataset	Precision	Recall	F1 Score	mAP@50	mAP@50–95
Custom	0.55	0.539	0.544	0.519	0.288
Public	0.64797	0.761	0.667	0.728	0.524
Pre-trained [60]	n/a	n/a	n/a	0.934	0.797

Table 4.1: Performance metrics comparison between models.

As shown in Table 4.1, the model trained on the public dataset achieved better performanced results than the one trained on the custom dataset. The result is due to a reduced number of labeled examples in the custom dataset, the imbalance between classes, the reduced number of training epochs, and occasional inconsistencies in the annotations.

The Figure 4.4a, shows the results of on the public dataset and includes 12 distinct classes. In this training, the "all classes" curve achieves a peak F1-score of 0.647 at a confidence threshold of 0.514. This indicates a good overall performance for the model on this diverse public dataset. An F1 of 0.64 suggests a balance between identifying the different document elements (recall) and ensuring those identifications are correct (precision). Classes like Caption, Title, and Section-header are among the best performers, with a peak close 0.8 F1. The Footnote and List-item are the most challenging classes, with their F1 curves being significantly lower, especially Footnote which barely breaks 0.4 F1.

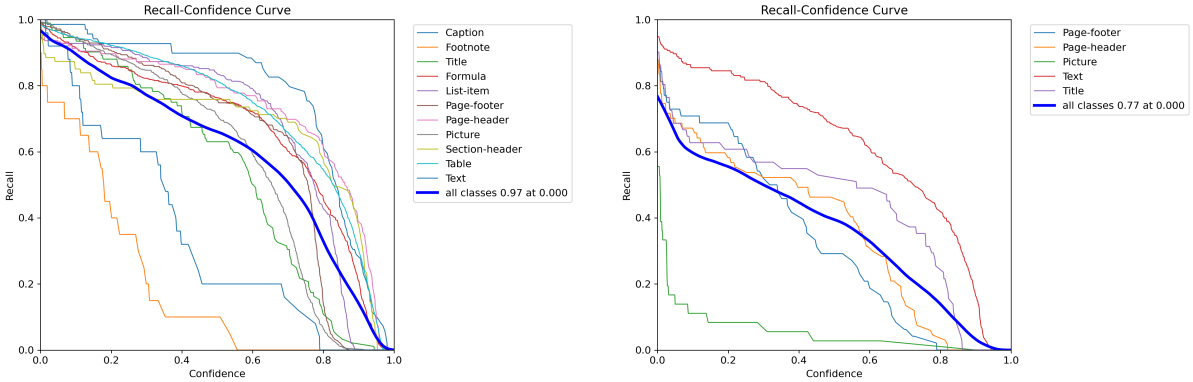
The Figure 4.4b shows the results of the train on the custom dataset and includes only 5 classes. The "all classes" curve peaks at a lower F1-score of 0.54 at a confidence threshold of 0.23. This is a significant drop if compared to the public dataset. The Text is the better in classes group, reaching an F1-score of over 0.7. The Picture has the worst perform, barely exceeding an F1 of 0.15 at its peak. It means that these classes might be inconsistent in dataset and uniformly labeled than other elements, what can unbalanced the dataset.



(a) (b)

Figure 4.4: F1-Confidence curve comparison.

As show in Figure 4.5a, the "all classes" curve achieves a higher peak, with a value of 0.97. The curve keep a considerable Recall across the entire Confidence spectrum. The "all classes" curve show in Figure 4.5b have a lower value of 0.77 at the lowest confidence.



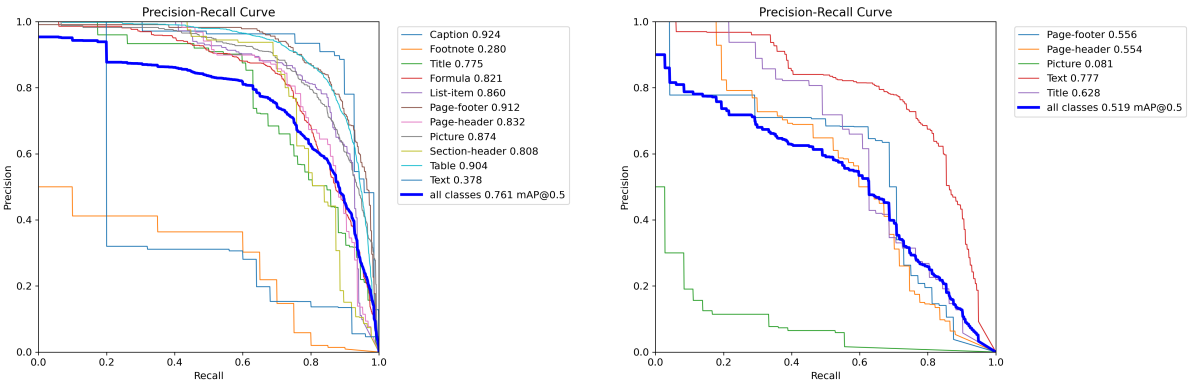
(a) (b)

Figure 4.5: Recall Confidence curve comparison.

The most probable cause for the performance gap is the difference in the size and quality of the annotations or the diversity of the datasets. The low Recall for the 'Picture' as show in Figure 4.5b class suggests an issue with the private data itself.

The main results demonstrate that the model trained on the public dataset exhibits higher Recall across all confidence levels and most individual classes compared to the model trained on the private dataset.

The model trained on the public dataset, Figure 4.6a, demonstrates a performance with an mAP@0.5 of 0.761. The Precision-Recall curves indicate that the model can maintain precision even as recall increases. Theses classes Caption, Page-footer and Table were detected with reliability. On the other hand, the Text and Footnote, these classes show a drop in performance.



(a) (b)

Figure 4.6: Recall Confidence curve comparison.

The model's performance on the private dataset, Figure 4.6b, shows a considerable decrease, with an overall mAP@0.5 of 0.519. The Text class is the have the best perform in the private dataset. This suggests a difference in labeling or data distribution. The model failed to detect pictures in the private dataset, with an Precision Recall near to zero.

4.2 Comparative Layout Detection Models

In this section is present and compare the results of the different layout detection models: YOLOv12m with two training database, pre-trained YOLOv10, and the fusion model as result.

4.2.1 Comparison of model predictions

The Figure 4.7 compares the predictions of the three detection models on the same magazine page. All predictions were obtained using identical inference settings, with a confidence threshold of 15%.

The pre-trained model, Figure 4.7c, detects all major layout components, including both photographs, the body text, section headers, and the page footer. The bounding boxes around the images visually align closely with the content, which suggests good accuracy for this class. However, the model appears to overuse the section-header label.

The custom model trained only on the manually annotated dataset, Figure 4.7b, exhibits a different behaviour. In this configuration, the detector focuses almost exclusively on textual regions: the three text columns are captured with compact and well aligned text boxes, and false positives are rare. At the same time, both photographs and the page footer are missed, producing false negatives for the picture and marginal classes. This pattern is consistent with precision and a good F1 score for the dominant text class, but low recall.

The model trained with approximately 2,000 images, Figure 4.7a, this approach restores the detection capabilities of the pre-trained, while mitigating its structural limitations. It correctly identifies the two photographs, the page header and footer, and all main text regions, producing recall across classes better than for the custom model.

(a) Model trained with Public Dataset.

(b) Model trained with Custom Dataset.

(c) Model pre-trained [60].

Figure 4.7: Qualitative comparison of YOLO layout detection models on the same magazine page.

However, the model trained into 2k images produces more fragmented and sometimes duplicated boxes, particularly for text, which increases the number of false positives and, consequently, lowers precision. These observations indicate that this model can be useful for the purposes of this work: it preserves the ability to recognize the layout elements required for downstream OCR and analysis, but with the cost of some over segmentation that can be addressed with simple post-processing heuristics.

4.2.2 Fusion of YOLOv12 models and YOLOv10

This section presents the predictions generated by the three models and the consolidated output obtained from merging their detections, to address the issue described in 3.2.2. The Figure 4.8 illustrates what each model identifies and how the merged result reflects the spatial agreement between bounding boxes.

After applying the fusion rules in subsection 3.2.2, the detections produced by the three models (YOLOv12-custom, YOLOv12-2k, and YOLOv10-pretrained) were consolidated into a single output. The Figure 4.9 shows the final prediction, already filtered.

The fusion process reduced redundancy and discarded unstable detections, retaining those that appeared consistently across models. Large structural components, such as the newspaper title, headers, and other dominant regions, tend to converge in the fused result because the models commonly agree on their boundaries.

The method also manages smaller or visually intricate elements, including photographs and captions. When individual models shifted their bounding boxes or produced contours more than necessary, the fusion logic relied to correct these deviations. This procedure kept the final layout aligned with the page structure.

In the merged prediction, Figure 4.9, each detected region receives an order label (L1, L2, L3, ...). This ordering corresponds to the reading sequence estimated by the two sorting algorithms described in subsection 3.2.3.

(a) Model trained with Custom Dataset.

(b) Model trained with Public Dataset.

(c) Model pre-trained [60].

(d) Merged detections.

Figure 4.8: Merged detections comparison of YOLO layout models.

The Algorithm 1 arranges regions from top to bottom and then from left to right. Since YOLO does not impose any ordering on its outputs, this step create a sequence that looks like how a human reader would approach a newspaper page.

The second algorithm, Algorithm 2, operates inside each region, especially when a paragraph or title has been fragmented into multiple boxes. It groups elements that belong to the same text line and then arranges them from left to right.

Figure 4.9: Final image predicted.

The resulting sequence starts at the top of Figure 4.9, moves through the central

regions, and concludes in the lower portions of the page. The ordering generated by the technique is consistent and simple to understand even in layouts that include graphics, headlines, and marginal notes.

4.2.3 Error Analysis and Limitations in Layout Detection

Although this project demonstrates good results for newspaper and article layouts, some performance problem was observed in specific scenarios, as show in Figure 4.10, particularly within advertisement pages and some magazine layouts. These failure cases can be categorized into two problem: background texture hallucinations and structural misinterpretation of elements.

A limitation observed in this approach is the tendency to hallucinate layout regions within natural images or complex graphic backgrounds. As illustrated in Figure 4.10a and Figure 4.10b, the model generates erroneous bounding boxes on layout items. In Figure 4.10b, the vertical iron bars of the fence and the texture of the pavement are incorrectly categorized as text or structural regions.

In Figure 4.10a, the reflective strip on the motorcyclist's jacket and the glossy highlights on the helmet were likely detected as a series of small, disjointed fragments. To deal with false positives, the post-processing algorithm which merged proximate text bounding boxes interpreted these bounding boxes as a single logical region, and the subtraction process also removed part of text to preserve the Quick Response code (QR code).

In addition to background hallucinations, the system's definition of any layout element's borders is inconsistent. Although there is a defined text column in Figure 4.10c, the model is unable to identify the full region. Parts of the column remain unlabeled because the bounding box does not cover the entire vertical length of the text.

On the other hand, the identification of page footers exhibits the reverse tendency, as seen in Figure 4.10d. In this case, the model creates a large bounding box for the footer area, including surrounding whitespace.

(a) Incomplete detection of textual and graphic elements in advertisement layout.

(b) Over detection and misclassification in complex visual layouts.

(c) Insufficient generalization to document layout.

(d) Error propagation and residual inconsistencies.

Figure 4.10: Failure cases and limitations of the layout detection models.

Although the footer content is limited to a page number and a brief text identifier, the model, or the subsequent post-processing aggregation, fails to encapsulate elements near the page margins, causing empty space to be included in the logical region.

4.3 Evaluation of Text Extraction and Structuring (PaddleOCR)

Following the identification of visual boundaries and document classes, the final step of the process is the extraction and structuring of textual content. This section will show the performance of the OCR module and the heuristics used to map raw text strings into the detected semantic regions.

4.3.1 Text Extraction with PaddleOCR

To retrieve the textual content, the approach adopted a single inference strategy into the image rather than a based region cropping approach. The entire image is processed by the PaddleOCR engine in a single step.

Each of the raw text fragments produced by the PaddleOCR engine has its own coordinates. The association logic described in Algorithm 3 is then used to assign these texts to the appropriate layout determined by the YOLO model. This algorithm assigns the text to the region with the IoU after calculating the ratio between the OCR detected text lines and the YOLO detected layout regions.

The result of integration of the layout model predict and show in Figure 4.9 with PaddleOCR is show in Appendix B. Here, the layout model correctly identifies the column boundaries, and the association algorithm aggregates multiple lines of raw text into a single logical block.

4.3.2 Impact of Layout Complexity

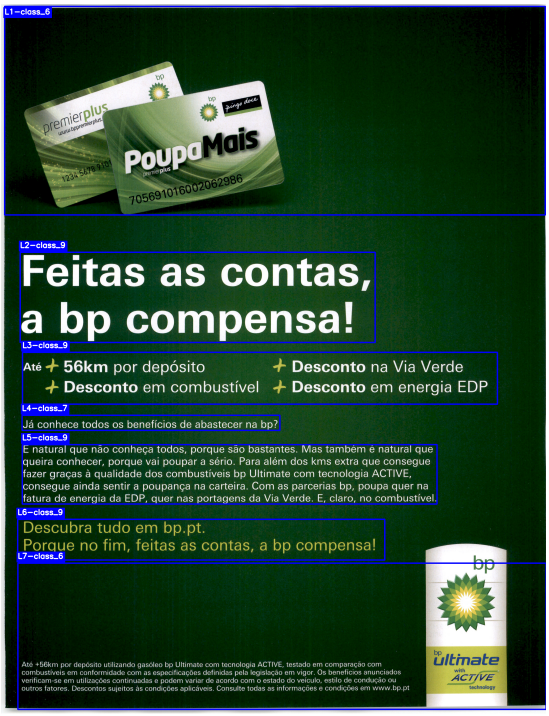
The visual properties of the input image directly affect the OCR extraction engine's performance. PaddleOCR shows consistent and precise recognition in situations with homogeneous backgrounds and strong contrast. As can be seen in Figure 4.11a, there are no visible mistakes in the text taken from the "PLANO A" section. While maintaining punctuation and diacritical marks, the OCR system was able to identify small fonts placed in dense columns ("A taxa de juro média anual implícita...").

The Figure 4.11b shows a different failure situation. In this case, specular reflections on a dark background and visual noise were mistaken for text. The model generated a series of nonexistent characters within the L5 detected bounding box ("Mas tambémé natural que eeee eecee ee cecee aererrreeeeere eeeeeee rr...") rather than accurately maintaining the sentence "Mas também é natural que queira conhecer...". This effect happens when high frequency texture patterns mimic the spatial distribution and stroke width of letters, causing the recognition model to infer text in areas with no semantic value.

Additionally, typography introduces systematic difficulties. The ornamental font used in "DESDE 1808" resulted in a character substitution in Figure 4.11c, where the digit '0' was misclassified as the capital letter 'O' ("18O8"). Furthermore, a segmentation fault that collapsed several words into a single token ("SEGURODEVIDAINDIVIDUAL") was caused by increased between characters spacing in the header. The limits of OCR models when processing display fonts optimized for visual impact rather than legibility are reflected in such errors.

Difficulties were also observed under inverted color schemes, as illustrated in Figure 4.11d. Although the main header was correctly detected, smaller textual elements positioned over complex gradients were partially lost. In these regions, the binarization stage failed to adequately separate foreground text from the background, producing missing characters and fragmented diacritics.

(a)



(b)

(c)

(d)

Figure 4.11: Analysis of OCR performance.

While the layout detection module localizes semantic regions across the page, the subsequent OCR stage remains sensitive to pixel level degradation. The shift from clean document scans to visually complex advertising layouts introduces noise patterns that affect recognition accuracy.

4.3.3 Evaluation of Reading Order Heuristic

The output of object detection models, such as YOLO, is unsorted, as detections are ordered by confidence score rather than spatial logic. To make a readable sequence, the heuristics described in Algorithm 1 and Algorithm 2 were implemented to reconstruct the logical flow of the document. The effectiveness of this approach is assessed through visual inspection of the final reading order labels $(L_1, L_2, \ldots, L_n)$ assigned to the test images.

The heuristic performs good reconstruction in layouts with a clear vertical hierarchy, such as the magazine cover shown in Figure 4.11d and the advertisement in Figure 4.11c, the heuristic achieves good reconstruction. In Figure 4.11d, the reading order follows a top-down progression, the main header is assigned L_1, followed by the secondary element at the bottom left (L_2), and concluding with the footer information (L_3).

As shown in Figure 4.11c, the heuristic follows a more articulated structure. The sequence correctly begins at the brand header (L_1) and moves on the upper content block (L_2). It then traverses horizontally to aligned the right image text (L_3) before returning to the vertical axis to process the subsequent body paragraph (L_4). Despite an imprecisely segmented bounding box (L_5), the sorting algorithm positions did it correctly in the logical sequence following the associated text. The footer region is resolved using a left-to-right strategy, the system prioritizes the leftmost segment (L_7), followed by the adjacent element (L_6), and finally the bottom component (L_8).

The newspaper layout in Figure 4.11a represents a difficult situation. The system correctly identifies the document title, "PLANO A," as the starting point (L_1), reflecting its dominant position in the layout. The subtitle is then processed as L_2, after which the algorithm proceeds to the information columns. In this case, the heuristic produces

a zig-zag sequence, it reads the sidebar header (L_3), going to the adjacent central header "DBRS" (L_4), and only then returns to the sidebar body text (L_5). This ordering error ($L_3 \rightarrow L_4 \rightarrow L_5$) happened because the central header (L_4) is positioned slightly higher along the y-axis than the sidebar text block (L_5). As a result, the heuristic prioritizes global geometric ordering over column level semantic continuity.

In general, the view of the visual sequence ($L_1 \ldots L_n$) indicates that the existing heuristics are more accurate for single-column layouts or structures which are limited to strict vertical order. Under these conditions, the method correctly approximates a linear reading flow. In contrast, its performance degrades in multicolumn layouts, where global y-axis sorting proves insufficient to preserve local column integrity, as evidenced by the observed zig-zag ordering.

4.3.4 Comparison with the Production System

A comparative evaluation was conducted against the production process used at TWA - Tagus World Analytics, which combines PaddleOCR for text extraction with Paddle Layout Detection for region segmentation. The evaluation compares layout detection quality and the resulting OCR output across a set of magazine pages and newspapers, in this evaluation, both approaches used PaddleOCR for text extraction and, both process, document images and produce JSON outputs with the detected regions and their associated textual content.

As shown in Figure 4.12a, the existing system failed to segment a text over graphic banner (region 2), which resulted in OCR mistake (e.g., "IAIASISAAI"). In the same example, the bounding box for "A PROMOÇÃO" was not detected. By contrast, the proposed solution,Figure 4.12b, segmented the banner and recovered the intended text ("FERIAS..."), represented by L_3. The proposed process also detected the "A PROMOÇÃO" region, along with other regions that the existing system detected. The Figure 4.12a and Figure 4.12b show that the proposed method reduces errors on text over

graphics. However, it does not provide the same explicit semantic labels as the production system, so any application that needs detailed content types would still require an additional classification step.

(a) Existing system result.

(b) Proposed solution result.

(c) Existing system result.

(d) Proposed solution result.

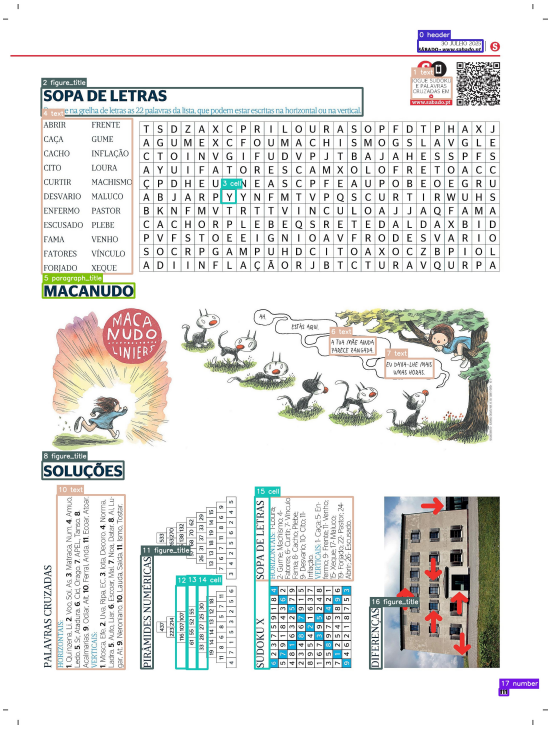
Figure 4.12: Comparative Production System vs. Proposed Solution.

The fragmentation is visible in Figure 4.12c, where multiple bounding boxes split continuous text into smaller segments. The proposed solution tends to aggregate vertically adjacent text blocks, Figure 4.12d, which maintain regions that better preserve the spatial continuity of the document. In this example, the predicted coordinates capture the relevant text extent with fewer discontinuities.

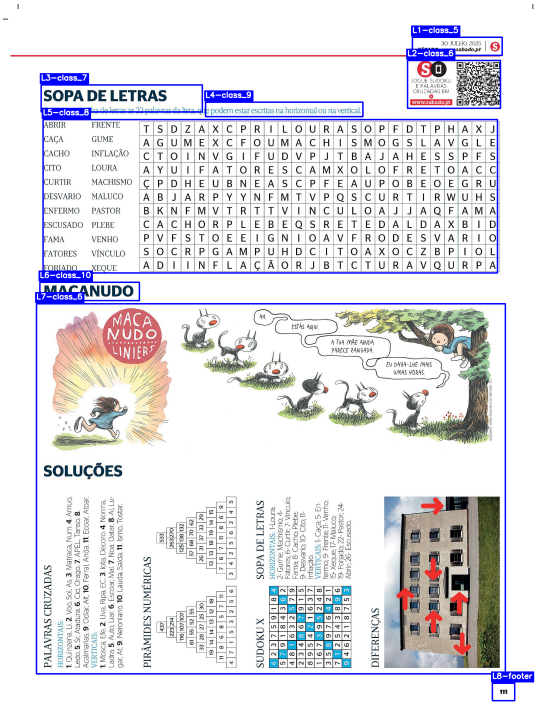
Limitations in layout analysis become more apparent in layouts such as puzzle and game sections, as shown in Figure 4.13a. In Figure 4.13b, the proposed system generates expansive regions that cover a portion of the page, and it also splits the L_3 and L_4 bounding boxes, which could be represented as a single region in this case. The same example shows that the proposed method can represent the puzzle grid as a single coordinate zone, while failing to separate internal components, this reduces its utility for detailed content extraction.

A similar case appears in Figure 4.13d, where the proposed system arranged the entire main article into a single region, including the headline, column text, main photograph, the author's photograph ("Carlos Rodrigues"), and the highlighted quotation ("Em defesa..."). Two distinct footer articles are also merged, and the right sidebar is not fully segmented. In contrast, Figure 4.13c illustrates how the current system breaks the page down into smaller sections. As a result, the body text is not handled as a single block, however, the right column is nearly completely divided without maintaining the article's reading continuity or the title area.

When content regions are detected incorrectly, especially if a single bounding box covers more than one column, the OCR engine reads all the text inside that box in a simple sequence (left-to-right and then top-to-bottom). It consequently mixes text from different columns instead of following the intended layout.



(a) Existing system result.



(b) Proposed solution result.

(c) Existing system result.

(d) Proposed solution result.

Figure 4.13: Extended comparative Production System vs. Proposed Solution.

Across the examples, the proposed solution produces cleaner text in some cases, while the existing system is more reliable in others.

4.4 Discussion of Results

This section evaluates the system's performance in relation to the project objectives presented in chapter 1.

4.4.1 Alignment with Objectives

The goal of this dissertation was to create a OCR process that could extract, segment, and organize data from newspapers with various layouts. The results are therefore mapped to the objectives defined in section 1.2.

The objective of studying and analyzing state-of-the-art technologies was achieved through the review presented in chapter 2. That review showed the evolution from methods based in rules to deep learning approaches, with particular attention to the YOLO family for object detection. Based on this analysis, YOLOv12 and PaddleOCR were selected as the primary tools for this work.

The goal of creating a layout detecting module was accomplished, although the training technique had an impact on performance. The work implemented a detection module based on the YOLO architecture. The model pre-trained on the DocLayout dataset achieved a mAP@50 of 0.934. The model trained on the public DocLayNet subset reached a mAP@50 of 0.728. The model trained on the custom dataset, on the other hand, had low recall and a mAP@50 of 0.519, indicating low generalization. The limited dataset size (151 images) and class imbalance plausibly contributed to this outcome. The fusion strategy in Algorithm 4 mitigated part of these weaknesses.

The objective of integrating text extraction with structural analysis was achieved. The process combined the visual segments detected by YOLO with text extracted by PaddleOCR, using the spatial association logic defined in Algorithm 3. The OCR engine occasionally produced erroneous text in areas dominated by high frequency noise or

textured backgrounds, what made integration difficult.

The goal related to reading order heuristics was partially achieved. The sorting heuristics in Algorithm 1 and Algorithm 2 reconstructed a reading flow for layouts with single column and documents with a clear vertical hierarchy. When headers were slightly misaligned vertically, limitations appeared in multi-column layouts, indicating that geometric heuristics alone are not adequate.

The objective of producing a structured output was achieved. The system generates a JSON file that preserves bounding box coordinates together with the extracted text content. This output converts the source image into a semi-structured representation.

Chapter 5

Conclusion and Future Work

This dissertation research and developed a solution that transforms unstructured content from newspapers and magazines into a semi-structured format. The investigation identified limitations in traditional OCR systems, which fail to preserve the logical reading order when layouts are complex. The study examined deep-learning approaches, with the evolution of CNN architectures and the YOLO family for object detection.

The proposed solution integrates DLA and text extraction through a multi-stage process. The YOLO models, YOLOv10 and YOLOv12, were used to detect and segment visual elements, the Algorithm 4 create a fusion of the models to combine predictions from multiple detectors. The PaddleOCR then extracts text within the selected regions, and a post-processing heuristic create a logical reading order.

The test validation demonstrated that the quality and size of the training dataset are the most crucial factors for performance. The model pre-trained on the DocLayout dataset achieved a high mAP@50 of 0.934 [60]. In comparison, the model trained on a subset of the public DocLayNet dataset achieved a mAP@50 of 0.728, while the model trained on a smaller, custom dataset achieved a mAP@50 of 0.519.

This dissertation identified opportunities that can be improved. Although the proposed approach can structure newspaper layouts, future work can address the current limitations.

The post-processing stage should also receive special attention, especially the logic

used to merge detections from multiple models. The present fusion approach depends on geometric heuristics to deal with conflicts between pre-trained YOLOv10 and custom YOLOv12 models. Future research also should explore non-heuristic methods that infer which detection to keep when overlaps happen.

The YOLOv12m was selected to balance inference speed and accuracy, but future iterations could benchmark YOLOv11 and other YOLOv12 version to check whether larger configurations yield consistent gains under the same deployment constraints.

The current implementation uses a multilingual model from PaddleOCR. To reduce errors associated with the design observed in this study, future work should evaluate alternative OCR engines and fine-tuning using representative newspaper data.

This work separates visual layout analysis from text extraction. Performance may improve by integrating multimodal Large Language Models (LLMs) or Vision Language Models (VLMs) that jointly model visual regions and textual content. Such models could mitigate semantic order of reading failures by inferring the logical flow of articles rather than on geometric sorting.

References

[1] T. Shehzadi, D. Stricker, and M. Z. Afzal, *A hybrid approach for document layout analysis in document images*, 2024. arXiv: 2404.17888 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2404.17888.

[2] M. M. B. A. N. Akanda, M. Ahmed, A. S. A. Rabby, and F. Rahman, "Optimum deep learning method for document layout analysis in low resource languages," in *Proceedings of the 2024 ACM Southeast Conference*, ser. ACMSE '24, Marietta, GA, USA: Association for Computing Machinery, 2024, pp. 199–204, ISBN: 9798400702372. DOI: 10.1145/3603287.3651184. [Online]. Available: https://doi.org/10.1145/3603287.3651184.

[3] S. Lokeshwar, V. R. MK, S. K. PS, V. G. Gowda, and P. Hemavathi, "Analog document search using crnn and keyphrase extraction," *International Journal of Image, Graphics and Signal Processing*, vol. 14, no. 2, p. 16, 2021.

[4] R. P. Kaur, M. K. Jindal, and M. Kumar, "Text and graphics segmentation of newspapers printed in gurmukhi script: A hybrid approach," *The Visual Computer*, vol. 37, no. 7, pp. 1637–1659, 2021. DOI: 10.1007/s00371-020-01927-0.

[5] A. Kumar and G. S. Lehal, "Layout detection of punjabi newspapers using the yolov8 model," *International Journal of Performability Engineering*, vol. 20, no. 3, 186, p. 186, 2024. DOI: 10.23940/ijpe.24.03.p7.186193. [Online]. Available: https://www.ijpe-online.com/EN/abstract/article_4849.shtml.

[6] J. Li, Y. Xu, T. Lv, L. Cui, C. Zhang, and F. Wei, *Dit: Self-supervised pre-training for document image transformer*, 2022. arXiv: 2203.02378 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2203.02378.

[7] W. Zhu, N. Sokhandan, G. Yang, S. Martin, and S. Sathyanarayana, *Docbed: A multi-stage ocr solution for documents with complex layouts*, 2022. arXiv: 2202.01414 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2202.01414.

[8] E. Borenstein, E. Sharon, and S. Ullman, "Combining top-down and bottom-up segmentation," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, 2004, pp. 46–46. DOI: 10.1109/CVPR.2004.314.

[9] N. Girdhar, M. Coustaty, and A. Doucet, "Digitizing history: Transitioning historical paper documents to digital content for information retrieval and mining—a comprehensive survey," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 5, pp. 6151–6180, 2024. DOI: 10.1109/TCSS.2024.3378419.

[10] H. Cheng, P. Zhang, S. Wu, *et al.*, *M^{6}doc: A large-scale multi-format, multi-type, multi-layout, multi-language, multi-annotation category dataset for modern document layout analysis*, 2023. arXiv: 2305.08719 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2305.08719.

[11] M. I. H. Shihab, M. R. Hasan, M. R. Emon, *et al.*, *Badlad: A large multi-domain bengali document layout analysis dataset*, 2023. arXiv: 2303.05325 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2303.05325.

[12] A. Fateh, M. Fateh, and V. Abolghasemi, "Enhancing optical character recognition: Efficient techniques for document layout analysis and text line detection," *Engineering Reports*, vol. 6, no. 9, e12832, 2024. DOI: https://doi.org/10.1002/eng2.12832. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/eng2.12832. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/eng2.12832.

[13] X. Zhong, J. Tang, and A. Jimeno Yepes, "Publaynet: Largest dataset ever for document layout analysis," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 1015–1022. DOI: 10.1109/ICDAR.2019.00166.

[14] B. Pfitzmann, C. Auer, M. Dolfi, A. S. Nassar, and P. Staar, "Doclaynet: A large human-annotated dataset for document-layout segmentation," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22, ACM, Aug. 2022, pp. 3743–3751. DOI: 10.1145/3534678.3539043. [Online]. Available: http://dx.doi.org/10.1145/3534678.3539043.

[15] M. Li, Y. Xu, L. Cui, *et al.*, *Docbank: A benchmark dataset for document layout analysis*, 2020. arXiv: 2006.01038 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2006.01038.

[16] M. Swapna, Y. K. Sharma, and B. Prasadh, "Cnn architectures: Alex net, le net, vgg, google net, res net," *Int. J. Recent Technol. Eng*, vol. 8, no. 6, pp. 953–960, 2020.

[17] Z. Zhao, H. Kang, B. Wang, and C. He, *Doclayout-yolo: Enhancing document layout analysis through diverse synthetic data and global-to-local adaptive perception*, 2024. arXiv: 2410.12628 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2410.12628.

[18] K. B. Shireesha, P. Thejas, and S. A. Kumar, "Document analysis using deep learning techniques," in *2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2023, pp. 1568–1572. DOI: 10.1109/ICESC57686.2023.10193742.

[19] M. Hussain and R. Khanam, "In-depth review of yolov1 to yolov10 variants for enhanced photovoltaic defect detection," *Solar*, vol. 4, no. 3, pp. 351–386, 2024, ISSN: 2673-9941. DOI: 10.3390/solar4030016. [Online]. Available: https://www.mdpi.com/2673-9941/4/3/16.

[20] U. Karn, *An intuitive explanation of convolutional neural networks*, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/, Accessed on 19 July 2025, 2017.

[21] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, *Activation functions in deep learning: A comprehensive survey and benchmark*, 2022. arXiv: 2109.14545 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2109.14545.

[22] M. H. F. d. Morais, *Disease detection and mapping in olive groves using uavs and deep learning for precision agriculture*, 2025.

[23] J. Wang, K. Hu, and Q. Huo, *Unihdsa: A unified relation prediction approach for hierarchical document structure analysis*, 2025. arXiv: 2503.15893 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2503.15893.

[24] G. Kim, T. Hong, M. Yim, *et al.*, *Ocr-free document understanding transformer*, 2022. arXiv: 2111.15664 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.15664.

[25] A. Sundaresan Geetha, M. A. R. Alif, M. Hussain, and P. Allen, "Comparative analysis of yolov8 and yolov10 in vehicle detection: Performance metrics and model efficacy," *Vehicles*, vol. 6, no. 3, pp. 1364–1382, 2024, ISSN: 2624-8921. DOI: 10.3390/vehicles6030065. [Online]. Available: https://www.mdpi.com/2624-8921/6/3/65.

[26] N. Arianna, R. Riccardo, P. Gabriele, and R. Giulio, "From unstructured documents to annotated information: An optimized pipeline to process industrial requirements," in *Proceedings of the 2024 8th International Conference on Natural Language Processing and Information Retrieval*, ser. NLPIR '24, Association for Computing Machinery, 2025, pp. 272–278, ISBN: 9798400717383. DOI: 10.1145/3711542.3711576. [Online]. Available: https://doi.org/10.1145/3711542.3711576.

[27] Z. Zhong, W. Jiawei, H. Sun, *et al.*, "A hybrid approach to document layout analysis for heterogeneous document images," in Aug. 2023, pp. 189–206, ISBN: 978-3-031-41733-7. DOI: 10.1007/978-3-031-41734-4_12.

[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[29] Y. Huang, Q. Yan, Y. Li, *et al.*, "A yolo-based table detection method," Sep. 2019, pp. 813–818. DOI: 10.1109/ICDAR.2019.00135.

[30] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[31] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: 1804.02767 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1804.02767.

[32] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2004.10934.

[33] C. Li, L. Li, H. Jiang, *et al.*, *Yolov6: A single-stage object detection framework for industrial applications*, 2022. arXiv: 2209.02976 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2209.02976.

[34] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, *Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, 2022. arXiv: 2207.02696 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2207.02696.

[35] M. A. R. Alif and M. Hussain, *Yolov1 to yolov10: A comprehensive review of yolo variants and their application in the agricultural domain*, 2024. arXiv: 2406.10139 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2406.10139.

[36] N. Jegham, C. Y. Koh, M. Abdelatti, and A. Hendawi, *Yolo evolution: A comprehensive benchmark and architectural review of yolov12, yolo11, and their previous versions*, 2025. arXiv: 2411.00201 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2411.00201.

[37] A. Wang, H. Chen, L. Liu, *et al.*, *Yolov10: Real-time end-to-end object detection*, 2024. arXiv: 2405.14458 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2405.14458.

[38] Z. Zong, G. Song, and Y. Liu, *Detrs with collaborative hybrid assignments training*, 2023. arXiv: 2211.12860 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2211.12860.

[39] Y. Chen, Q. Chen, Q. Hu, and J. Cheng, *Date: Dual assignment for end-to-end fully convolutional object detection*, 2022. arXiv: 2211.13859 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2211.13859.

[40] F. Chollet, *Xception: Deep learning with depthwise separable convolutions*, 2017. arXiv: 1610.02357 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1610.02357.

[41] Y. Tian, Q. Ye, and D. Doermann, *Yolov12: Attention-centric real-time object detectors*, 2025. arXiv: 2502.12524 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2502.12524.

[42] X. Yin, Z. Zhao, and L. Weng, "Mas-yolo: A lightweight detection algorithm for pcb defect detection based on improved yolov12," *Applied Sciences*, vol. 15, no. 11, 2025, ISSN: 2076-3417. DOI: 10.3390/app15116238. [Online]. Available: https://www.mdpi.com/2076-3417/15/11/6238.

[43] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, *Going deeper with image transformers*, 2021. arXiv: 2103.17239 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.17239.

[44] M. A. R. Alif and M. Hussain, *Yolov12: A breakdown of the key architectural features*, 2025. arXiv: 2502.14740 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2502.14740.

[45] E. D. Fajrianti, Y. Y. F. Panduman, N. Funabiki, A. L. Haz, K. C. Brata, and S. Sukaridhoto, "A user location reset method through object recognition in indoor navigation system using unity and a smartphone (insus)," *Network*, vol. 4, no. 3, pp. 295–312, 2024, ISSN: 2673-8732. DOI: 10.3390/network4030014. [Online]. Available: https://www.mdpi.com/2673-8732/4/3/14.

[46] A. Biró, A. I. Cuesta-Vargas, J. Martín-Martín, L. Szilágyi, and S. M. Szilágyi, "Synthetized multilanguage ocr using crnn and svtr models for realtime collaborative tools," *Applied Sciences*, vol. 13, no. 7, 2023, ISSN: 2076-3417. DOI: 10.3390/app13074419. [Online]. Available: https://www.mdpi.com/2076-3417/13/7/4419.

[47] R. Ahmed, N. Shabbir, M. W. Raza, A. Zeb, and H. Elahi, "Evaluation of model degradation in paddleocr, ultocr, and trocr across baseline and tensorflow lite environments," in *2024 International Conference on Robotics and Automation in Industry (ICRAI)*, 2024, pp. 1–5. DOI: 10.1109/ICRAI62391.2024.10894257.

[48] H. Qin, J. Jin, W. Sun, Y. Qian, H. Wang, and Z. Wang, "An analysis method for recognizing and analyzing multi-type documents in electric power operation site based on paddleocr," Oct. 2024, pp. 363–367. DOI: 10.1109/ICSGSC62639.2024.10813769.

[49] C. Cui, T. Sun, M. Lin, *et al.*, *Paddleocr 3.0 technical report*, 2025. arXiv: 2507.05595 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2507.05595.

[50] W. Hu, X. Cai, J. Hou, S. Yi, and Z. Lin, *Gtc: Guided training of ctc towards efficient and accurate scene text recognition*, 2020. arXiv: 2002.01276 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2002.01276.

[51] Y. Du, Z. Chen, C. Jia, *et al.*, *Svtr: Scene text recognition with a single visual model*, 2022. arXiv: 2205.00159 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2205.00159.

[52] C. Li, W. Liu, R. Guo, *et al.*, *Pp-ocrv3: More attempts for the improvement of ultra lightweight ocr system*, 2022. arXiv: 2206.03001 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2206.03001.

[53] A. Morris, V. Maier, and P. Green, "From wer and ril to mer and wil: Improved evaluation measures for connected speech recognition.," Jan. 2004.

[54] A. Morris, V. Maier, and P. Green, "From wer and ril to mer and wil: Improved evaluation measures for connected speech recognition," in *Proceedings of Interspeech 2004*, 2004, pp. 2765–2768. DOI: 10.21437/Interspeech.2004-668.

[55] M. A. N. Hadi, M. Gul, M. Khan, G. N. Alwakid, and N. Z. Jhanjhi, "Benchmarking performance analysis of optical character recognition techniques," in *2024 26th International Multi-Topic Conference (INMIC)*, IEEE, 2024, pp. 1–6.

[56] S. G. E. Brucal, L. C. M. de Jesus, S. R. Peruda, L. A. Samaniego, and E. D. Yong, "Development of tomato leaf disease detection using yolov8 model via roboflow 2.0," in *2023 IEEE 12th Global Conference on Consumer Electronics (GCCE)*, 2023, pp. 692–694. DOI: 10.1109/GCCE59613.2023.10315251.

[57] D. Kholiya, A. K. Mishra, N. K. Pandey, and N. Tripathi, "Plant detection and counting using yolo based technique," in *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, 2023, pp. 1–5. DOI: 10.1109/ASIANCON58793.2023.10270530.

[58] M. Pavithra, P. Surya Karthikesh, B. Jahnavi, M. Navyalokesh, R. C, and K. Lokesh Krishna, "Implementation of enhanced security system using roboflow," in *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2024, pp. 1–5. DOI: 10.1109/ICRITO61523.2024.10522313.

[59] *Nvidia tesla t4 gpu specifications*, https://www.techpowerup.com/gpu-specs/tesla-t4.c3316, Consultado em 7 de agosto de 2025, TechPowerUp, 2025.

[60] J. Zhao, *Doclayout-yolo-docstructbench-imgsz1280-2501*, https://huggingface.co/juliozhao/DocLayout-YOLO-DocStructBench-imgsz1280-2501, Accessed: 2025-08-18, 2024.

[61] J. Zhao, *Doclayout-yolo-doclaynet-from_scratch*, https://huggingface.co/juliozhao/DocLayout-YOLO-DocLayNet-from_scratch, Accessed: 2025-08-18, 2024.

[62] B. Pfitzmann, C. Auer, M. Dolfi, A. S. Nassar, and P. W. J. Staar, "Doclaynet: A large human-annotated dataset for document-layout analysis," 2022. DOI: 10.1145/3534678.353904. [Online]. Available: https://arxiv.org/abs/2206.01062.

Appendix A

Post-Processing Algorithms

Algorithm 4 Multi-Model Layout Detection with Overlap Resolution

Input: Image I; Models M_1, M_2, M_3

Output: Final detections B_{final}

Phase 1: Detect and Clean (for each model)

    Run detection with confidence threshold

    Filter small boxes (area, width, height thresholds)

    Filter overly large boxes ($> 50\%$ of image)

    Remove nested boxes (containment $> 80\%$)

Phase 2: Progressive Merging

    $B_{base} \leftarrow B_3$ (YOLOv10)

    Merge B_2 into base, remove nested

    Merge B_1 into combined, remove nested

Phase 3: Overlap Resolution

    Subtract intersections to resolve overlaps

return B_{final}

Algorithm 5 RemoveNestedBoxes

Input: Boxes B; containment threshold θ_{cont}

Output: Filtered boxes B'

Step 1: Sort boxes by area (largest first)

Step 2: Initialize all boxes as "keep"

Step 3: Detect and mark nested boxes

    for *each box* b_i *(largest to smallest)* **do**

        for *each smaller box* b_j *after* b_i **do**

            Compute containment ratio: $\frac{\text{Intersection}(b_i, b_j)}{\text{Area}(b_j)}$

            if $ratio \geq \theta_{cont}$ **then**

                Mark b_j for removal

Step 4: Return boxes marked as "keep"

Algorithm 6 MergeBoxesIntoBase

Input: Base boxes B_{base}; New boxes B_{new}; Thresholds $\theta_{\text{cont}}, \theta_{\text{merge}}$

Output: Merged boxes B_{result}

Step 1: Initialize result with base boxes

Step 2: Process each new box

```
foreach new box b_new do
    Find base box with maximum containment ratio
    if containment ≥ θ_cont (e.g., 0.8) then
        Skip (already represented)
    else if containment ≥ θ_merge (e.g., 0.5) then
        if base box not yet merged then
            Expand base box to encompass both
            Average confidence scores
            Mark base box as merged
        else
            Add new box separately
    else
        Add new box separately
```

return B_{result}

Algorithm 7 SubtractIntersections

Input: Boxes B; Image size (w, h); Width threshold θ_w

Output: Adjusted boxes B'

```
Initialize all boxes as "not processed"
foreach box b_i in B do
    if b_i already processed then
        skip
    Set b_i as current box
    foreach other box b_j in B do
        if b_i and b_j intersect then
            Adjust current box to remove intersection
            if adjusted box too small then
                Mark b_i as eliminated  Break
            else
                Update current box with new boundaries
    if b_i not eliminated then
        Add to result set
return B'
```

Algorithm 8 CutBox: Geometric Subtraction

Input: Large box b_L; Small box b_S; Padding $p = 15$

Output: Adjusted box or null

Step 1: Calculate intersection between b_L and b_S

    if *no intersection* **then**

        return b_L unchanged

Step 2: Generate candidate cuts (up to 4);

    Try cutting from each edge of intersection:

        - Left: keep region right of intersection + padding

        - Right: keep region left of intersection + padding

        - Top: keep region below intersection + padding

        - Bottom: keep region above intersection + padding

Step 3: Select best candidate

    if *no valid candidates* **then**

        return null

    Choose candidate with maximum area **if** *area < 15% of original* **then**

        return null

return best adjusted box

Appendix B

Example of JSON output


```
{
{
  "1": {
    "cls": "class_7",
    "bbox": [
      141,
      226,
      3076,
      401
    ],
    "text": "OPERADORAS CONDENADAS A DEVOLVER 40 MILHOES A CONSUMIDORES P.12"
  },
  "2": {
    "cls": "text",
    "bbox": [
      1063,
      430,
      1645,
      502
    ],
```

```
    "text": "Segunda-feira,22.09.2025·Diário €1,50(C/IVA)"
  },
  "3": {
    "cls": "class_6",
    "bbox": [
      144,
      522,
      1570,
      1034
    ],
    "text": "CORREIO www.cmjornal.pt da manha
    ↪ DIRETOR-GERALEDITORIAL:CARLOSRODRIGUES
    ↪ DIRETORES-GERAISEDITORIAISADJUNTOS:ARMANDOESTEVESPEREIRA,
    ↪ ARRRIRRRAEIRRNRAERRESRRATRSRRIER"
  },
  "4": {
    "cls": "class_6",
    "bbox": [
      1654,
      528,
      3130,
      1072
    ],
    "text": "MULHER MORRE A CAMINHO DO SANTUÁRIO P.15 Milhares de motards rezam
    ↪ em Fatima"
  },
  "5": {
    "cls": "class_6",
    "bbox": [
      92,
```

```
        1164,
        1366,
        1541
      ],
      "text": "POLEMICA P.24 ISALTINO E VENTURA EM GUERRA ABERTA
      ↪  \"Andrézito,monstrinho\".\"Sim, sou persigo os corruptos"
    },
    "6": {
      "cls": "class_6",
      "bbox": [
        1454,
        1145,
        2467,
        2071
      ],
      "text": "Leoes obrigados a vencer hoje em Alvalade P.4e5"
    },
    "7": {
      "cls": "class_7",
      "bbox": [
        93,
        1675,
        597,
        1720
      ],
      "text": "VILAFRANCADEXIRAP19"
    },
    "8": {
      "cls": "class_6",
      "bbox": [
```

```
        2556,
        1166,
        3131,
        2353
      ],
      "text": "PAULO NUNCIO P.44 Deputado colhido por bezerro em pega"
    },
    "9": {
      "cls": "class_6",
      "bbox": [
        1337,
        2175,
        1501,
        2422
      ],
      "text": ""
    },
    "10": {
      "cls": "class_9",
      "bbox": [
        88,
        1735,
        1322,
        3259
      ],
      "text": "MEDICO Seguia jogo ou telemovel enquantoa atendia A VER BOLA
      ↪ doente urgente ERRA FOI AO HOSPITAL e saiu de lá como entrou.\"Mostra
      ↪ oestado a que chegou o SNS\", DIAGNOSTICO diz um familiar revoltado
      ↪ Utente teve de recorrer oP privado, onde Ihe foi detetado ombro
      ↪ congelado' PUBLICIDADE"
```

```
  },
  "11": {
    "cls": "class_10",
    "bbox": [
      1540,
      2153,
      2458,
      2859
    ],
    "text": "SPORTING-MOREIRENSE 20H15,SPORT TV1 RUI BORGES VARANDAS ESCONDEU
    ↪  TELEFONEMA DE MOURINHO"
  },
  "12": {
    "cls": "class_7",
    "bbox": [
      2562,
      2446,
      3014,
      2673
    ],
    "text": "ACUSADOS P.10 REDEABRE PORTA A ILEGAIS PORSMILEUROS"
  },
  "13": {
    "cls": "class_7",
    "bbox": [
      2561,
      2702,
      3063,
      2929
    ],
```

```
    "text": "CASCAIS P.14 PUBLICAViDEO A QUEIMARROUPA DA MULHER"
  },
  "14": {
    "cls": "class_9",
    "bbox": [
      2556,
      2952,
      3094,
      3213
    ],
    "text": "INQUERITO P.8 PJINVESTIGA CASA DE LUXO DEGOVERNANTE"
  },
  "15": {
    "cls": "class_7",
    "bbox": [
      2561,
      3220,
      2683,
      3263
    ],
    "text": "PUBLICIDADE"
  },
  "16": {
    "cls": "class_6",
    "bbox": [
      93,
      3274,
      657,
      3937
    ],
```

```
      "text": "ERMELINDA FAZ PARTE DA SUA VIDA
      ↪  TRMELINDERMELINDERMELINDAERMELINDBMELIND BESEVA  CASA ERMELINDA HL
      ↪  FRETTAS 1020"
    },
    "17": {
      "cls": "class_6",
      "bbox": [
        2572,
        3272,
        3130,
        3943
      ],
      "text": "Precisa de dinheiro? Venda osseus Valores com Opcaodevoltar e
      ↪  Compra-losate24meses! CREIREENN NONN AAURE OUTROSVALORES VQC24
      ↪  VALORES.PT Vendacom Opgao deCompra especialistas em Ouro Chamada p/
      ↪  Custo de € 0,07+IVA redefixanacional por minuto 253 11155 808 256737"
    },
    "18": {
      "cls": "class_6",
      "bbox": [
        707,
        3339,
        2464,
        3962
      ],
      "text": "Vidas P.38 a 41 PNOIEE SEGREDO DA ESTREIAP.6e 7 JA PODE PALESTRA
      ↪  REUNIR AO INTERVALO FAMILIA LIZAJA FOI A CHAVE DEU ALUZ AO VITORIA"
    }
  }
}
```

Appendix C

Repository

GitHub repository: `https://github.com/pr-junior/ocr_model_detection`