UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Segmentación y clasificación del layout de páginas manuscritas con herramientas de código abierto de segmentación de imágenes.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Branchadell Allepuz, Javier

Tutor/a: Pastor Gadea, Moisés

CURSO ACADÉMICO: 2024/2025

Resum

La digitalització massiva de fons documentals manuscrits resulta essencial per a la preservació i difusió del patrimoni històric, però els deterioraments físics i la gran heterogeneïtat cal·ligràfica dificulten l'automatització del procés. Aquest Treball Fi de Grau avalua la viabilitat de dues plataformes de codi obert, DETECTRON2 i YOLOV8, per a la segmentació estructural de pàgines manuscrites. El mètode combina scripts de preprocés i conversió automàtica de dos corpus anotats (*OHG* i *VORAU-253*) amb un pipeline d'avaluació homogeni basat en la mètrica COCO (mAP, AP50, AP75 i AR). S'han entrenat sis variants de models (des de zero, amb pesos pre-entrenats en COCO i en conjunts de domini) i s'han analitzat tant resultats quantitatius com qualitatius. Els experiments mostren que YOLOV8-L ofereix una lleu millora en mAP global respecte de DETECTRON2, mentre que aquest últim conserva un millor recall en classes minoritàries. El treball conclou que les eines de codi obert permeten una segmentació fiable sense requerir entrenaments massius i proposa línies futures com la segmentació de màscares i l'ús de models multimodals.

Paraules clau: Segmentació de documents, Detectron2, YOLOv8, Patrimoni històric, Aprenentatge profund, Digitalització.

Resumen

La digitalización masiva de fondos manuscritos es crucial para conservar y difundir el patrimonio histórico, pero la degradación física y la variabilidad caligráfica dificultan su procesamiento automático. Este Trabajo Fin de Grado evalúa la aplicabilidad de dos frameworks de código abierto, DETECTRON2 y YOLOV8, a la segmentación estructural de documentos manuscritos. Se desarrollaron scripts de preprocesamiento y conversión automática para dos corpus anotados (*OHG* y *VORAU-253*) y se implementó un sistema de evaluación uniforme siguiendo el protocolo COCO (mAP, AP50, AP75 y AR). Se entrenaron seis variantes de modelos (desde cero, con pesos COCO y con pesos de dominio) y se analizaron los resultados tanto cuantitativa como cualitativamente. Los experimentos indican que YOLOV8-L supera ligeramente a DETECTRON2 en mAP global, mientras que DETECTRON2 mantiene mejor recall en ciertas clases infrecuentes. El estudio demuestra que las herramientas open-source pueden lograr segmentaciones fiables sin entrenamientos extensivos y sugiere futuras mejoras como la incorporación de máscaras de instancia y modelos multimodales.

Palabras clave: Segmentación de documentos, Detectron2, YOLOv8, Patrimonio histórico, Aprendizaje profundo, Digitalización.

Abstract

The massive digitisation of handwritten collections is vital for preserving cultural heritage, yet physical degradation and calligraphic variability make automatic processing challenging. This Bachelor's Thesis assesses two open-source frameworks, DETECTRON2 and YOLOV8, for structural page segmentation in historical manuscripts. Custom preprocessing and conversion scripts were built for two annotated corpora (*OHG* and *VORAU-253*); a unified COCO-style evaluation pipeline (mAP, AP50, AP75 and AR) was implemented. Six model variants (from-scratch, COCO-pretrained and domain-pretrained) were trained and analysed both quantitatively and qualitatively. Experiments

show that YOLOv8-L achieves slightly higher overall mAP, whereas DETECTRON2 attains better recall for minority classes. Results confirm that open-source tools can deliver reliable segmentation without extensive training and outline future work such as instance-mask prediction and multimodal models.

Key words: Document segmentation, Detectron2, YOLOv8, Historical heritage, Deep learning, Digitisation.

Índice general

[1] `https://www2.udg.edu/tabid/18216/Default.aspx`
[2] `https://manuscripta.at/`

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

La digitalización masiva de fondos documentales se ha convertido en un pilar esencial para la preservación y difusión del patrimonio histórico y cultural. Sin embargo, buena parte de los documentos manuscritos conservados en archivos notariales, judiciales, eclesiásticos, entre otros, presentan una gran heterogeneidad caligráfica, deterioros físicos y estructuras de página nada estandarizadas, lo que dificulta sobremanera su procesamiento automático.

En este contexto, la segmentación estructural se erige como un paso previo imprescindible para tareas de reconocimiento de texto manuscrito (HTR), recuperación de información o análisis paleográfico.

Los avances recientes en visión por computador, y en particular las arquitecturas de aprendizaje profundo de código abierto como Detectron2 y YOLOv8, han demostrado una notable capacidad para extraer patrones complejos a partir de imágenes, haciendo viable abordar problemas que hasta hace poco dependían de meticulosas intervenciones manuales.

1.1 Motivación

Durante mis prácticas en empresa, pude comprobar de primera mano cómo el proceso de digitalización está transformando la gestión documental en distintos sectores. Esta digitalización no se limita a documentos generados por ordenador, sino que abarca también documentos impresos y, especialmente, documentos manuscritos, cuya complejidad añade un desafío técnico significativo.

En este contexto, surgió mi interés particular por los documentos históricos manuscritos, que representan uno de los mayores retos técnicos en el campo del análisis documental debido a su variabilidad, deterioro y falta de estandarización. La digitalización de estos documentos trasciende la mera conversión de texto manuscrito en texto de ordenador mediante técnicas de Reconocimiento Automático de Texto Manuscrito (HTR en inglés); requiere identificar y segmentar correctamente las diferentes partes que los componen: encabezados, firmas, cuerpos de texto, sellos, anotaciones marginales, entre otros elementos estructurales.

La tarea de segmentación emerge así como el auténtico núcleo del problema. Mientras existen numerosas herramientas que permiten extraer texto una vez se ha delimitado adecuadamente la región de interés, el verdadero valor añadido reside en cómo segmentar correctamente los elementos relevantes del documento, estableciendo claramente a qué corresponde cada parte. Este problema se intensifica en documentos históricos donde el

deterioro físico, las variaciones caligráficas y los formatos no estandarizados complican significativamente el análisis automatizado.

Movido por esta curiosidad, surge el interés en explorar cómo los modelos actuales de aprendizaje profundo, particularmente frameworks como Detectron2 y YOLOv8, pueden abordar estos desafíos. Estos modelos, diseñados originalmente para la detección y segmentación de objetos en fotografías, ofrecen arquitecturas adaptables al problema específico de la segmentación documental. Su principal ventaja radica en permitir trabajar con estructuras complejas sin necesidad de realizar entrenamientos masivos desde cero, aprovechando su capacidad de generalización a partir de modelos preentrenados existentes.

La motivación de este trabajo, por tanto, no solo se fundamenta en el deseo de aplicar técnicas de inteligencia artificial modernas a un problema real de interés social y cultural, sino también en evaluar de manera crítica hasta qué punto estas herramientas de código abierto son efectivas para facilitar la digitalización inteligente de documentos manuscritos. El potencial impacto de este estudio se extiende a múltiples ámbitos, desde la preservación del patrimonio cultural hasta la mejora de procesos de digitalización en entornos administrativos, jurídicos o académicos donde aún abundan los documentos manuscritos históricos.

1.2 Objetivos

El objetivo general de este trabajo es evaluar la aplicabilidad y el rendimiento de herramientas de código abierto basadas en aprendizaje profundo para la segmentación estructural de documentos manuscritos históricos. El enfoque combina el análisis técnico de estas herramientas con su puesta en práctica sobre corpus reales, incorporando tanto métricas cuantitativas como valoraciones cualitativas sobre la utilidad de los resultados.

Para ello, se han definido los siguientes objetivos específicos:

1. **Explorar y analizar herramientas de segmentación de código abierto**, valorando su facilidad de instalación, configuración y uso práctico en entornos locales y en la nube. Este objetivo incluye la selección fundamentada de los modelos empleados en el desarrollo del trabajo.

2. **Desarrollar scripts de preprocesamiento y conversión automática** que permitan adaptar conjuntos de datos históricos anotados (como OHG y VORAU-253) a los formatos requeridos por cada herramienta, garantizando la compatibilidad y coherencia de las anotaciones.

3. **Implementar un sistema automatizado de evaluación**, independiente del modelo utilizado, que calcule métricas homogéneas siguiendo el protocolo COCO (mAP@[.50:.95], AP50, AP75, AR@k), permitiendo así una comparación objetiva y reproducible de los resultados obtenidos.

4. **Evaluar la utilidad práctica de los resultados obtenidos**, complementando el análisis cuantitativo con una valoración cualitativa basada en la inspección visual de las segmentaciones generadas, y analizando su idoneidad para tareas posteriores como recorte, clasificación o reconocimiento de texto manuscrito.

1.3 Planificación

La ejecución de este trabajo se estructuró siguiendo una metodología ágil que permitiera adaptabilidad ante los desafíos inherentes al desarrollo de sistemas de aprendizaje profundo. El cronograma general del TFG se organizó en cinco bloques principales que abarcan desde la investigación inicial hasta la documentación final, tal como se muestra en la Figura 1.1.

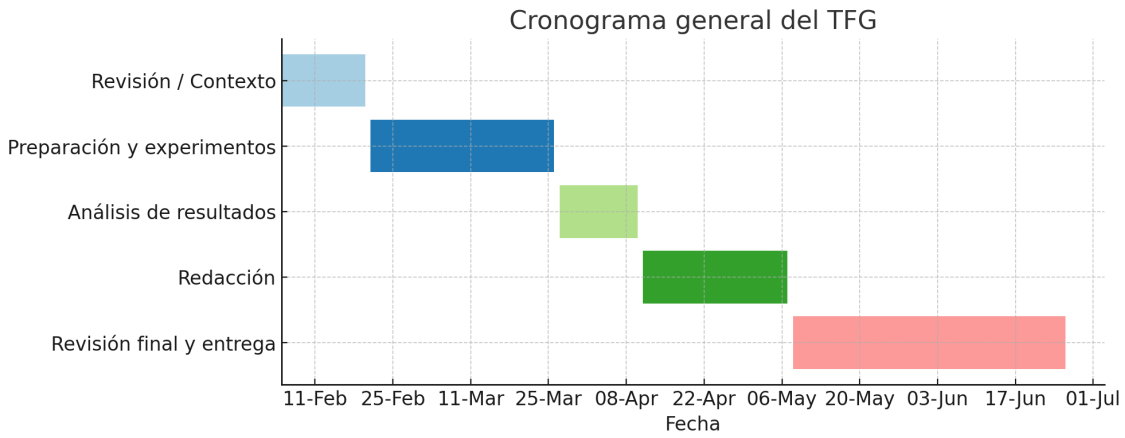

Figura 1.1: Cronograma general del Trabajo Fin de Grado, organizado en cinco bloques principales de desarrollo.

El seguimiento del progreso se realizó mediante herramientas de gestión de proyectos como Trello, complementadas con este diagrama de Gantt para mantener una visión global del desarrollo. Aunque se respetaron en gran medida los plazos establecidos inicialmente, la naturaleza iterativa del desarrollo de modelos de aprendizaje profundo requirió frecuentes revisiones y mejoras de tareas ya finalizadas, especialmente durante las fases de análisis de resultados y redacción de la memoria.

Esta flexibilidad metodológica resultó fundamental para garantizar la calidad del trabajo final, priorizando la coherencia técnica y el rigor científico sobre la rigidez temporal.

1.4 Estructura de la memoria

Esta memoria se estructura en cinco capítulos principales que abordan de manera progresiva el desarrollo del trabajo:

El **Capítulo 1** presenta la introducción al problema, incluyendo la motivación que llevó al desarrollo de este trabajo, los objetivos específicos planteados y la planificación seguida durante su ejecución.

El **Capítulo 2** desarrolla el estado del arte, revisando los fundamentos teóricos necesarios: procesamiento de imágenes, aprendizaje automático, redes neuronales artificiales y análisis de estructura de documentos. Esta base teórica contextualiza las herramientas y métodos empleados.

El **Capítulo 3** detalla la experimentación realizada, describiendo el entorno experimental, las herramientas seleccionadas (Detectron2 y YOLOv8), los corpus utilizados (OHG y VORAU-253), las métricas de evaluación implementadas y el proceso de entrenamiento y evaluación de los modelos.

El **Capítulo 4** presenta las conclusiones del trabajo, analizando la consecución de objetivos, los problemas encontrados durante el desarrollo, las líneas de trabajo futuro identificadas y la relación del proyecto con las competencias del grado.

Finalmente, se incluyen capítulos complementarios sobre los Objetivos de Desarrollo Sostenible y un glosario de términos técnicos empleados.

CAPÍTULO 2
Estado del Arte

Este capítulo presenta una revisión sistemática del estado actual de conocimiento en las áreas fundamentales para la comprensión y análisis de documentos históricos. Se abordan conceptos esenciales como el procesamiento de imágenes, métodos de aprendizaje automático, arquitecturas de redes neuronales, y técnicas específicas para el análisis estructural de documentos. Esta base teórica es fundamental para contextualizar las soluciones propuestas y comprender los desafíos inherentes a la manipulación de documentos históricos.

2.1 Procesamiento de Imágenes

El procesamiento digital de imágenes constituye la base fundamental para cualquier sistema de análisis documental. Esta disciplina comprende el conjunto de técnicas y algoritmos que permiten manipular, analizar y transformar imágenes digitales para extraer información significativa [1]. En el contexto de documentos históricos, estas técnicas son especialmente relevantes debido a los desafíos particulares que presentan estos materiales: deterioro físico, variabilidad en la iluminación, manchas, tinta desvanecida, entre otros.

2.1.1. Fundamentos y Conceptos Básicos

Una imagen digital se representa como una matriz bidimensional de valores de intensidad, donde cada elemento (píxel) codifica información de color o luminosidad [2]. Para documentos históricos, el procesamiento comienza generalmente con la adquisición mediante digitalización, lo que transforma el documento físico en una representación digital manipulable.

El procesamiento de imágenes se fundamenta en la representación del espacio de color (RGB, escala de grises, binario), resolución espacial (medida en píxeles por pulgada o DPI), y profundidad de color (bits por píxel). Estos parámetros determinan tanto la calidad como la complejidad del procesamiento posterior.

2.1.2. Técnicas de Preprocesamiento

El preprocesamiento busca mejorar la calidad de la imagen y facilitar la extracción posterior de características. Las técnicas más relevantes para documentos históricos incluyen:

- **Normalización y mejora de contraste**: Métodos como la ecualización de histograma y sus variantes adaptativas permiten mejorar el contraste global y local, fundamentales para resaltar texto desvanecido [3].
- **Binarización**: La conversión de imágenes en escala de grises a formato binario mediante umbrales globales como el método de Otsu o adaptativos como Sauvola, facilita la identificación de texto y regiones de interés.
- **Reducción de ruido**: Aplicación de filtros como el gaussiano o el de mediana para eliminar imperfecciones puntuales preservando bordes y estructuras fundamentales.
- **Corrección de la pendiente**: Detección y corrección de la inclinación del documento mediante transformaciones geométricas, frecuentemente basadas en la transformada de Hough.

Estas operaciones de preprocesamiento no son etapas aisladas sino complementarias, y su aplicación secuencial construye un pipeline robusto que compensa las deficiencias inherentes a los documentos históricos.

2.2 Aprendizaje Automático

El aprendizaje automático proporciona el marco metodológico para desarrollar sistemas capaces de aprender patrones a partir de datos, sin necesidad de programación explícita de reglas [4]. En el contexto del análisis documental, estos métodos permiten abordar tareas complejas como la clasificación de regiones, reconocimiento de texto o interpretación estructural.

2.2.1. Paradigmas Fundamentales

Los enfoques de aprendizaje automático se categorizan según la naturaleza del aprendizaje y la información disponible durante el entrenamiento:

- **Aprendizaje supervisado**: Utilizando conjuntos de datos etiquetados, estos algoritmos aprenden a mapear características de entrada a salidas deseadas. En documentos históricos, se aplica para clasificación de regiones, reconocimiento de caracteres o detección de elementos estructurales específicos.
- **Aprendizaje no supervisado**: Centrado en descubrir patrones intrínsecos sin etiquetas previas, resulta valioso para agrupamiento de documentos similares o descubrimiento de layouts comunes en corpus extensos.
- **Aprendizaje por refuerzo**: Basado en la optimización de políticas mediante recompensas, este paradigma tiene aplicaciones emergentes en la secuenciación de operaciones de procesamiento adaptativo.
- **Aprendizaje semi-supervisado**: Combinando datos etiquetados y no etiquetados, permite abordar escenarios con etiquetado parcial, frecuentes en colecciones históricas donde solo una pequeña fracción puede ser anotada manualmente.

La selección del paradigma adecuado depende de factores como la disponibilidad de datos etiquetados, la naturaleza de la tarea y los recursos computacionales disponibles.

2.2.2. Métodos Clásicos Relevantes para Análisis Documental

Antes del auge de las técnicas de aprendizaje profundo, diversos métodos clásicos demostraron efectividad en el análisis de documentos [5]:

- **Máquinas de Vectores Soporte (SVM)**: Destacan por su capacidad para crear hiperplanos de separación óptimos en espacios de características de alta dimensionalidad, aplicándose exitosamente a la clasificación de componentes documentales.
- **Bosques Aleatorios**: Estos ensambles de árboles de decisión ofrecen robustez y capacidad interpretativa, valiosos para la selección de características relevantes en análisis documental.
- **Algoritmos de Agrupamiento**: K-means y métodos jerárquicos facilitan la identificación no supervisada de regiones coherentes en documentos basándose en similitudes visuales o espaciales.
- **Modelos de Mezclas Gaussianas**: Permiten modelar distribuciones complejas de características, útiles para la segmentación adaptativa de regiones documentales.
- **Campos Aleatorios Condicionales (CRF)**: Extienden los modelos gráficos probabilísticos a datos secuenciales, capturando dependencias espaciales entre etiquetas de regiones adyacentes.

Estos métodos tradicionales siguen siendo relevantes, especialmente cuando los datos de entrenamiento son limitados o cuando la interpretabilidad del modelo es prioritaria.

2.2.3. Evaluación y Métricas

La evaluación rigurosa de los métodos de aprendizaje automático para análisis documental requiere métricas específicas que reflejen tanto la precisión a nivel de píxel como la coherencia estructural:

- **Métricas basadas en píxeles**: Precisión, exhaustividad, F1-score y exactitud evalúan la correspondencia píxel a píxel entre segmentaciones predichas y anotaciones manuales.
- **Métricas basadas en regiones**: Intersección sobre Unión (IoU) y variantes como mIoU (mean IoU) cuantifican el solapamiento entre regiones predichas y regiones de referencia.
- **Métricas de error estructural**: Evalúan la preservación de relaciones jerárquicas o espaciales entre componentes, fundamentales para capturar la organización lógica del documento.
- **Métricas específicas de dominio**: Como la precisión en la detección de líneas base para documentos manuscritos o la correcta identificación de tablas y sus celdas.

Estas métricas deben complementarse con validación cruzada y evaluación en conjuntos de datos representativos para garantizar la generalización a documentos no vistos durante el entrenamiento.

2.3 Redes Neuronales Artificiales

Las redes neuronales artificiales han revolucionado el procesamiento de imágenes y el análisis documental, proporcionando capacidades de modelado que superan significativamente a los métodos tradicionales en tareas complejas [6]. Su capacidad para aprender representaciones jerárquicas de características las hace especialmente adecuadas para documentos históricos con gran variabilidad.

2.3.1. Arquitecturas Fundamentales

La diversidad de arquitecturas neuronales permite abordar diferentes aspectos del análisis documental [7]:

- **Redes Neuronales Feedforward**: Constituyen la base conceptual de arquitecturas más complejas, con capas densamente conectadas que aprenden representaciones abstractas progresivas.
- **Redes Neuronales Convolucionales (CNN)**: Fundamentales para el procesamiento de imágenes, explotan la localidad espacial y las invarianzas traslacionales mediante filtros convolucionales compartidos.
- **Redes Recurrentes (RNN, LSTM, GRU)**: Capturan dependencias secuenciales, útiles para modelar texto, líneas de escritura o la estructura secuencial de documentos.
- **Modelos basados en Atención y Transformers**: Representan el estado del arte actual para modelado multidimensional, permitiendo capturar relaciones de largo alcance entre elementos documentales distantes.
- **Autoencoders y Redes Generativas**: Facilitan el aprendizaje no supervisado de representaciones compactas y la generación de nuevos ejemplos, útiles para restauración documental o data augmentation.

La combinación de estas arquitecturas permite desarrollar sistemas híbridos que aprovechen las fortalezas de cada paradigma para las diferentes subtareas del análisis documental.

2.3.2. Arquitecturas Especializadas para Análisis Documental

El análisis de documentos ha motivado el desarrollo de arquitecturas específicamente diseñadas para abordar sus desafíos particulares:

- **Redes Completamente Convolucionales (FCN)**: Extienden las CNN tradicionales para generar mapas de segmentación densos, fundamentales para la segmentación semántica de regiones documentales.
- **Arquitecturas U-Net**: Con su característica estructura de codificador-decodificador con conexiones residuales, permiten segmentaciones precisas preservando detalles de bordes y estructuras finas.
- **Mask R-CNN**: Integra detección de objetos y segmentación de instancias [8], facilitando la identificación individualizada de elementos como párrafos, tablas o figuras.

- **Modelos Multimodales**: Arquitecturas como LayoutLM y sus variantes [24] integran información textual, visual y posicional, capturando la naturaleza multidimensional de los documentos.
- **Modelos Preentrenados con Supervisión Visual-Lingüística**: Aprovechan grandes corpus de imágenes y texto para aprender representaciones transferibles a tareas documentales específicas.

Estas arquitecturas especializadas han permitido avances significativos en benchmarks como la detección de límites de página o la competición cBAD para detección de líneas base [25].

2.3.3. Avances en Segmentación Mediante Aprendizaje Profundo

La segmentación de imágenes mediante aprendizaje profundo ha experimentado una rápida evolución aplicable al análisis documental:

- **Segmentación Semántica**: Asigna una etiqueta de clase a cada píxel, permitiendo identificar regiones como texto, imágenes, tablas o fondo. Arquitecturas como PSPNet utilizan contexto piramidal para capturar información multiescala [41].
- **Segmentación de Instancias**: Diferencia entre múltiples instancias de la misma clase (por ejemplo, distinguir entre párrafos individuales), utilizando enfoques como YOLACT para segmentación en tiempo real [42].
- **Segmentación Panóptica**: Unifica segmentación semántica e instancia [9], proporcionando una comprensión holística de todos los elementos del documento.
- **Aprendizaje con Campos Receptivos Dilatados**: Técnicas como DeepLab emplean convoluciones atrous para capturar contexto más amplio sin reducir la resolución espacial [43].

Estos avances han mejorado significativamente la precisión en la delineación de regiones complejas, superando limitaciones de enfoques anteriores en cuanto a sensibilidad al ruido y capacidad para manejar variaciones de estilo y formato.

2.4 Análisis de la Estructura de Documentos

El análisis de la estructura de documentos (DLA, Document Layout Analysis) comprende el conjunto de técnicas destinadas a identificar y categorizar los componentes estructurales y su organización dentro de un documento [10]. Esta disciplina constituye un paso fundamental para la interpretación semántica posterior, especialmente en documentos históricos con layouts complejos y variables.

2.4.1. Definición del Problema

El análisis estructural de documentos busca descomponer la imagen del documento en regiones homogéneas (bloques de texto, tablas, figuras, fórmulas, etc.) y establecer relaciones jerárquicas o espaciales entre ellas. Esta tarea implica múltiples niveles de análisis:

- **Segmentación física**: División del documento en regiones visualmente coherentes.

- **Clasificación de regiones**: Asignación de categorías funcionales a cada región identificada.
- **Análisis de relaciones**: Determinación del orden de lectura, agrupaciones lógicas y jerarquías entre componentes.
- **Identificación de la estructura lógica**: Asignación de roles semánticos (título, autor, apartado, pie de figura) a las regiones.

La complejidad del análisis estructural aumenta significativamente en documentos históricos debido a factores como la ausencia de convenciones estandarizadas, deterioro físico, variabilidad caligráfica y presencia de elementos decorativos no textuales.

2.4.2. Desafíos en Documentos Históricos

Los documentos históricos presentan retos particulares para el análisis estructural [11]:

- **Variabilidad histórica**: Evolución de estilos, formatos y convenciones a través de diferentes períodos y regiones geográficas.
- **Degradación física**: Manchas, pliegues, roturas y transparencia del soporte que dificultan la separación primer plano-fondo.
- **Layouts no estándar**: Organización espacial compleja con márgenes irregulares, anotaciones marginales y elementos decorativos.
- **Escasez de anotaciones**: Limitada disponibilidad de datos etiquetados para entrenamiento de modelos supervisados.
- **Heterogeneidad**: Gran diversidad de tipologías documentales, desde manuscritos a documentos impresos tempranos con características híbridas.

Estos desafíos han motivado el desarrollo de enfoques robustos que combinen conocimiento específico del dominio con técnicas adaptativas capaces de generalizar a nuevas tipologías documentales.

2.4.3. Taxonomía del Análisis de Estructura

Las aproximaciones al análisis estructural de documentos pueden clasificarse según diferentes criterios:

- **Según el punto de partida del análisis**:
  - Métodos ascendentes (bottom-up): Comienzan con elementos atómicos (píxeles o componentes conectados) que se agrupan progresivamente en estructuras más complejas.
  - Métodos descendentes (top-down): Parten del documento completo y lo subdividen recursivamente en regiones más pequeñas.
  - Métodos híbridos: Combinan ambas estrategias para aprovechar sus respectivas fortalezas.
- **Según el conocimiento empleado**:

  - Métodos dirigidos por datos: Basan la segmentación en características visuales sin incorporar conocimiento específico del dominio.
  - Métodos basados en modelos: Utilizan conocimiento previo sobre la estructura típica de la clase de documentos analizada.
  - Métodos de aprendizaje: Infieren automáticamente patrones estructurales a partir de ejemplos.

- **Según el nivel de integración**:
  - Métodos secuenciales: Abordan la segmentación física, clasificación y análisis de relaciones como etapas independientes.
  - Métodos integrados: Realizan múltiples niveles de análisis simultáneamente, explotando interdependencias.

Esta taxonomía no establece categorías mutuamente excluyentes; los sistemas avanzados suelen incorporar elementos de múltiples paradigmas para abordar la complejidad inherente al análisis documental.

2.4.4. Aplicaciones y Evaluación

El análisis estructural constituye un componente esencial en numerosas aplicaciones relacionadas con documentos históricos y contemporáneos:

- **Sistemas de HTR**: La delimitación precisa de regiones textuales mejora significativamente la precisión del reconocimiento automático de texto manuscrito.
- **Indexación y recuperación**: La identificación de componentes estructurales facilita búsquedas específicas por tipo de contenido.
- **Análisis de tablas**: La detección y comprensión de estructuras tabulares permite la extracción de datos relacionales.
- **Preservación digital**: La captura estructural complementa la conservación de contenido, permitiendo reconstrucciones fieles al original.
- **Análisis comparativo**: Facilita estudios evolutivos de formatos y convenciones documentales a través de diferentes períodos históricos.

La comparación de sistemas de análisis estructural se realiza mediante competiciones internacionales organizadas por congresos como ICDAR (International Conference on Document Analysis and Recognition), utilizando métricas específicas que consideran tanto la precisión en la segmentación como la correcta interpretación de relaciones espaciales y jerárquicas. Los protocolos de evaluación han evolucionado desde métricas simplificadas basadas en píxeles hacia evaluaciones más sofisticadas que incorporan criterios estructurales y funcionales.

CAPÍTULO 3
Experimentación

3.1 Introducción

En este capítulo se describe el desarrollo experimental llevado a cabo para evaluar distintas arquitecturas de segmentación de instancias aplicadas a documentos manuscritos históricos. La secuencia de secciones responde a un enfoque lógico y progresivo que parte de la preparación técnica hasta llegar a la obtención e interpretación crítica de los resultados.

Comenzaremos exponiendo en detalle la infraestructura utilizada, tanto a nivel de hardware como de software, ya que condiciona en gran medida la viabilidad y el rendimiento de los entrenamientos. A continuación, se presentarán los modelos seleccionados —Detectron2 y YOLOv8— junto con una justificación de su elección, basada en su relevancia actual, su adecuación a la tarea y sus capacidades de segmentación.

El estudio se ha desarrollado sobre dos conjuntos de datos principales: el corpus OHG (Online Handwritten Girona), centrado en escrituras notariales manuscritas del archivo de Girona, y el corpus VORAU-253, que contiene documentos del monasterio de Vorau. Ambos datasets han sido preparados y anotados con un conjunto de clases común, lo que permite una evaluación comparativa coherente.

Una vez definido el contexto de trabajo, se especificarán las métricas de evaluación adoptadas, centradas en variantes del Average Precision (AP), seguidas de la descripción de los distintos experimentos diseñados: desde entrenamientos desde cero hasta estrategias de fine-tuning sobre pesos preentrenados, abarcando distintas configuraciones arquitectónicas.

Finalmente, se presentarán y analizarán los resultados obtenidos, tanto a nivel cuantitativo como cualitativo. El objetivo no será únicamente comparar métricas, sino también reflexionar sobre las fortalezas y limitaciones prácticas de cada enfoque en el contexto específico del corpus OHG.

La estructura del capítulo es la siguiente:

- La Sección 3.2 describe el entorno experimental utilizado, incluyendo hardware, software y librerías complementarias.
- La Sección 3.3 explica las herramientas y modelos empleados en los experimentos.
- La Sección 3.4 presenta los datasets utilizados y su proceso de anotación.
- La Sección 3.5 detalla las métricas de evaluación utilizadas.
- La Sección 3.6 describe los experimentos realizados.

- La Sección 3.7 presenta los resultados obtenidos y su análisis detallado.

3.2 Entorno Experimental

Esta sección tiene como objetivo presentar en detalle los entornos de ejecución utilizados para llevar a cabo el desarrollo experimental del trabajo, así como las herramientas, librerías y configuraciones técnicas implicadas. Para ello, se han utilizado dos entornos claramente diferenciados pero complementarios: por un lado, un entorno local configurado en un ordenador personal, que permitió un control más directo del entorno de ejecución y la instalación personalizada de herramientas específicas como Detectron2; y, por otro lado, un entorno en la nube mediante la plataforma Google Colab, que ofreció acceso a recursos computacionales más potentes y configurables, como distintas GPUs proporcionadas por Google.

Ambos entornos han sido empleados en distintas fases del trabajo, dependiendo de las necesidades de procesamiento, disponibilidad de recursos o facilidad de integración con los distintos modelos y bibliotecas. La descripción técnica de cada uno de ellos se desarrollará en las secciones siguientes, con el fin de proporcionar una visión clara de los contextos en los que se han entrenado y evaluado los modelos.

Finalmente, una vez presentado el análisis de los resultados experimentales y completado el proceso de evaluación, se incluye en la conclusión un breve comentario comparativo sobre las ventajas y limitaciones que ha presentado cada uno de los entornos utilizados.

3.2.1. Configuración de Hardware y Software

Configuración de Hardware Local El entorno experimental para Detectron2 se configuró con los siguientes componentes:

- **Sistema operativo**: Linux (Ubuntu)
- **Versión de Python**: 3.12.3
- **Versión de PyTorch**: 2.6.0 con soporte CUDA 12.4
- **Versión de torchvision**: 0.21.0
- **Compilador utilizado**: GCC 13.3
- **Versión de CUDA instalada**: 12.0
- **Driver de NVIDIA**: versión 572.16
- **GPU disponible**: NVIDIA GeForce GTX 1660 (arquitectura 7.5)
- **Aceleración por hardware**: habilitada mediante CUDA y compatible con las versiones de arquitectura requeridas por PyTorch, Detectron2 y Ultralytics.

Configuración de Hardware Google Para YOLOv8, se utilizó un entorno con las siguientes especificaciones:

- **Python**: versión 3.11.12
- **PyTorch**: versión 2.6.0 con CUDA 12.4

- **Hardware de Aceleración**: Tesla T4 (15095MiB)
- **Sistema de Archivos**: Integración con Google Drive para almacenamiento de modelos y resultados

3.2.2. Librerías Complementarias

Otras librerías complementarias relevantes para la ejecución de los modelos incluyen:

- `numpy` versión 2.2.3.
- `Pillow` versión 11.1.0.
- `opencv-python (cv2)` versión 4.11.0.
- `fvcore` versión 0.1.5.post20221221.
- `iopath` versión 0.1.9.

Cabe destacar que el entorno se configuró manualmente en un entorno virtual de Python, lo cual permitió mantener las dependencias del proyecto aisladas del sistema base.

Esta configuración asegura la compatibilidad entre librerías, así como el aprovechamiento óptimo del hardware disponible durante las fases de entrenamiento, inferencia y validación del modelo. La utilización de aceleración GPU resulta particularmente relevante para el procesamiento de imágenes de documentos históricos de alta resolución, reduciendo significativamente los tiempos de entrenamiento y permitiendo la experimentación con arquitecturas más complejas y conjuntos de datos más extensos.

3.3 Herramientas y Modelos

En esta sección se describen las herramientas y modelos utilizados a lo largo del trabajo, con especial atención a los dos frameworks principales: Detectron2 y YOLOv8. La elección de ambos responde a criterios de disponibilidad, versatilidad y relevancia en tareas de detección y segmentación de instancias.

En lugar de realizar una revisión general de los frameworks, se detallan las arquitecturas específicas utilizadas en este proyecto dentro de cada uno de ellos. Para Detectron2 se ha empleado la arquitectura Mask R-CNN con un backbone ResNet-50 y Feature Pyramid Network (R50-FPN), mientras que en el caso de YOLOv8 se hautilizado la variante yolov8l por su semejanza directa con la arquitectura de Detectron. A lo largo de la sección se analizan las características técnicas y estructurales de estas arquitecturas, así como las decisiones de configuración adoptadas para su entrenamiento y evaluación en el contexto del análisis de documentos manuscritos.

Esta revisión técnica permitirá contextualizar adecuadamente los resultados obtenidos en fases posteriores del trabajo.

3.3.1. Detectron2

Introducción

Detectron2 constituye un framework avanzado de código abierto para visión por computador, desarrollado por Facebook AI Research (FAIR) [13]. Este sistema propor-

ciona implementaciones de alta calidad de algoritmos estado del arte para detección de objetos y segmentación de imágenes, representando una evolución significativa respecto a su predecesor Detectron. Su implementación en PyTorch le confiere ventajas sustanciales en términos de flexibilidad, rendimiento y facilidad de uso tanto en entornos de investigación como de aplicación práctica.

La génesis de Detectron2 se enmarca en el contexto del acelerado desarrollo de métodos de visión artificial basados en redes neuronales profundas. Tras el éxito de Detectron, lanzado a principios de 2018, el equipo de FAIR identificó la necesidad de desarrollar una plataforma más modular, extensible y eficiente que pudiera adaptarse a la creciente sofisticación de los modelos y a los requerimientos de aplicaciones cada vez más exigentes. El resultado fue una reescritura completa del sistema que adoptó PyTorch como framework base, mejorando sustancialmente la experiencia de desarrollo y experimentación.

Entre las principales aplicaciones de Detectron2 destaca la detección de objetos, implementando arquitecturas de referencia como Faster R-CNN y RetinaNet que permiten identificar y localizar múltiples elementos en imágenes con alta precisión. Estas capacidades resultan fundamentales en campos tan diversos como la conducción autónoma, robótica, análisis de imágenes médicas y, particularmente relevante para este trabajo, el análisis y comprensión de documentos. La flexibilidad del framework facilita su adaptación a dominios específicos, permitiendo la transferencia de conocimiento desde modelos entrenados en conjuntos de datos generales hacia tareas especializadas.

Aunque Detectron2 también incluye funcionalidades avanzadas de segmentación de instancias —implementadas principalmente a través de la arquitectura Mask R-CNN [8]—, en este trabajo dicha funcionalidad ha sido desactivada deliberadamente. Se ha empleado únicamente la capacidad de detección por medio de bounding boxes, al considerar que este tipo de anotación es suficiente para capturar la estructura visual de los documentos analizados. La segmentación a nivel de píxel puede ser especialmente útil en casos de oclusión o solapamiento, pero no se ha considerado necesaria en esta fase del proyecto.

En el contexto específico del análisis documental, Detectron2 ha demostrado su valía para tareas de segmentación y clasificación de componentes estructurales [19, 20, 21]. Su capacidad para manejar imágenes de alta resolución, adaptarse a diferentes dominios y definir clases personalizadas lo convierte en una herramienta especialmente adecuada para trabajar con documentos históricos, donde la variabilidad visual y estructural plantea desafíos significativos. Diversos estudios han aplicado este framework para la segmentación de manuscritos, documentos impresos históricos y materiales de archivo, obteniendo resultados prometedores en la identificación automática de elementos como texto principal, anotaciones marginales, títulos, ilustraciones y tablas [22, 23].

La relevancia actual de Detectron2 en el campo de la visión artificial puede atribuirse a varios factores clave. En primer lugar, su diseño modular facilita la experimentación con nuevos componentes sin necesidad de reescritura completa, acelerando el ciclo de adaptación. Además, sus implementaciones optimizadas permiten entrenamientos eficientes incluso en entornos con recursos computacionales limitados. Por último, su extenso ecosistema de modelos preentrenados, herramientas de visualización y convertidores de formato facilita su adopción en proyectos de diversa envergadura [9].

Para este proyecto específico, Detectron2 representa una elección estratégica debido a tres factores principales: su capacidad demostrada en tareas de segmentación, su robustez frente a variaciones en la calidad de imagen (frecuentes en documentos históricos), y su flexibilidad para adaptarse a las necesidades específicas de segmentación estructural en páginas manuscritas. Aunque en esta implementación se haya optado por desactivar

la predicción de máscaras, la arquitectura subyacente permite ampliar el enfoque en futuras iteraciones, incorporando segmentación a nivel de píxel si se considera necesario.

Arquitectura General

Detectron2 está diseñado siguiendo una arquitectura modular y altamente escalable, lo que permite separar claramente las responsabilidades funcionales de cada componente y facilita su análisis y personalización. Esta modularidad también habilita la integración de nuevas tareas, como segmentación o detección de keypoints, sin necesidad de rediseñar la estructura base.

En términos generales, la arquitectura de Detectron2 se compone de cinco bloques fundamentales:

1. **Backbone (espina dorsal)**
2. **Feature Pyramid Network (FPN)**
3. **Region Proposal Network (RPN)**
4. **ROI Heads (Región de Interés)**
5. **Módulos adicionales** (según la tarea: Mask Head, Keypoint Head, etc.)

La Figura 3.1 proporciona una visión general de esta estructura, desde la imagen de entrada hasta la predicción final.

Backbone El Backbone es una red convolucional profunda cuya función es extraer representaciones de alto nivel de las imágenes de entrada. Detectron2 utiliza como backbone modelos como ResNet-50, ResNet-101, o MobileNetV2, según los requisitos de precisión y eficiencia.

Las salidas del backbone son tensores tetradimensionales (batch, height, width, channels) que representan las características detectadas a diferentes niveles. Sin embargo, por sí solos, estos mapas carecen de sensibilidad multiescala, lo que es crítico en tareas de detección de objetos, donde los objetos pueden variar enormemente en tamaño.

Feature Pyramid Network (FPN) Para abordar este problema, Detectron2 incorpora una Feature Pyramid Network (FPN) encima del backbone. La FPN toma las salidas intermedias del backbone y construye una jerarquía de mapas de características a múltiples escalas.

- Cada nivel (P2, P3, P4, P5, P6) corresponde a una resolución diferente, con P2 teniendo la mayor resolución espacial y P6 la menor.
- Cada uno de estos niveles mantiene 256 canales y se construye mediante una combinación de convoluciones ascendentes (top-down) y laterales (lateral connections).

Esto permite conservar tanto la información semántica de alto nivel como la resolución espacial detallada, crucial para detectar objetos pequeños o con formas complejas.

Region Proposal Network (RPN) La RPN es una subred que opera sobre cada nivel de la pirámide de características generada por la FPN. Su objetivo es generar un conjunto de propuestas de regiones candidatas (bounding boxes) que probablemente contengan objetos.

El proceso incluye:

- Colocación de anclas (anchors) predefinidas en cada posición del mapa de características.
- Cálculo de dos tensores por cada ancla:
  - objectness logits: probabilidad de que el ancla contenga un objeto.
  - anchor deltas: desplazamientos que ajustan las coordenadas del ancla a una propuesta más precisa.

Después de este proceso, se realiza un filtrado mediante Non-Maximum Suppression (NMS) para eliminar regiones redundantes y conservar solo aquellas con mayor puntuación de confianza.

ROI Heads Las regiones propuestas por la RPN se refinan en el bloque de ROI Heads, que tiene múltiples funciones:

1. **Asignación de nivel**: Cada región es asignada al nivel de pirámide correspondiente en función de su área. Por ejemplo, regiones pequeñas van a P2, medianas a P3, etc.
2. **ROIAlign**: Una operación de muestreo espacial precisa que extrae un tensor de características fijo (normalmente 7x7x256) de cada región, independientemente de su tamaño original.
3. **Clasificación y regresión**: El tensor extraído pasa por una red completamente conectada que:
   - Clasifica la región (por ejemplo, perro, bicicleta, etc.)
   - Ajusta aún más sus coordenadas (bounding box regression)

En el caso de tareas adicionales como segmentación o keypoints, se activan cabezas especializadas (Mask Head, Keypoint Head) que trabajan en paralelo con la clasificación y regresión.

Otros módulos y extensiones Gracias a su diseño, Detectron2 permite añadir funcionalidades avanzadas como:

- **Panoptic Segmentation**: combina segmentación semántica y de instancias.
- **DensePose**: mapea coordenadas de la superficie humana en 3D.
- **Custom Heads**: para tareas específicas, como predicción de bordes o relaciones entre objetos.

Estas extensiones comparten la base estructural del sistema y se integran fácilmente dentro del marco modular.

Funcionamiento Interno Detallado

En esta sección se detalla el flujo interno y la estructura del framework Detectron2.

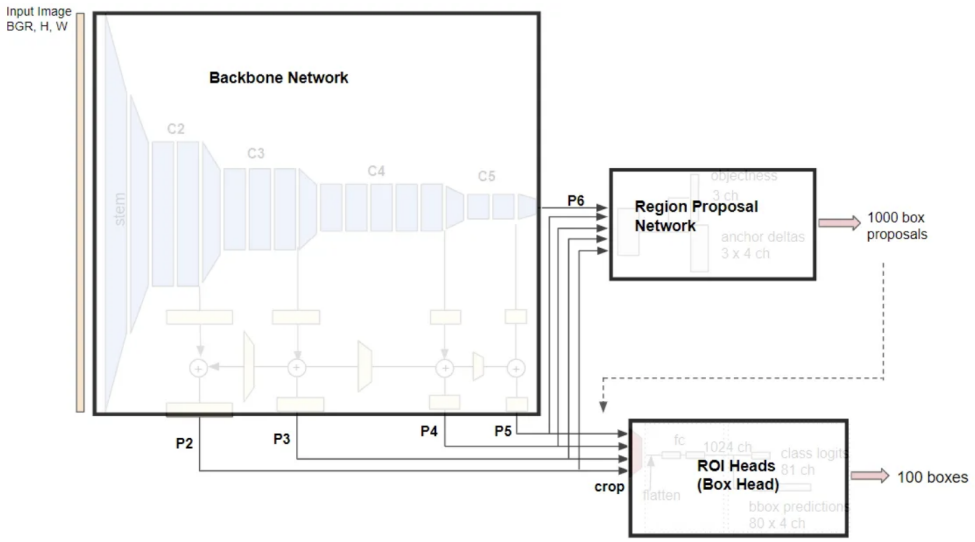

Figura 3.1: Esquema básico del flujo de procesamiento en Detectron2 [14].

El flujo de datos en Detectron2 puede verse como una cadena de transformación progresiva desde datos brutos (imágenes y anotaciones) hasta predicciones estructuradas [15, 16]. Cada paso intermedio está diseñado para encapsular eficientemente la información necesaria, maximizando la reutilización de estructuras y evitando cómputo redundante. La Figura 3.2 muestra un esquema detallado de este proceso, incluyendo las clases y módulos específicos de implementación.

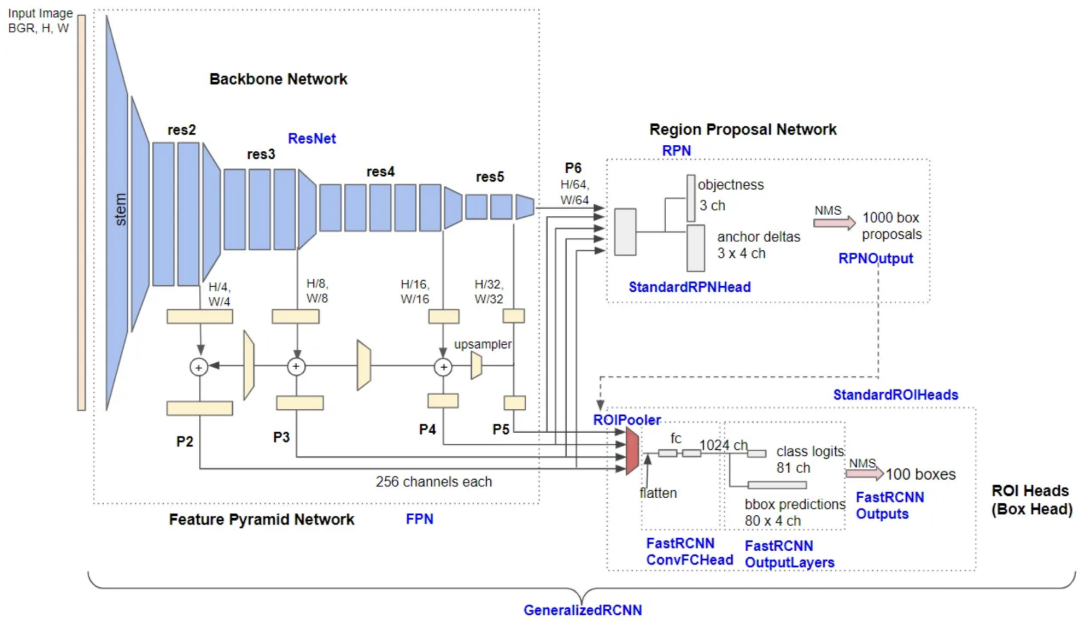

Figura 3.2: Diagrama detallado de la arquitectura interna de Detectron2 [14].

Carga de datos y anotaciones Los datos se cargan mediante un sistema centralizado de catálogos (DatasetCatalog y MetadataCatalog), que permite registrar datasets personalizados en formato COCO, LVIS, Pascal VOC, etc.

El data loader está compuesto por:

- **DatasetFromList**: transforma la lista de ejemplos en un dataset PyTorch.
- **MapDataset**: aplica transformaciones a cada muestra.
- **DatasetMapper**: realiza transformaciones como redimensionado, flip horizontal, normalización, y finalmente crea una instancia del objeto Instances, que encapsula:
  - Cajas delimitadoras (bounding boxes)
  - Clases
  - Máscaras o keypoints (si aplica)

Este diseño permite que el modelo reciba un input homogéneo independientemente de las transformaciones realizadas.

Forward pass: extracción de características Las imágenes transformadas pasan al Backbone, donde se extraen mapas de características. Estos son refinados por la FPN, generando 5 niveles (P2 a P6) que capturan información multiescala.

Cada nivel es un tensor de dimensiones (batch_size, 256, H_i, W_i), y todos juntos forman la representación espacial-semántica de la imagen.

Generación de propuestas (RPN) Los niveles P2 a P6 se procesan por separado en la RPN:

- En cada punto de la parrilla, se generan múltiples anclas con diferentes escalas y proporciones.
- Cada ancla produce:
  - Una predicción binaria (object vs background)
  - Un desplazamiento para ajustar la caja

Miles de anclas se generan por imagen, pero solo unas pocas (por ejemplo, 1000) se conservan tras aplicar top-K filtering y NMS.

Refinamiento mediante ROI Heads Las propuestas seleccionadas se asignan a niveles FPN mediante la fórmula:

```
level = floor(4 + log2(sqrt(area) / 224))
```

Esto garantiza que cada propuesta use el nivel más informativo según su tamaño [17]. La Figura 3.3 detalla este proceso de refinamiento, donde se aplican:

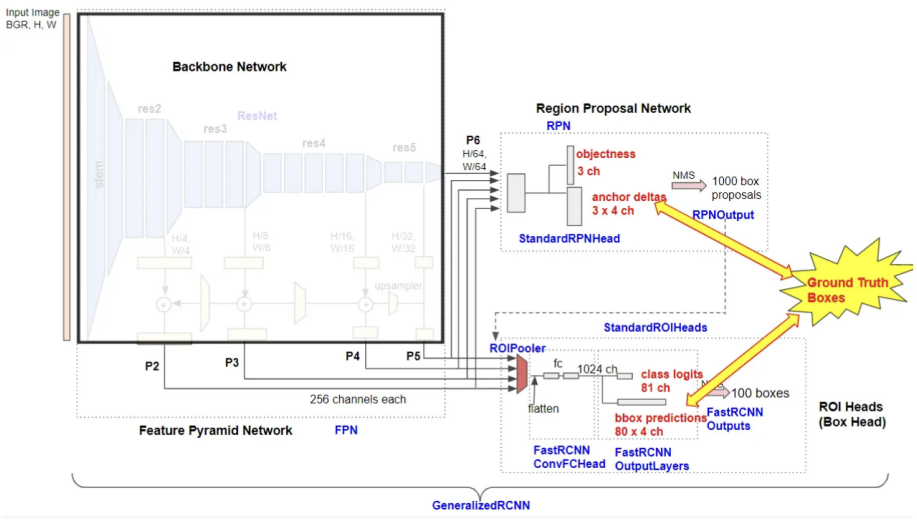

Figura 3.3: Detalle de la sección ROI Heads del pipeline de Detectron2 [14].

- **ROIAlign**: interpolación bilineal sobre el mapa de características asignado.
- **Box Head**: red completamente conectada que:
  - Clasifica la propuesta
  - Refina aún más las coordenadas
- **Mask Head / Keypoint Head** (si aplica): generan máscaras binarias o coordenadas clave.

Durante el entrenamiento, las predicciones se comparan contra el ground truth usando funciones de pérdida como Cross Entropy y Smooth L1 Loss.

Salida y evaluación Al final del forward pass, se obtienen:

- Cajas ajustadas
- Etiquetas de clase
- Puntuaciones de confianza
- Máscaras (si aplica)
- Keypoints (si aplica)

Estas predicciones se visualizan o evalúan mediante métricas como:

- mAP (mean Average Precision)
- IoU por clase
- AP50 / AP75 (precision con diferentes umbrales)

La evaluación de modelos en Detectron2 sigue los protocolos estándar establecidos por los principales benchmarks de visión por computador [18]. La Figura 3.4 muestra una visión simplificada del flujo completo sin las etiquetas de clases específicas, permitiendo apreciar mejor el proceso general.

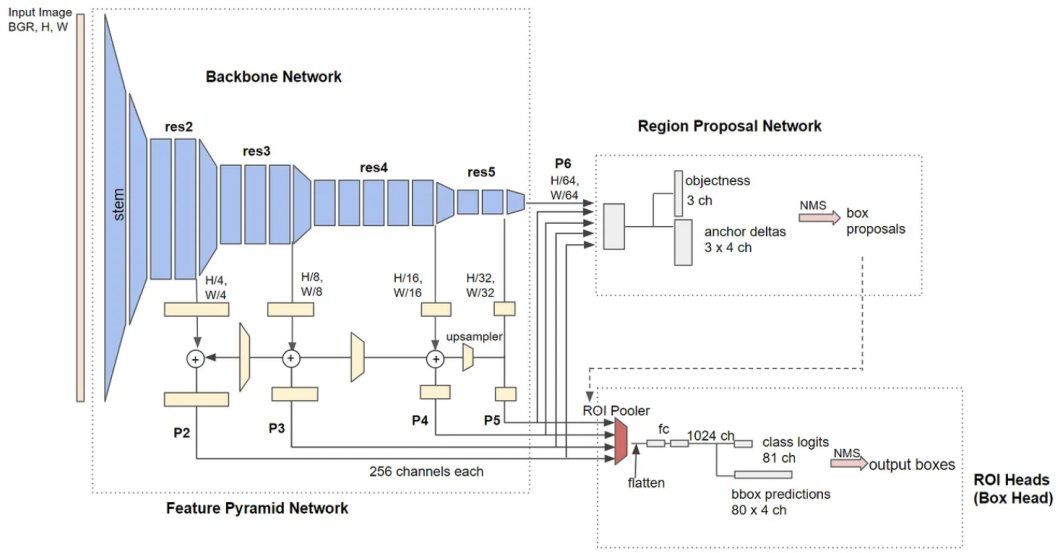

Figura 3.4: Visión global del pipeline de procesamiento de Detectron2 [14, 15].

3.3.2. YOLOv8

Introducción

YOLOv8 representa la última evolución en la familia de detectores YOLO (You Only Look Once), desarrollado por Ultralytics como sucesor de YOLOv5 [26]. Este framework de código abierto constituye un avance significativo en el campo de la visión por computador, optimizando tanto la precisión como la velocidad en tareas de detección, segmentación y clasificación de objetos [31]. A diferencia de sus predecesores, YOLOv8 implementa una arquitectura modernizada que aborda limitaciones presentes en versiones anteriores, particularmente en la detección de objetos pequeños y el manejo de oclusiones parciales.

La evolución de la arquitectura YOLO desde su introducción por Redmon et al. en 2016 [27] ha seguido una trayectoria de mejoras incrementales. Mientras que YOLOv3 introdujo detección en múltiples escalas mediante Feature Pyramid Networks [28], YOLOv4 y YOLOv5 incorporaron técnicas de aumento de datos y optimizaciones de arquitectura [29]. YOLOv7 experimentó con transformadores de atención y capas de cuello más complejas [30]. YOLOv8, por su parte, representa una reescritura sustancial con cambios fundamentales en el diseño de anclas, la estructura de backbone y los mecanismos de detección final [46].

En contraste con arquitecturas de dos etapas como Faster R-CNN [33], que primero proponen regiones y luego las clasifican, o detectores de una etapa como SSD (Single Shot Detector) [34], YOLOv8 mantiene el enfoque unificado característico de la familia YOLO, procesando la imagen completa en una sola pasada. Sin embargo, introduce un nuevo sistema de predicción denominado "distribution-guided", que reemplaza el anterior sistema basado en anclajes (anchor-based) con un enfoque más flexible y eficiente computacionalmente.

Entre las aplicaciones más relevantes de YOLOv8 se encuentra la detección en tiempo real para sistemas de vigilancia, vehículos autónomos, robótica y, particularmente pertinente para este trabajo, el análisis de documentos. Su capacidad para operar eficientemente en dispositivos con recursos limitados, manteniendo una precisión competitiva

respecto a modelos más pesados, lo posiciona como una herramienta especialmente valiosa para casos de uso donde el equilibrio entre precisión y velocidad resulta crítico.

La relevancia de YOLOv8 para el análisis de documentos históricos se fundamenta en su capacidad para detectar y segmentar elementos estructurales con distintas escalas y características visuales, desde grandes bloques de texto hasta pequeñas anotaciones marginales, manteniendo una inferencia suficientemente rápida para procesar grandes volúmenes documentales [35] [36].

En el desarrollo de este trabajo se explorarán distintas variantes del modelo YOLOv8, aunque el análisis principal se centrará en la versión YOLOv8-L.

Arquitectura General

La arquitectura de YOLOv8 sigue un diseño modular organizado en tres componentes principales: backbone, neck y head, cada uno con funciones específicas y optimizaciones respecto a modelos anteriores [31].

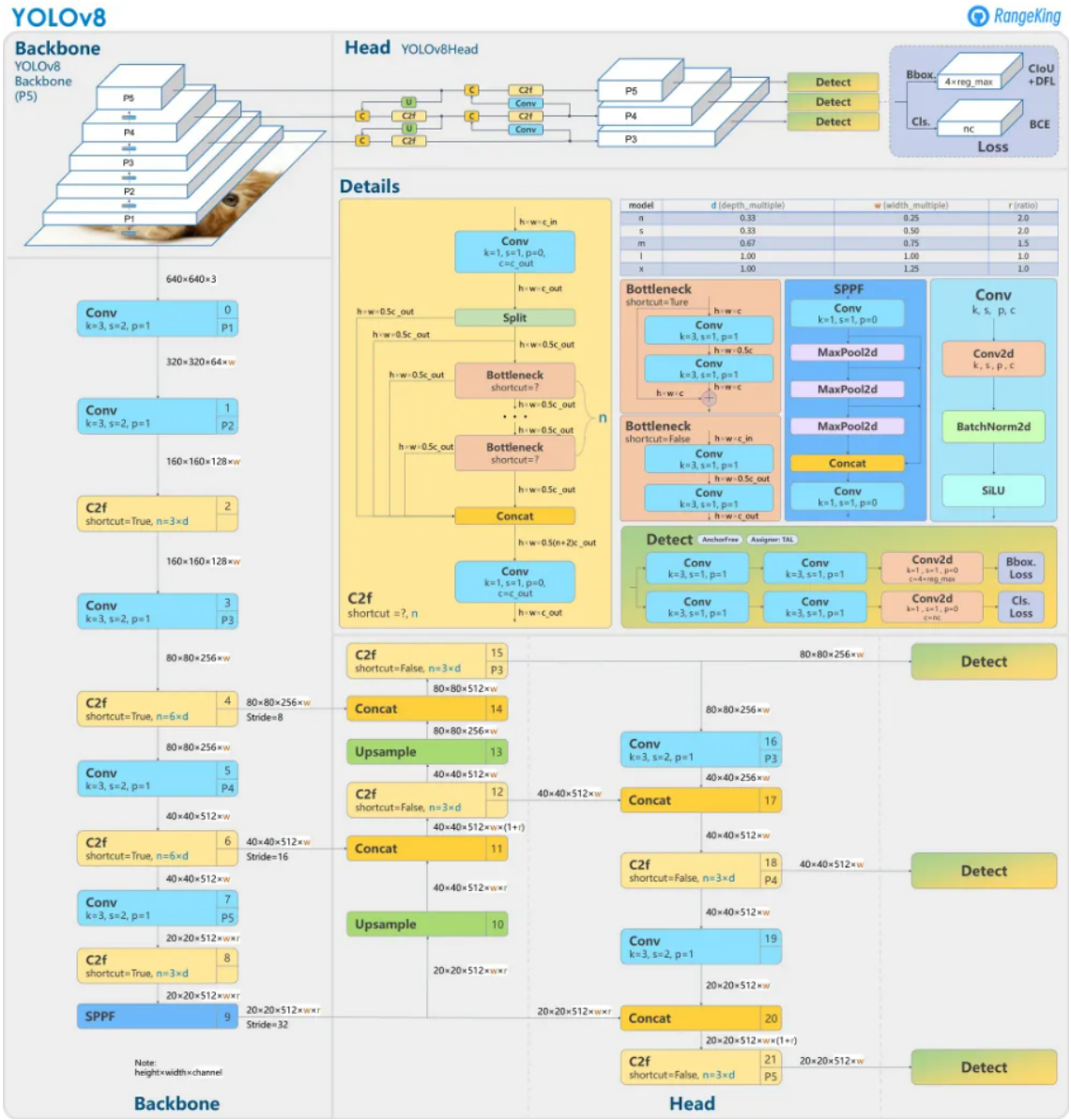

Figura 3.5: Arquitectura detallada de YOLOv8, visualización creada por GitHub user RangeKing [32].

Backbone El backbone de YOLOv8 emplea una arquitectura CSPNet-X (Cross Stage Partial Network) [46], diseñada específicamente para la extracción eficiente de características visuales. Este componente es el responsable de procesar la imagen de entrada y generar representaciones jerárquicas con diferentes niveles de abstracción. La estructura del backbone incluye:

- **C2f Blocks (Cross-Stage-Partial 2f)**: Los bloques C2f representan un componente clave en la arquitectura del *backbone* de YOLOv8. Están compuestos por múltiples capas convolucionales dispuestas siguiendo un esquema de conexiones parciales entre etapas, basado en el diseño de CSPNet-X. Este enfoque divide el tensor de características en dos subconjuntos: una parte atraviesa un camino denso de convoluciones, mientras que la otra sigue una ruta directa y menos costosa. Esta estructura optimiza el uso de recursos computacionales, al reducir el número de operaciones sin sacrificar la capacidad de representación del modelo. Además, las conexiones cruzadas permiten una integración progresiva de la información, mejorando la robustez en la extracción de características.

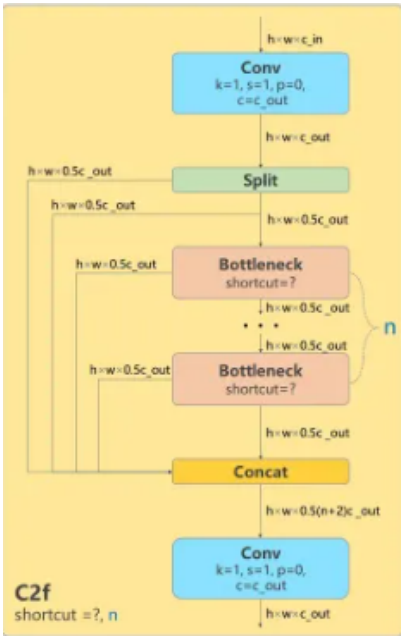

Figura 3.6: Detalle de la estructura de un bloque C2f en YOLOv8, recorte de la arquitectura general mostrada en la Figura 3.5 [32].

- **Convoluciones Especializadas**: YOLOv8 incorpora diversos tipos de convoluciones optimizadas:

  - GhostConv: Generan más características con menor coste computacional mediante la aplicación de convoluciones ligeras a las características ya obtenidas.
  - VoVNet: Diseñadas para aumentar el campo receptivo de manera eficiente, permitiendo que cada neurona perciba una porción mayor de la imagen original.
  - BottleNeck: Reducen temporalmente la dimensionalidad de los canales para disminuir el coste computacional en operaciones intermedias.

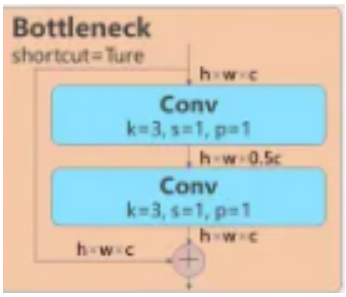

Figura 3.7: Bottleneck con shortcut=True en YOLOv8, recorte de la arquitectura general mostrada en la Figura 3.5.

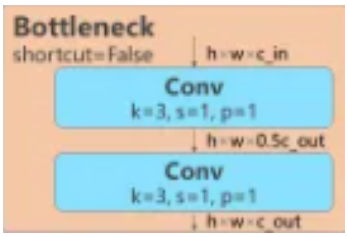

Figura 3.8: Bottleneck con shortcut=False en YOLOv8, recorte de la arquitectura general mostrada en la Figura 3.5.

- **Sistema de Activación y Normalización**: El backbone utiliza:
  - SiLU (Sigmoid Linear Unit): Función de activación que combina las ventajas de operaciones sigmoidales y lineales, facilitando el flujo de gradientes durante el entrenamiento.
  - Normalización por grupos: Estabiliza el aprendizaje dividiendo los canales en grupos y normalizando cada grupo independientemente, lo que resulta más robusto que la normalización por lotes en redes profundas.

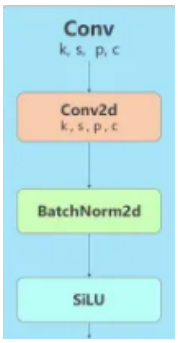

Figura 3.9: Detalle de la estructura de un bloque convolucional en YOLOv8, recorte de la arquitectura general mostrada en la Figura 3.5.

La estructura piramidal del backbone produce mapas de características a múltiples escalas, capturando tanto información detallada (texturas, bordes) en niveles poco profundos como patrones semánticos complejos en niveles más profundos. Estas representaciones multi-escala son fundamentales para la detección efectiva de elementos documentales de diferentes tamaños, desde pequeñas anotaciones hasta bloques de texto completos.

Neck El componente neck de YOLOv8 actúa como puente entre el backbone y la cabeza de detección, enriqueciendo los mapas de características mediante flujos de información bidireccionales. Su diseño está optimizado para fusionar información semántica de diferentes niveles de resolución, permitiendo que la información contextual de alto nivel enriquezca la información espacial detallada y viceversa. Los elementos clave del neck incluyen:

- **Feature Pyramid Network (FPN) Bidireccional**: Implementa un flujo de información en dos direcciones:
  - Top-down (descendente): Propaga información semántica desde capas profundas hacia capas superficiales mediante conexiones de upsampling (escalado hacia arriba).
  - Bottom-up (ascendente): Refuerza las capas profundas con detalles espaciales de capas superficiales mediante conexiones de downsampling (escalado hacia abajo).
- **Spatial Pyramid Pooling Fast (SPPF)**: Este módulo aplica operaciones de pooling a diferentes escalas sobre un mismo mapa de características, generando representaciones con campos receptivos variables. A diferencia del SPP tradicional, el SPPF reduce la redundancia computacional aplicando primero un único pooling máximo y luego realizando operaciones adicionales sobre el resultado, en lugar de aplicar múltiples operaciones de pooling sobre la entrada original.

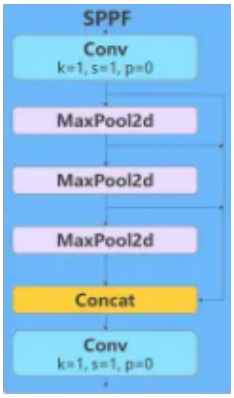

Figura 3.10: Estructura del módulo SPPF (Spatial Pyramid Pooling Fast) en YOLOv8, recorte de la arquitectura general mostrada en la Figura 3.5.

- **Mecanismos de Atención por Canal**: Estos mecanismos calculan pesos para cada canal de características basándose en su importancia para la tarea de detección, permitiendo al modelo enfocarse en las características más relevantes para cada tipo de objeto. Funcionan mediante:
  - Compresión espacial mediante pooling global.
  - Transformación no lineal para calcular pesos por canal.
  - Re-escalado adaptativo de cada canal según su importancia.

El neck genera tres niveles de mapas de características (P3, P4, P5) con resoluciones progresivamente menores pero mayor riqueza semántica. Cada nivel mantiene un número constante de canales (generalmente 256), pero difiere en sus dimensiones espaciales. Esta estructura piramidal permite la detección eficiente de objetos a múltiples escalas: los mapas de baja resolución (P5) son óptimos para detectar elementos grandes, mientras que los de alta resolución (P3) capturan detalles finos como texto pequeño o marcas de puntuación.

Head La cabeza (head) de YOLOv8 es el componente responsable de transformar los mapas de características enriquecidos en predicciones concretas de objetos. Utiliza un enfoque anchor-free (sin anclajes) que simplifica el proceso de detección:

- **Sistema de Detección sin Anclajes**: A diferencia de los detectores tradicionales que predicen desplazamientos relativos a cajas de anclaje predefinidas, YOLOv8 implementa un enfoque directo:
  - Divide la imagen en una cuadrícula donde cada celda es responsable de detectar objetos cuyo centro cae dentro de ella.
  - Cada celda predice directamente: coordenadas normalizadas del centro (x, y), dimensiones (ancho, alto) y puntuaciones de confianza para cada clase posible.
  - Esta aproximación elimina la necesidad de diseñar anclajes óptimos para cada dominio específico, facilitando la adaptación a nuevos tipos de documentos.
- **Arquitectura Multi-rama**: La cabeza se divide en ramas independientes especializadas:
  - Rama de clasificación: Determina a qué clase pertenece cada objeto detectado.
  - Rama de regresión: Predice las coordenadas precisas de las cajas delimitadoras.
  - Rama de segmentación (en YOLOv8-seg): Genera máscaras a nivel de píxel para cada instancia.

  Esta separación permite que cada tarea se optimice de manera independiente, mejorando la precisión general del modelo.
- **Sistema de Asignación Distribution-Guided**: Durante el entrenamiento, determina qué celdas de la cuadrícula son responsables de detectar cada objeto:
  - Considera tanto la posición del centro como las dimensiones del objeto.
  - Asigna múltiples celdas a cada objeto basándose en una distribución gaussiana centrada en el centro del objeto.
  - Esta estrategia produce asignaciones más naturales que los métodos basados en IoU, especialmente para objetos con formas irregulares comunes en documentos históricos.
- **Cabezas Alineadas con la Tarea**: Para modelos de segmentación (YOLOv8-seg), utiliza:
  - Convoluciones transpuestas para escalar las características a la resolución original.
  - Prototipos de máscara compartidos que se combinan con coeficientes específicos para cada instancia.
  - Operaciones matriciales optimizadas que generan máscaras precisas con bajo coste computacional.

Esta arquitectura de cabeza permite predicciones precisas y eficientes, incluso en documentos con layouts complejos donde los elementos pueden variar considerablemente en tamaño, forma y distribución espacial.

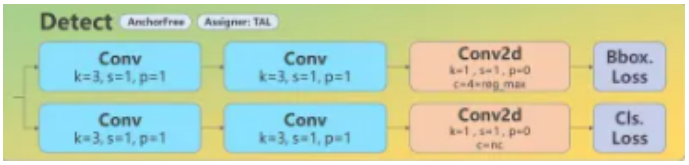

Figura 3.11: Estructura del módulo de detección (head) en YOLOv8, recorte de la arquitectura general mostrada en la Figura 3.5.

Tecnologías Complementarias YOLOv8 incorpora diversas tecnologías complementarias que optimizan su funcionamiento para tareas de análisis documental:

- **Entrenamiento Multi-escala**: El modelo se entrena con imágenes redimensionadas aleatoriamente dentro de un rango predefinido en cada iteración:
  - En cada lote, las imágenes pueden variar desde 0.5× hasta 1.5× su tamaño original.
  - Esta técnica simula la variabilidad natural en la escala de digitalización de documentos históricos.
  - Mejora la robustez frente a documentos digitalizados con diferentes resoluciones y tamaños.
- **Técnicas Avanzadas de Aumento de Datos**: Implementa estrategias sofisticadas para diversificar artificialmente los datos de entrenamiento:
  - Mosaic: Combina cuatro imágenes diferentes en una sola, creando contextos variados con múltiples objetos.
  - MixUp: Fusiona dos imágenes con una ponderación aleatoria, ayudando al modelo a aprender representaciones más robustas.
  - Random Affine: Aplica transformaciones como rotación, escalado y cizallamiento que simulan variaciones en la orientación y perspectiva de los documentos.
  - Simulación de deterioro: Añade artificialmente manchas, ruido y variaciones de contraste similares a las encontradas en documentos antiguos.
- **Mecanismos de Regularización Adaptativa**: Ajusta dinámicamente los parámetros de regularización durante el entrenamiento:
  - Weight decay adaptativo: Modifica la penalización de los pesos según el progreso del entrenamiento.
  - Dropout espacial: Desactiva aleatoriamente regiones enteras del mapa de características, forzando al modelo a aprender representaciones redundantes.
  - Early stopping controlado: Monitoriza múltiples métricas para determinar el punto óptimo de detención del entrenamiento.

Estas tecnologías complementarias son fundamentales para la adaptación efectiva de YOLOv8 al dominio específico de documentos históricos, donde la escasez de datos de entrenamiento y la variabilidad inherente de los materiales representan desafíos significativos para los sistemas de visión por computador.

Funcionamiento Interno Detallado

El flujo de procesamiento en YOLOv8 comienza con una imagen de entrada y culmina con predicciones refinadas de objetos detectados, siguiendo una secuencia bien definida de operaciones que optimizan tanto la precisión como la eficiencia.

Preprocesamiento y entrada de datos La primera fase involucra la preparación de las imágenes para su procesamiento:

- La imagen se redimensiona a una resolución objetivo (típicamente 640×640 píxeles), manteniendo la relación de aspecto mediante padding.

- Se normaliza cada canal con valores entre 0 y 1, aplicando opcionalmente transformaciones de color como parte de la estrategia de aumento.
- Para documentos históricos, este preprocesamiento es particularmente relevante dado que las imágenes suelen provenir de escaneos con distintas resoluciones y características.

El batch de imágenes resultante se organiza como un tensor 4D (batch_size, canales, altura, anchura) antes de ser procesado por el backbone.

Extracción de características jerárquicas El backbone procesa la imagen de entrada secuencialmente:

- Las primeras capas capturan características de bajo nivel como bordes y texturas, cruciales para distinguir elementos textuales en documentos manuscritos.
- Las capas intermedias agregan estas características en patrones más complejos que representan estructuras como párrafos, tablas o figuras.
- Las capas profundas desarrollan representaciones semánticas de alto nivel con información contextual sobre el layout del documento.

A diferencia de YOLOv5, YOLOv8 extrae características de tres profundidades diferentes (P3, P4, P5) que corresponden a resoluciones espaciales progresivamente reducidas pero con mayor riqueza semántica.

Generación de predicciones multiescala El componente neck fusiona características de diferentes niveles del backbone:

- Mediante conexiones top-down, la información semántica de capas profundas se propaga hacia capas con mayor resolución.
- Conexiones bottom-up complementarias permiten que detalles espaciales precisos informen a las capas semánticamente más ricas.
- Esta arquitectura bidireccional genera tres conjuntos de mapas de características enriquecidos que preservan tanto detalles finos como información contextual.

Para cada nivel de características (P3, P4, P5), el head genera predicciones independientes, lo que permite detectar objetos grandes (como páginas completas) en niveles con menor resolución espacial y elementos pequeños (como anotaciones marginales) en niveles de mayor resolución.

Decodificación y refinamiento La etapa final transforma las predicciones en formato utilizable:

- Cada celda en los mapas de características predice directamente coordenadas normalizadas (centro_x, centro_y, ancho, alto) y puntuaciones de confianza para cada clase.
- Para modelos de segmentación, se añaden coeficientes de máscara que, combinados con prototipos de máscara generados por una rama paralela, producen máscaras binarias precisas.

- Las predicciones se transforman desde coordenadas relativas a la cuadrícula a coordenadas absolutas en la imagen original, invirtiendo el escalado y padding aplicados durante el preprocesamiento.

Para eliminar predicciones redundantes, YOLOv8 aplica Non-Maximum Suppression (NMS) con un enfoque optimizado:

- Las predicciones se ordenan por puntuación de confianza.
- Se selecciona la predicción con mayor confianza y se eliminan las predicciones con una IoU superior a un umbral con la seleccionada.
- Se itera hasta procesar todas las predicciones, logrando una supresión eficiente de detecciones duplicadas.

Esta estrategia es particularmente efectiva para documentos históricos, donde elementos como párrafos o secciones pueden generar múltiples detecciones válidas pero redundantes.

3.3.3. Comparativa con benchmarks estándar

A continuación se presentan los resultados oficiales de los modelos seleccionados sobre el conjunto de validación COCO `val2017`, utilizando la métrica estándar mAP@[.50:.95]:

Tabla 3.1: Comparativa de resultados en COCO val2017

Modelo	Tamaño entrada	mAP (%)	Fuente	Fecha
Detectron2 – Mask R-CNN R50-FPN 3×	lado corto = 800 px	41.0	Detectron2 Model Zoo [49]	2020-04-01
Ultralytics YOLOv8-Large (yolov8l.pt)	640×640 px	52.9	Ultralytics YOLOv8 Docs [50]	2023-01-10

Ambos modelos fueron entrenados sobre COCO train2017 y evaluados sobre COCO val2017. Los resultados, extraídos de las implementaciones oficiales de cada framework, permiten realizar una comparación razonable bajo las mismas condiciones de evaluación. Aunque los tamaños de entrada difieren, se trata de las configuraciones recomendadas por los autores para obtener un equilibrio óptimo entre precisión y rendimiento. Además, el número de parámetros de ambos modelos es muy similar (43.934.184 en Mask R-CNN y 43.632.924 en YOLOv8-L), lo cual refuerza la comparabilidad.

3.4 Corpus

La selección adecuada del corpus es un factor determinante en el éxito de proyectos de segmentación de documentos históricos. Un corpus bien estructurado proporciona los datos necesarios para entrenar modelos robustos, mientras que sus características particulares influyen directamente en la capacidad de generalización del sistema. En este trabajo, se ha optado por utilizar corpus especializados en documentos manuscritos con layouts complejos, que presentan los desafíos reales encontrados en archivos históricos: variaciones caligráficas, estructuras no estandarizadas y elementos específicos como anotaciones marginales o sellos notariales.

3.4.1. Corpus OHG (Online Handwritten Girona)[1]

Descripción General del Corpus

El Corpus OHG (Online Handwritten Girona) constituye un conjunto de datos compuesto por escrituras notariales digitalizadas procedentes de la región de Girona. Este corpus se centra en documentos manuscritos históricos, principalmente relacionados con transacciones legales, contratos y otros registros notariales de valor histórico y jurídico. La digitalización y estructuración de este corpus tiene un propósito doble:

- **Preservación digital:** Evitar la pérdida de información valiosa debido al deterioro físico de los documentos.
- **Accesibilidad y análisis:** Permitir el estudio automatizado mediante técnicas de Visión Artificial y Reconocimiento de Texto Manuscrito (HTR, Handwritten Text Recognition).

Composición del Corpus

El corpus está formado por un conjunto significativo de documentos manuscritos digitalizados, que incluyen:

- **Imágenes digitalizadas:** Capturadas en alta resolución para preservar detalles como el trazo de la caligrafía y las anotaciones marginales.
- **Metadatos asociados:** Información sobre el notario, fecha de emisión, tipo de escritura y localización geográfica.
- **Etiquetado estructural:** Algunas páginas están segmentadas y etiquetadas manualmente para identificar campos específicos como nombres, fechas, firmas, y títulos.

Las anotaciones originales se encuentran en formato **PAGE-XML**, ampliamente utilizado en proyectos de digitalización de documentos históricos manuscritos.

Tamaño del Dataset

En este proyecto se está utilizando un subconjunto del corpus OHG, compuesto por:

- **450 imágenes para entrenamiento.**
- **80 imágenes para validación.**
- **66 imágenes para test.**

Modificaciones Realizadas

Se realizaron modificaciones en el dataset para corregir errores y optimizar su estructura. Las anotaciones se han reorganizado en cuatro categorías principales:

- **NOP:** Sellos o firmas específicas presentes en el documento.
- **TIP:** Encabezados previos al párrafo, que identifican secciones dentro de la escritura.

[1] `https://www2.udg.edu/tabid/18216/Default.aspx`

- **PAR:** Párrafos completos, donde se encuentran las transacciones o acuerdos legales.
- **PAG:** Referencia a la página completa, útil para el análisis global del documento.

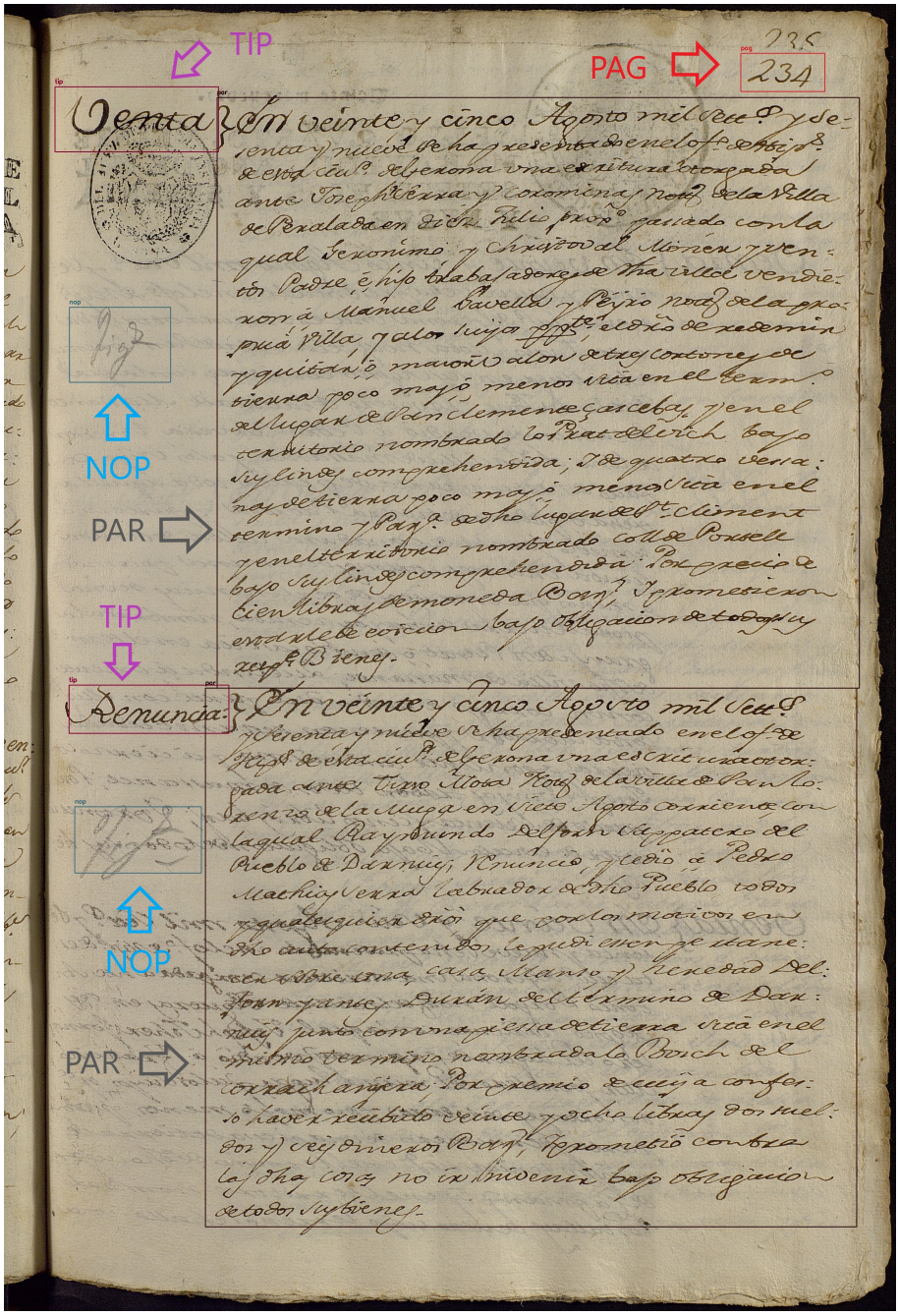

Figura 3.12: Ejemplo de página del corpus OHG con anotaciones visuales normalizadas. Aclaraciones añadidas a mano, para un aspecto más visual.

Estas modificaciones se realizaron con el objetivo de normalizar el esquema de anotación y facilitar el entrenamiento de modelos de segmentación y detección (véase Figura 3.12). En la versión original del corpus se distinguían dos tipos de anotaciones marginales: las incluidas en el momento de redacción del documento y aquellas añadidas con posterioridad. Ambas fueron unificadas en una única categoría, al no considerarse esta distinción relevante para los fines del experimento. Del mismo modo, los párrafos se etiquetaban originalmente como «párrafo» y «continuación» de párrafo; esta diferen-

ciación también se eliminó por no aportar valor discriminativo en el contexto del modelo utilizado.

Estructura del Documento

Cada documento en el corpus está estructurado siguiendo un esquema lógico que permite identificar secciones relevantes:

1. **Encabezado:** Datos sobre el notario, el lugar y la fecha de emisión.
2. **Cuerpo del documento:** El contenido principal del acuerdo legal o transacción.
3. **Firmas y sellos:** Validación legal del documento.
4. **Anotaciones marginales:** Comentarios adicionales o correcciones manuscritas.

Procesos de Digitalización y Etiquetado

Los documentos han sido digitalizados utilizando escáneres de alta resolución, garantizando una captura precisa de los detalles en la escritura manuscrita. Además, se han aplicado técnicas de preprocesado para mejorar el contraste y reducir el ruido, facilitando así el análisis.

Aplicaciones Tecnológicas

El corpus OHG se ha utilizado en múltiples investigaciones relacionadas con la segmentación y el reconocimiento de texto manuscrito:

- **Detección de Firmas:** Localización precisa de firmas y sellos notariales.
- **Segmentación de campos:** Extracción automatizada de datos como nombres, fechas y lugares.
- **Reconocimiento de Texto Manuscrito (HTR):** Transcripción automática del contenido textual.

Un ejemplo relevante es la tesis doctoral de Lorenzo Quirós Díaz, titulada "Layout Analysis for Handwritten Documents", donde se emplea este corpus para el análisis probabilístico del diseño de documentos manuscritos y la determinación del orden de lectura (Reading Order Determination). En su trabajo, Quirós desarrolla un marco teórico para la detección de líneas base (Baseline Detection) y la segmentación de regiones (Region Segmentation), utilizando modelos de aprendizaje automático para procesar colecciones de miles de imágenes manuscritas, incluyendo el corpus OHG [37].

La preservación y digitalización de estos documentos forma parte de un proyecto más amplio de colaboración entre instituciones académicas y el Archivo Histórico de Girona [38], cuyo objetivo es salvaguardar el patrimonio documental y facilitar su acceso a investigadores e historiadores. Además, este corpus ha servido como referencia en competiciones internacionales de detección de líneas base en documentos históricos, como la cBAD (Competition on Baseline Detection in Archival Documents) [39], donde diferentes equipos compiten para desarrollar los algoritmos más precisos para esta tarea.

Desafíos Asociados

El análisis de estos documentos enfrenta múltiples desafíos:

- **Deterioro físico:** El paso del tiempo afecta la legibilidad y la conservación de los textos.
- **Variabilidad en la caligrafía:** Diferentes notarios y épocas presentan estilos de escritura únicos.
- **Errores en el etiquetado:** La segmentación manual puede introducir errores en los datos de entrenamiento.
- **Ruido y manchas:** Muchos documentos contienen manchas, roturas y marcas que dificultan el reconocimiento.

Referencias Importantes

Las principales referencias sobre este corpus incluyen los trabajos de Quirós [37] y el proyecto de digitalización de la Universidad de Girona [38].

3.4.2. Corpus VORAU-253 (Vorau Abbey Cod. 253)[2]

Descripción General del Corpus

El Corpus VORAU-253 [37] es un manuscrito musical medieval (ca. 1450) preservado en la abadía de Vorau (Austria). Su maquetación combina pentagramas, texto cantado y capitulares ornamentadas, lo que lo convierte en un reto significativo para el análisis de estructura de documentos (DLA, Document Layout Analysis).

Composición del Corpus

El corpus está formado por [52]:

- **229 imágenes digitalizadas** en color (300 ppi, 24-bit RGB).
- **Metadatos asociados:** Información detallada sobre procedencia y datación.
- **Etiquetado manual** de tres regiones principales.

Las anotaciones originales se encuentran en formato **PAGE-XML**, ampliamente utilizado en proyectos de digitalización de documentos históricos manuscritos.

Tamaño del Conjunto de Datos

El conjunto de datos se divide en [37]:

- **109 imágenes para entrenamiento.**
- **20 imágenes para validación.**
- **100 imágenes para test.**

[2] `https://manuscripta.at/`

Modificaciones Realizadas

Sobre este dataset no se ha realizado ninguna modificación o ajuste.

Estructura del Documento

Cada folio del manuscrito presenta una estructura jerárquica característica [52]:

1. **Bloques horizontales** `staff`: Pentagramas musicales que estructuran la página
2. **Texto cantado** `lyrics`: Alineado cuidadosamente bajo cada pentagrama
3. **Capitulares decoradas** `drop-capital`: Elementos ornamentales que marcan inicios de sección

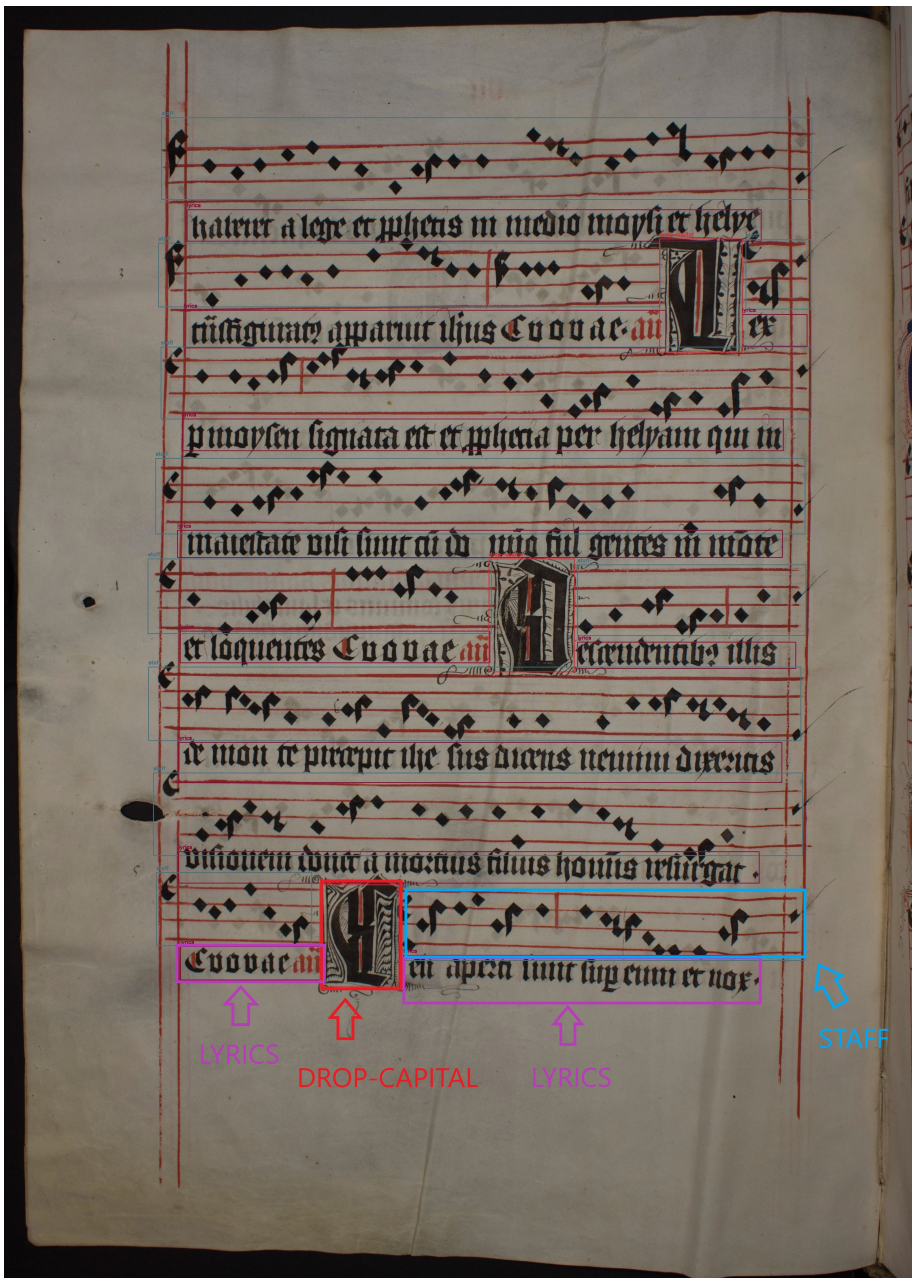

Figura 3.13: Ejemplo de página del corpus Vorau con anotaciones visuales normalizadas. Aclaraciones añadidas a mano, para un aspecto más visual.

Procesos de Digitalización y Etiquetado

Los documentos han sido digitalizados mediante un proceso de escaneado en color de alta calidad (300 ppi, 24-bit RGB) con corrección mínima de inclinación [37]. El proceso de anotación se ha realizado utilizando la herramienta PAGE-Editor, y cada página ha sido validada por expertos musicólogos para garantizar la correcta correspondencia entre notación musical y texto [53].

Aplicaciones Tecnológicas

El corpus ha sido empleado en diversos experimentos de análisis documental [52, 53]:

- **Segmentación de regiones:** Implementación y evaluación de arquitecturas Mask R-CNN
- **Detección de líneas base musicales:** Desarrollo de algoritmos específicos para notación musical
- **Estudios comparativos:** Análisis del impacto del tamaño del conjunto de entrenamiento en el rendimiento de los modelos

Desafíos Asociados

El análisis de este corpus presenta varios retos específicos [37]:

- **Interferencia visual:** Los pentagramas musicales pueden dificultar la detección precisa de texto
- **Complejidad ornamental:** Las capitulares decoradas presentan bordes difusos y patrones intrincados
- **Ambigüedad contextual:** La distinción entre texto cantado y anotaciones marginales requiere comprensión del contexto musical

El conjunto de datos completo está disponible públicamente en el repositorio Zenodo [51], facilitando su uso en investigaciones sobre análisis de documentos históricos y musicales.

3.4.3. Transformaciones

Dado que las anotaciones originales en ambos corpus se encuentran en formato **PAGE-XML**, se desarrolló un **script** para transformarlas a un formato compatible con YOLOv8. El proceso incluyó las siguientes etapas:

- Conversión inicial de los archivos PAGE-XML al formato **COCO-JSON**, facilitando su integración con Detectron2.
- Unificación de las anotaciones en un único archivo **JSON estructurado**, siguiendo el esquema de categorías y bounding boxes utilizado en Detectron2.
- Conversión final al formato **YOLOv8**, generando los correspondientes archivos `.txt` (por imagen) y un archivo `.yaml` con la configuración del dataset.

3.5 Métricas

En esta sección se describen las métricas utilizadas para evaluar el rendimiento de los modelos de segmentación y clasificación implementados con Detectron2.

3.5.1. IoU (Intersection over Union)

La IoU (Intersección sobre Unión) es una métrica fundamental que compara el solapamiento entre dos cajas: una predicha por el modelo y una de verdad (ground truth) [40]. Se calcula como el área de intersección dividida por el área de unión entre ambas cajas:

$$\text{IoU} = \frac{\text{Área de intersección}}{\text{Área de unión}} \tag{3.1}$$

Un valor de IoU cercano a 1 indica que la predicción coincide muy bien con el objeto real. Se considera generalmente que una predicción es correcta si la IoU es mayor a un cierto umbral, como 0.5 o 0.75.

Para tareas de segmentación de instancias, donde se generan máscaras a nivel de píxel en lugar de simples cajas delimitadoras, la IoU se calcula de manera similar pero utilizando el número de píxeles en la intersección y unión de las máscaras predichas y ground truth.

3.5.2. AP (Average Precision)

La Precisión Media (AP) mide la calidad de las predicciones del modelo combinando dos aspectos:

- **Precisión**: proporción de predicciones que son correctas.
- **Recall (cubrimiento o cobertura)**: proporción de objetos que han sido detectados.

Se obtiene el área bajo la curva precisión-recall para cada clase.

En Detectron2, se reportan versiones específicas de AP:

- **AP50**: Precisión media con un umbral de IoU ≥ 0.5
- **AP75**: Precisión media con un umbral de IoU ≥ 0.75

AP50 es más tolerante, mientras que AP75 exige una mayor precisión en la localización de las cajas o máscaras.

3.5.3. mAP (mean Average Precision)

El mAP es la media del AP sobre diferentes clases y distintos umbrales de IoU. Se calcula promediando los AP para umbrales que van desde 0.5 hasta 0.95 en pasos de 0.05, lo cual lo convierte en una métrica especialmente exigente.

Un modelo con alto mAP no solo hace buenas predicciones en general, sino que mantiene una calidad alta en múltiples condiciones de evaluación.

3.5.4. AR (Average Recall)

La cobertura media (AR) es una métrica complementaria que evalúa la capacidad del detector para encontrar todos los objetos presentes en una imagen. Mientras que la precisión media (AP) evalúa tanto la precisión como el recall, el AR se centra específicamente en maximizar la cobertura de detección.

El AR se calcula como el recall máximo dado un número fijo de detecciones por imagen (normalmente 1, 10 o 100), promediado sobre todos los umbrales de IoU (de 0.5 a 0.95).

Se suelen reportar diferentes versiones:

- **AR@1**: Recall medio considerando solo la detección con mayor confianza por imagen.
- **AR@10**: Recall medio considerando hasta 10 detecciones por imagen.
- **AR@100**: Recall medio considerando hasta 100 detecciones por imagen.

Un AR alto indica que el modelo es capaz de localizar correctamente una gran proporción de los objetos presentes, independientemente de la clase asignada. Esto resulta especialmente relevante en contextos donde la exhaustividad en la detección es prioritaria, como en el análisis de documentos históricos donde es crucial no perder ningún elemento estructural.

3.5.5. Comparativa de Métricas

La Tabla 3.2 proporciona un resumen comparativo de las métricas utilizadas para la evaluación de modelos en Detectron2, destacando sus características y aplicaciones principales.

Tabla 3.2: Comparativa de métricas de evaluación utilizadas en Detectron2

Métrica	Definición	Características	Aplicación
IoU	Ratio del área de intersección sobre el área de unión entre predicción y ground truth	- Valores de 0 a 1 - Mide precisión espacial	- Evaluación básica de solapamiento - Determina si una detección es válida
AP50	Precisión media usando IoU ≥ 0.5 como umbral	- Más permisivo - Sensible a falsos positivos	- Evaluación general de rendimiento - Útil en aplicaciones con requisitos moderados de precisión espacial
AP75	Precisión media usando IoU ≥ 0.75 como umbral	- Más exigente - Requiere localización precisa	- Evaluaciones donde la localización exacta es importante - Aplicaciones médicas, análisis preciso de documentos
mAP	Media de AP sobre múltiples umbrales de IoU (0.5-0.95)	- Visión integral - Robusta a diferentes condiciones	- Comparación justa entre modelos - Métrica estándar en competiciones y benchmarks
AR	Cobertura media sobre múltiples umbrales de IoU para un número fijo de detecciones por imagen	- Mide exhaustividad - Evalúa capacidad de cobertura - Varía según detecciones permitidas (AR@1, AR@10, AR@100)	- Prioritario en sistemas donde no perder objetos es crucial - Evaluación de cobertura en documentos históricos complejos

Para la evaluación de segmentación de documentos históricos, estas métricas resultan particularmente relevantes, ya que permiten cuantificar tanto la capacidad del modelo para identificar correctamente los diferentes elementos estructurales (AP) como su precisión en la delimitación espacial de estos elementos (IoU) y su exhaustividad para detectar todos los elementos presentes (AR). Esto es especialmente importante dado que los documentos históricos presentan variabilidad en el diseño, deterioro y otros factores que complican la segmentación precisa.

3.6 Entrenamiento y Evaluación

En esta sección se detallan los parámetros de entrenamiento utilizados para cada framework, así como la justificación de las decisiones tomadas en la configuración de cada uno.

3.6.1. Entrenamiento

En esta sección se detallan las configuraciones de entrenamiento adoptadas para cada uno de los modelos empleados, Detectron2 y YOLOv8. A pesar de tratarse de arquitec-

turas distintas y con filosofías de implementación diferentes, se ha procurado definir una configuración lo más equivalente posible en términos de hiperparámetros relevantes, con el objetivo de garantizar una base justa para la comparación posterior de resultados.

Entrenamiento con Detectron2

Para Detectron2 se utilizó la arquitectura Mask R-CNN con backbone ResNet-50 y FPN. Se desactivó la segmentación de máscaras, empleando únicamente detección basada en bounding boxes, al considerarse suficiente para representar la estructura de los documentos trabajados. La incorporación de segmentación a nivel de píxel se contempla como una posible mejora futura.

Se llevaron a cabo tres variantes de entrenamiento: (1) desde cero con pesos aleatorios, (2) con pesos COCO descargados de un repositorio público, y (3) con pesos preentrenados ajustados al dominio documental (PubLayNet). Estas variantes permitieron analizar cómo influye el punto de partida en la capacidad de adaptación del modelo.

El entrenamiento se planificó para un máximo de 20.000 épocas, con batch size 4 y aumentos de datos moderados (principalmente volteo horizontal aleatorio). Se utilizó el optimizador SGD con tasa de aprendizaje de 0.0025, momentum de 0.9, weight decay de 0.0001 y un warm-up de 3 épocas.

Dado que Detectron2 no dispone de un sistema nativo de early stopping, se modificó un script personalizado para implementar este mecanismo. Este ajuste modificó el flujo de entrenamiento por defecto del framework, introduciendo validación al final de cada época (mediante el parámetro 'EVAL_PERIOD') y estableciendo una paciencia configurable sobre la métrica 'bbox/AP'. Además, se configuró un guardado de checkpoints que permite almacenar automáticamente el mejor modelo alcanzado ('best.pth') y el último ('last.pt'), replicando el comportamiento habitual de otros frameworks como YOLOv8.

Entrenamiento con YOLOv8

Para YOLOv8 se utilizó la variante `yolov8l`, seleccionada por su similitud en número de parámetros con la arquitectura usada en Detectron2, facilitando así una comparación más equilibrada.

También en este caso se entrenaron tres variantes del modelo: (1) desde cero con pesos aleatorios, (2) con pesos COCO descargados de un repositorio público, y (3) con pesos ajustados al dominio (DocLayNet). Todas las variantes compartieron el mismo conjunto de datos y configuraciones de entrenamiento, lo que garantiza la validez de las comparaciones.

El entrenamiento se configuró con un máximo de 20.000 épocas, aunque se empleó el sistema de early stopping integrado en el framework, fijando un valor de paciencia (patience) de 50. El batch size fue de 4, la resolución de entrada de 800×800 píxeles, y se aplicaron técnicas de aumentación como flipping, randaugment y modificaciones cromáticas ligeras.

El optimizador seleccionado automáticamente por el sistema fue SGD, con una tasa de aprendizaje de 0.0.0025, momentum de 0.9 y 3 épocas de warm-up. Aunque inicialmente se definieron valores manuales, estos fueron sobreescritos por la configuración automática del framework durante el proceso de ajuste interno.

Comparativa de configuración de entrenamiento

A continuación se presenta una tabla comparativa con los principales parámetros de entrenamiento utilizados para ambos modelos:

Tabla 3.3: Mejor configuración de entrenamiento

Parámetro	Detectron2	YOLOv8
Optimizador	SGD	SGD
Learning rate	0.0025	0.0025
Momentum	0.9	0.9
Weight decay	0.0001	0.0005
Warmup	3 epochs	3 epochs
Warmup bias LR	-	0.1
Tamaño entrada	800 px	800 px
Epochs	4000	4000
Early stopping	50 epochs	50 epochs
Batch size	4	4
Parámetros	43.9M	43.6M
Aumentación	Flip H	Flip+mosaic+rand
Framework	PyTorch 2.6	PyTorch 2.6
Backbone	ResNet-50	CSPDarknet
Neck	FPN	SPPF
Head	Mask R-CNN	Decoupled Head

Adaptaciones al Problema En ambos entornos se ajustó el número de clases a las que el corpus requirió, y se configuró un sistema de guardado y monitorización de los resultados para facilitar el seguimiento y análisis del proceso de entrenamiento. Las configuraciones también respetaron los umbrales internos de cada framework y los flujos de entrenamiento que cada uno establece por defecto, permitiendo así que la comparación se centre en los elementos esenciales: batch size, resolución de imagen, arquitectura y estrategia de optimización.

Resumen comparativo En resumen, las configuraciones son comparables en términos de objetivos y recursos disponibles, pero cada una explora enfoques distintos: Detectron2 prioriza la precisión mediante detección detallada basada en bounding boxes, mientras que YOLOv8 apuesta por una mayor eficiencia operativa con técnicas más agresivas de aumentación de datos y aprendizaje. Esta complementariedad en los enfoques proporciona una perspectiva más completa sobre las capacidades y limitaciones de cada framework en el contexto específico del análisis de documentos históricos manuscritos.

3.6.2. Evaluación

En esta sección se describe el proceso de evaluación seguido para comparar objetivamente el rendimiento de los modelos entrenados. Dado que el objetivo del trabajo es valorar la calidad de la detección estructural en documentos históricos manuscritos, se optó por emplear las métricas estándar del benchmark COCO [12], ampliamente aceptadas en tareas de detección y segmentación de instancias.

Criterios y Métricas de Evaluación

La métrica principal empleada fue la **precisión media** o *Average Precision* (AP), calculada en el rango de umbrales de solapamiento (IoU) entre 0.50 y 0.95, en saltos de 0.05,

según el protocolo propuesto por Lin et al. (2014). Se reportan además las métricas AP50 y AP75, que reflejan el rendimiento con criterios más permisivos o estrictos de solapamiento, respectivamente.

Esta elección de métricas permite evaluar no solo si un objeto ha sido detectado, sino también la precisión con la que se ha localizado. Dado que ambos frameworks permiten obtener predicciones en formato compatible con el estándar COCO, se ha garantizado una evaluación equitativa y directamente comparable.

Script Genérico de Evaluación

Para homogeneizar el proceso de evaluación, se desarrolló un script en Python que implementa el flujo completo de evaluación a partir de dos ficheros JSON: uno con las anotaciones reales (ground truth) y otro con las predicciones generadas por los modelos. Este script realiza las siguientes operaciones:

- **Remapeo de IDs de imagen**: algunos frameworks (como YOLOv8) asignan como identificador de imagen el nombre del fichero, mientras que COCO requiere un ID numérico. El script incluye una función para convertir los identificadores al formato esperado, garantizando la compatibilidad sin alterar el contenido semántico.
- **Evaluación global y por clase**: se invoca el evaluador oficial de COCO (`COCOeval`) para calcular las métricas globales (mAP, AP50, AP75), así como el *mAP por clase*, es decir, la precisión media específica de cada una de las categorías del layout. Estas métricas se imprimen en consola y se resumen al final del proceso.

Este enfoque modular y replicable permite aplicar la misma lógica de evaluación tanto a modelos entrenados con YOLOv8 como con Detectron2, manteniendo una coherencia en el análisis de resultados.

Aplicación en ambos Frameworks

En ambos casos, los modelos se evaluaron sobre el mismo conjunto de test, anotado en formato COCO, lo que permitió calcular las métricas de forma unificada. Tanto en Detectron2 como en YOLOv8, tras el proceso de inferencia, las predicciones se exportaron a archivos JSON, que posteriormente fueron evaluados utilizando el script mencionado.

Resumen Gracias a este enfoque común de evaluación, se asegura la comparabilidad entre arquitecturas y configuraciones distintas. Además, la disponibilidad de métricas por clase y de métricas agregadas permite no solo comparar el rendimiento global de los modelos, sino también identificar fortalezas y debilidades específicas en la detección de distintos elementos del layout.

3.6.3. Pesos preentrenados utilizados

Justificación del uso de pesos preentrenados: PubLayNet vs DocLayNet (YOLOv8)

En el contexto de este trabajo, centrado en la segmentación estructural de documentos manuscritos, se ha buscado utilizar modelos de aprendizaje profundo con pesos previamente entrenados en tareas específicas de **análisis documental**. La idea central ha sido aprovechar redes ya entrenadas sobre conjuntos de datos con estructuras reales y variadas, que compartan similitudes con el dominio objetivo.

Esta estrategia responde tanto a razones **computacionales** (evitar entrenamientos desde cero poco eficientes) como a criterios de **transferencia de conocimiento**, ya que los modelos preentrenados pueden proporcionar una base sólida para adaptar la segmentación al caso particular de documentos históricos manuscritos, incluso cuando el volumen de datos anotados es limitado.

En este trabajo se han explorado dos enfoques complementarios:

- **Detectron2**, utilizando un modelo preentrenado con el dataset **PubLayNet**, centrado en artículos científicos;
- **YOLOv8**, utilizando un modelo preentrenado con el dataset **DocLayNet**, que abarca una amplia variedad de documentos reales.

Framework 1: Detectron2 + PubLayNet

Se ha empleado un modelo preentrenado con el dataset **PubLayNet**, derivado de artículos científicos de **PubMed Central**, que presenta estructuras típicas de documentos académicos formateados en PDF [44].

Framework 2: YOLOv8 + DocLayNet (modelo preentrenado por NeuralShift)

Se ha empleado el modelo `doc-layout-yolov8l`, publicado por **NeuralShift** en Hugging Face, entrenado sobre el dataset **DocLayNet**, compuesto por una gran variedad de documentos reales con diversidad estructural y semántica [45].

Comparativa técnica y justificativa

Criterio	**PubLayNet (Detectron2)**	**DocLayNet (YOLOv8 - NeuralShift)**
Origen del dataset	Artículos científicos (PubMed Central)	Documentos reales (contratos, artículos…)
Volumen de imágenes	∼360.000	∼80.000
Número de clases	5	11
Formato de anotación	COCO	JSON personalizado
Resolución media	2480×3508 px	Variable
Framework	Detectron2	YOLOv8
Modelo utilizado	Mask R-CNN R50 FPN	YOLOv8l
mAP (según referencia)	≈ 0,74 [48]	≈ 0,71 [47]
Ventaja destacada	Layout técnico definido	Alta variedad estructural y semántica

Tabla 3.4: Comparativa de características entre los modelos preentrenados utilizados

Justificación académica

Aunque los datasets no son idénticos, la elección de **DocLayNet** para el modelo YOLOv8 resulta metodológicamente sólida por las siguientes razones:

1. Ambos conjuntos de datos abordan la segmentación estructural de documentos, con clases superpuestas y enfoques complementarios.
2. El modelo YOLOv8 utilizado parte de pesos entrenados específicamente para análisis documental, lo que garantiza una base sólida de conocimiento estructural.

3. La diversidad documental de DocLayNet complementa la estructura académica más rígida de PubLayNet, permitiendo evaluar la capacidad de generalización hacia un nuevo dominio.

4. El uso de modelos preentrenados en ambos frameworks permite una comparación equitativa del rendimiento bajo condiciones similares de transferencia de aprendizaje, enfocada en el análisis de layouts documentales reales.

3.7 Resultados

3.7.1. Presentación de métricas globales (AP, AP50, AP75)

Resultados para Detectron2

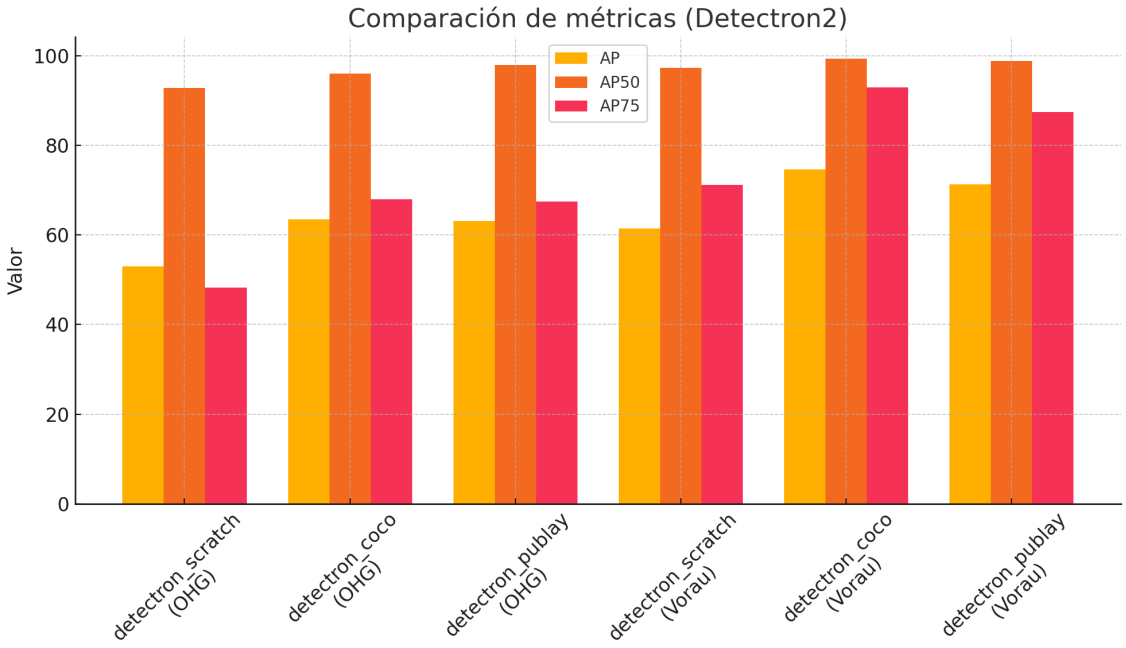

Figura 3.14: Comparación de métricas (AP, AP50, AP75) para Detectron2 en OHG y Vorau

Tabla 3.5: Resultados de AP, AP50 y AP75 para Detectron2

Modelo	AP (%)	AP50 (%)	AP75 (%)
Detectron2 (OHG, sin preentrenamiento)	53,04	92,84	48,20
Detectron2 (OHG, COCO)	**63,50**	96,00	**68,00**
Detectron2 (OHG, PubLayNet)	63,10	**97,90**	67,40
Detectron2 (Vorau, sin preentrenamiento)	61,40	97,30	71,20
Detectron2 (Vorau, COCO)	**74,60**	**99,30**	**93,00**
Detectron2 (Vorau, PubLayNet)	71,30	98,90	87,50

Los resultados obtenidos con Detectron2 sobre el conjunto OHG muestran que los valores de AP oscilan entre 53,04 % (modelo sin preentrenamiento) y 63,50 % (modelo con COCO). Los modelos preentrenados en COCO y PubLayNet presentan resultados similares en AP, con valores superiores al 63 %. La métrica AP50 es alta en todos los casos, superando el 92 %, mientras que AP75 refleja una mayor variabilidad, con el modelo sin

preentrenamiento obteniendo 48,20 % frente a valores cercanos a 68 % en los modelos preentrenados. Por ello en este apartado OHG, con COCO parece el mas completo.

En el conjunto Vorau, los modelos de Detectron2 obtienen mejores resultados en todas las métricas. La AP se sitúa entre 61,40 % (sin preentrenamiento) y 74,60 % (con COCO). AP50 alcanza hasta 99,30 %, y AP75 presenta valores elevados, con un máximo de 93,00 % en el modelo preentrenado con COCO. Estos resultados muestran, como en el caso anterior, que el preentrenamiento con COCO es el mas efectivo

Resultados para YOLOv8l

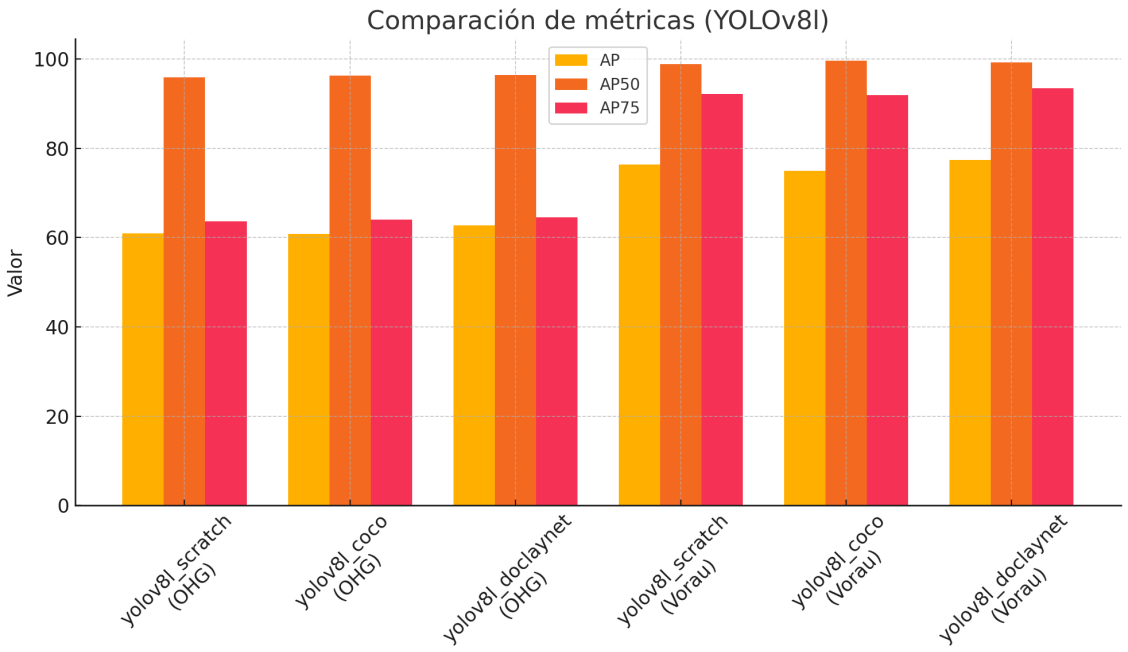

Figura 3.15: Comparación de métricas (AP, AP50, AP75) para YOLOv8l en OHG y Vorau

Tabla 3.6: Resultados de AP, AP50 y AP75 para YOLOv8l

Modelo	AP (%)	AP50 (%)	AP75 (%)
YOLOv8l (OHG, sin preentrenamiento)	60,96	95,94	63,65
YOLOv8l (OHG, COCO)	60,82	96,21	64,05
YOLOv8l (OHG, DocLayNet)	**62,74**	**96,38**	**64,54**
YOLOv8l (Vorau, sin preentrenamiento)	76,32	98,80	92,10
YOLOv8l (Vorau, COCO)	74,92	**99,55**	91,95
YOLOv8l (Vorau, DocLayNet)	**77,36**	99,17	**93,47**

En el conjunto OHG, los valores de AP obtenidos por YOLOv8l se sitúan entre 60,82 % y 62,74 %, con el modelo preentrenado en DocLayNet alcanzando el mejor resultado. Las métricas AP50 y AP75 también muestran resultados muy próximos entre modelos, con valores ligeramente superiores al 95 % y alrededor del 64 %, respectivamente. Destancando en todas las metricas los porcentaces del entrenamiento con pesos preentrenados con DocLayNet.

En Vorau, YOLOv8l alcanza sus mejores resultados en todo el análisis. Los valores de AP van desde 74,92 % hasta 77,36 %, siendo este último el correspondiente al modelo

con preentrenamiento en DocLayNet. AP50 se mantiene por encima del 98,80 % y AP75 supera el 91,90 % en todos los casos. De nuevo el preentrenamiento con DocLayNet solo se ha visto superado por la metrica del modelo preentrenado con COCO, en el caso de AP50, siendo esta la métrica mas permisiva de las presentadas.

Valoración cruzada

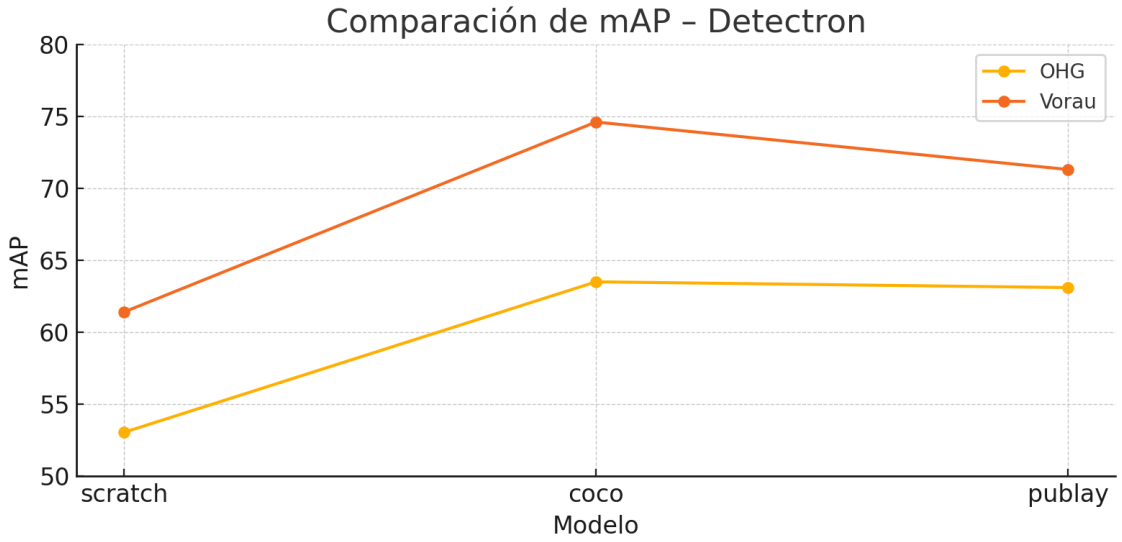

Figura 3.16: mAP de los modelos **Detectron2** en los conjuntos OHG y Vorau.

En ambos conjuntos la curva sigue la misma silueta: el entrenamiento desde cero parte del valor más bajo; el preentrenamiento con COCO introduce el salto positivo más acusado y se alza como la mejor configuración; finalmente, el ajuste en PubLayNet conserva buena parte de la ganancia, aunque ya sin superar a COCO. La línea de Vorau se mantiene paralela a la de OHG y la adelanta sistemáticamente en torno a diez puntos, lo que sugiere que las tres variantes se comportan de manera muy similar en los dos dominios, pero con un sesgo general a favor de Vorau.

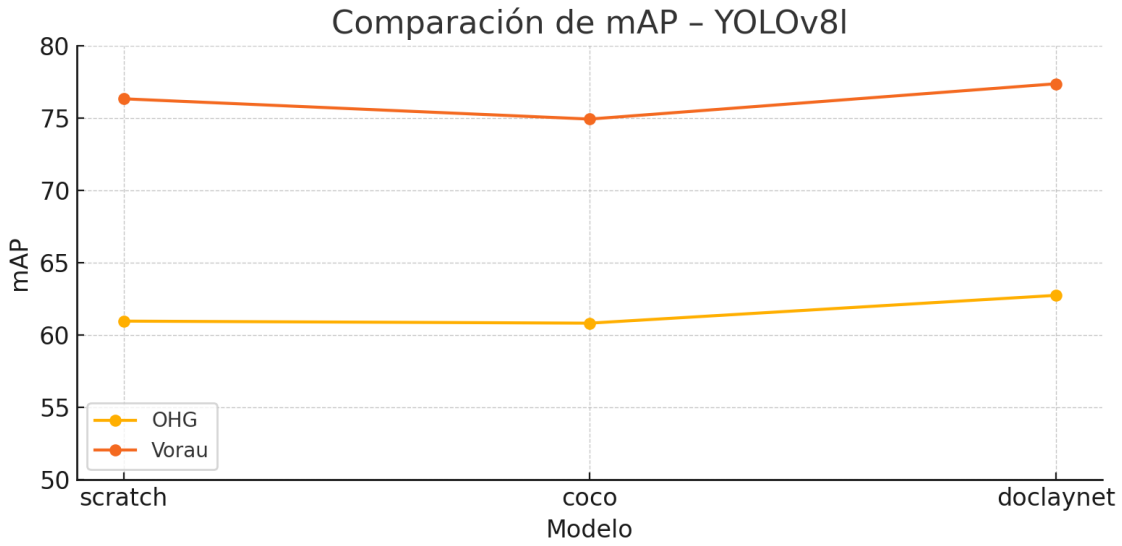

Figura 3.17: mAP de los modelos **YOLOv8l** en los conjuntos OHG y Vorau.

Para YOLOv8l se repite la pauta paralela, aunque la altura absoluta de las curvas es mayor. El modelo entrenado desde cero ya arranca con un mAP notable —sobre todo en Vorau—; el ajuste con COCO, lejos de impulsar el rendimiento, provoca una leve caída en ambos datasets; el entrenamiento adicional con DocLayNet revierte esa pérdida y firma el mejor resultado global. La separación vertical entre Vorau y OHG ronda los quince puntos a lo largo de toda la secuencia, lo que indica que la contribución específica de cada preentrenamiento es prácticamente idéntica en los dos conjuntos.

Tabla 3.7: Comparativa de mAP entre Detectron2 y YOLOv8l en OHG y Vorau.

Modelo	**OHG (%)**	**Vorau (%)**
Detectron2 (sin preentrenamiento)	53,04	61,40
Detectron2 + COCO	**63,50**	74,60
Detectron2 + PubLayNet	63,10	71,30
YOLOv8l (sin preentrenamiento)	60,96	76,32
YOLOv8l + COCO	60,82	74,92
YOLOv8l + DocLayNet	62,74	**77,36**

En el conjunto OHG las diferencias entre frameworks son moderadas: ambos convergen alrededor del 63 % en su mejor configuración. En Vorau, en cambio, YOLOv8l domina con claridad; incluso su versión entrenada desde cero supera a todas las variantes de Detectron2. El modelo *YOLOv8l + DocLayNet* —77,36 %— establece el techo de rendimiento y, a falta de otras consideraciones, sería la elección lógica si la métrica de referencia fuera únicamente el mAP.

3.7.2. Evaluación del recall (AR@1, AR@10, AR@100)

Resultados para Detectron2

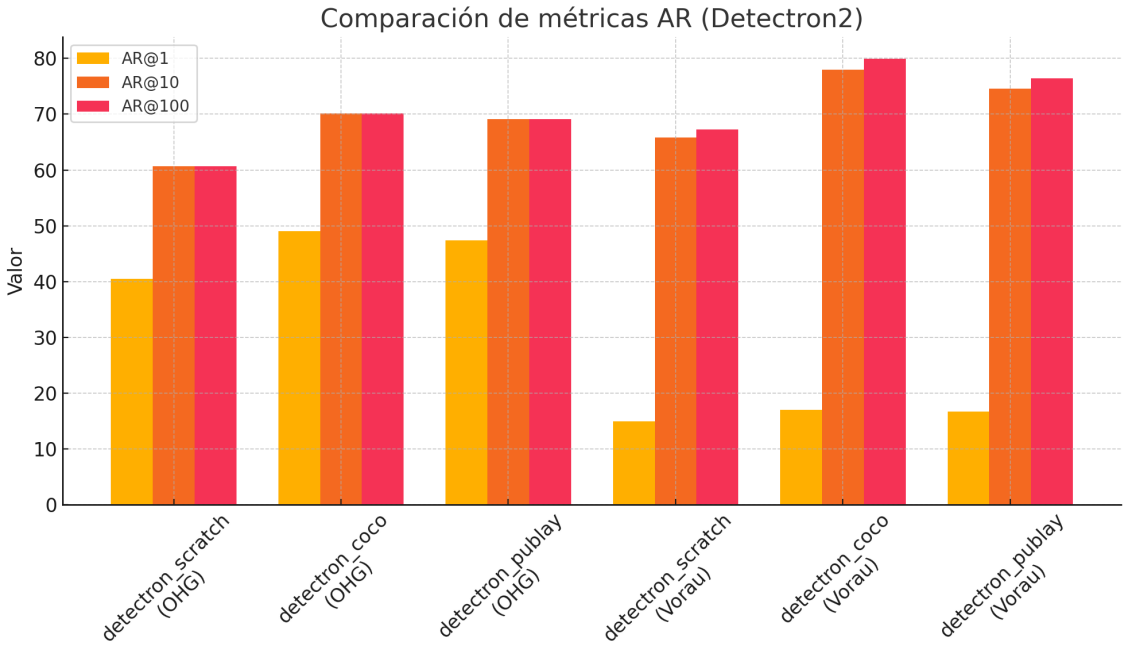

Figura 3.18: Comparación de métricas de recall (AR@1, AR@10, AR@100) para Detectron2 en OHG y Vorau

Tabla 3.8: Resultados de AR@1, AR@10 y AR@100 para Detectron2

Modelo	AR@1 (%)	AR@10 (%)	AR@100 (%)
Detectron2 (OHG, sin preentrenamiento)	40,50	60,70	60,70
Detectron2 (OHG, COCO)	**49,00**	**70,20**	**70,20**
Detectron2 (OHG, PubLayNet)	47,40	69,10	69,10
Detectron2 (Vorau, sin preentrenamiento)	14,90	65,80	67,30
Detectron2 (Vorau, COCO)	**17,00**	**78,00**	**79,90**
Detectron2 (Vorau, PubLayNet)	16,70	74,60	76,40

En el conjunto **OHG**, los valores de AR@1 oscilan entre 40,50 % y 49,00 %, siendo el modelo preentrenado con COCO el que obtiene el valor más alto. Tanto AR@10 como AR@100 coinciden en todos los modelos, indicando que el techo de recuperación se alcanza con diez propuestas. El mejor resultado global lo alcanza el modelo preentrenado en COCO, con un 70,20 %.

En el conjunto **Vorau**, los valores de AR@1 descienden notablemente, situándose entre 14,90 % y 17,00 %. Sin embargo, AR@10 y AR@100 presentan valores más elevados que en OHG. En este caso sí se observan diferencias entre AR@10 y AR@100, lo que sugiere un pequeño beneficio al permitir más predicciones. El modelo detectron_coco alcanza el valor más alto, con 79,90 % en AR@100.

Resultados para YOLOv8l

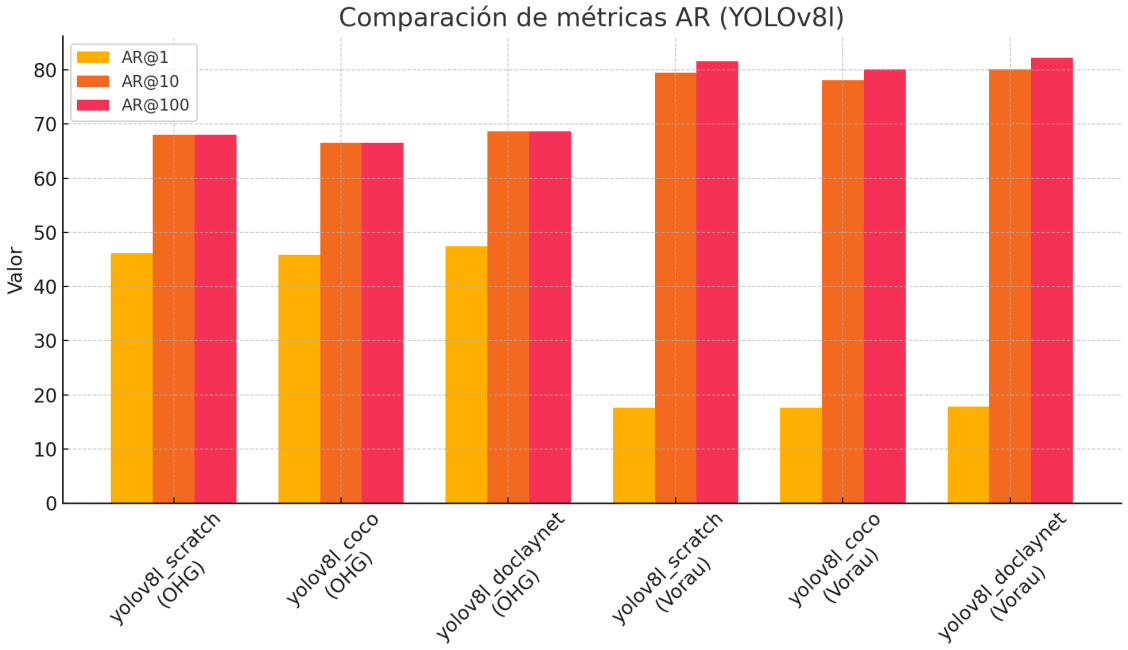

Figura 3.19: Comparación de métricas de recall (AR@1, AR@10, AR@100) para YOLOv8l en OHG y Vorau

Tabla 3.9: Resultados de AR@1, AR@10 y AR@100 para YOLOv8l

Modelo	**AR@1 (%)**	**AR@10 (%)**	**AR@100 (%)**
YOLOv8l (OHG, sin preentrenamiento)	46,20	68,00	68,00
YOLOv8l (OHG, COCO)	45,90	66,50	66,50
YOLOv8l (OHG, DocLayNet)	**47,40**	**68,70**	**68,70**
YOLOv8l (Vorau, sin preentrenamiento)	17,60	79,50	81,60
YOLOv8l (Vorau, COCO)	17,60	78,10	80,10
YOLOv8l (Vorau, DocLayNet)	**17,80**	**80,10**	**82,20**

En el conjunto **OHG**, los valores de AR@1 se encuentran entre 45,90 % y 47,40 %, con el modelo preentrenado en DocLayNet alcanzando el valor más alto. AR@10 y AR@100 son idénticos en todos los modelos, lo que indica que se alcanza la máxima recuperación con diez predicciones. También es destacable que el modelo sin preentrenamiento obtiene resultados muy competitivos (68,00 %).

En el conjunto **Vorau**, los valores de AR@1 son significativamente más bajos, entre 17,60 % y 17,80 %, pero AR@10 y AR@100 muestran una mejora sustancial respecto a OHG. El modelo preentrenado con DocLayNet alcanza los mejores resultados, con un 82,20 % en AR@100, seguido por el modelo sin preentrenamiento con 81,60 %.

Valoración cruzada del recall medio (mRecall)

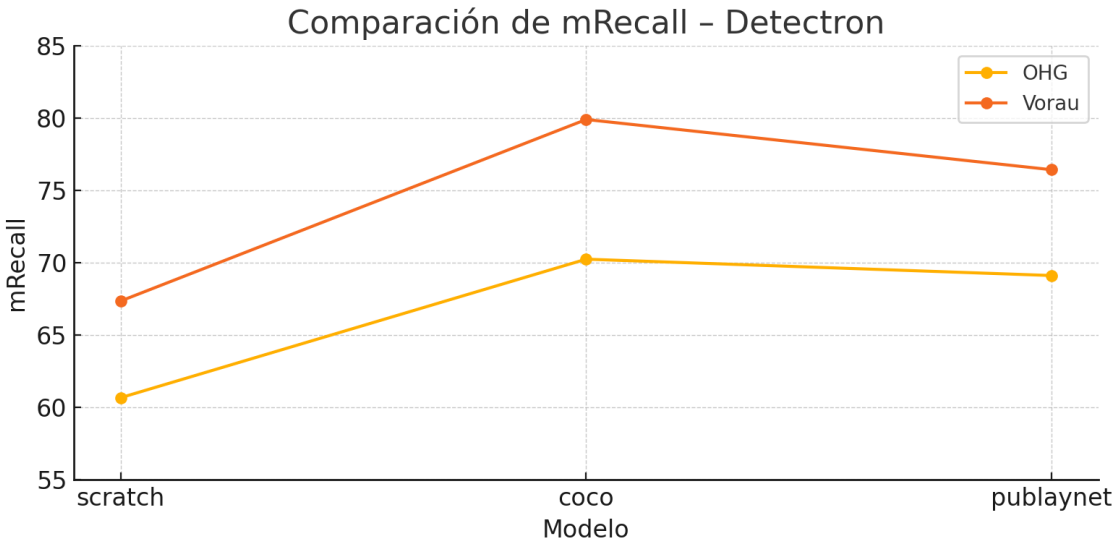

Figura 3.20: mRecall de los modelos **Detectron2** en los conjuntos OHG y Vorau.

La evolución es casi idéntica en ambos dominios. En los dos conjuntos, el entrenamiento desde cero parte de la cota más baja; el salto importante llega al incorporar el preentrenamiento en **COCO**, que fija el máximo absoluto; la afinación con PubLayNet conserva buena parte de la ganancia, aunque desciende ligeramente. La curva de Vorau mantiene una separación estable de unos siete a diez puntos por encima de la de OHG, lo que indica que la jerarquía interna de configuraciones es la misma en los dos datasets y que el aumento de dificultad (o la diferencia de dominio) se refleja únicamente como un desplazamiento vertical.

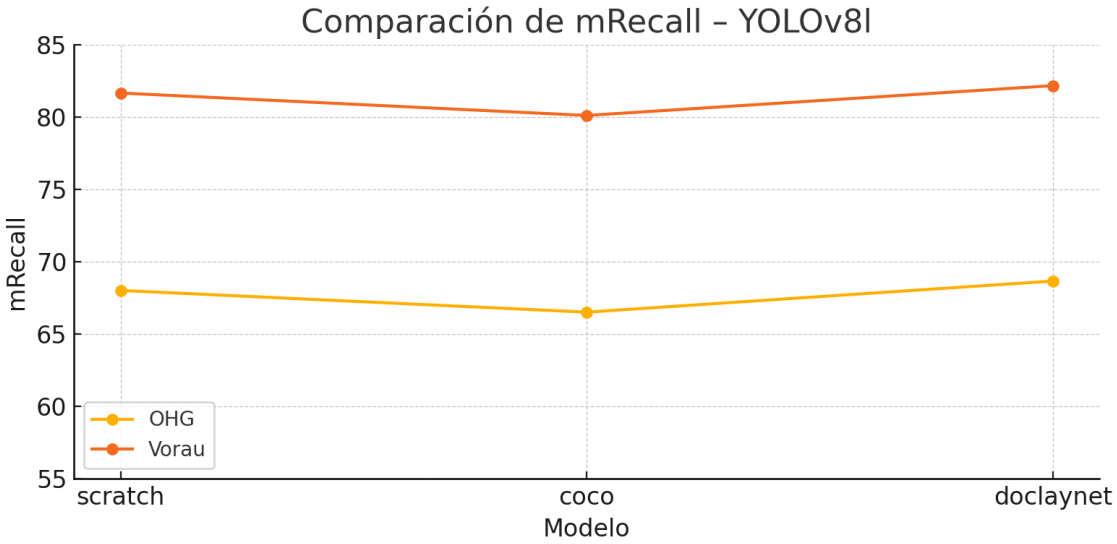

Figura 3.21: mRecall de los modelos **YOLOv8l** en los conjuntos OHG y Vorau.

En YOLOv8l el punto de partida ya es alto, sobre todo en Vorau. El preentrenamiento en **COCO** no aporta valor añadido: provoca una leve bajada en ambos dominios. El ajuste en **DocLayNet** revierte esa pérdida y se alza como la mejor opción general. De nuevo, la

diferencia entre Vorau y OHG se mantiene prácticamente constante (aprox.12–14 puntos) a lo largo de toda la secuencia, de modo que el efecto de cada preentrenamiento es esencialmente paralelo en los dos conjuntos.

Tabla 3.10: Comparativa de recall medio (mRecall) entre Detectron2 y YOLOv8l en OHG y Vorau.

Modelo	OHG (%)	Vorau (%)
Detectron2 (sin preentrenamiento)	60,67	67,35
Detectron2 + COCO	**70,24**	79,89
Detectron2 + PubLayNet	69,11	76,42
YOLOv8l (sin preentrenamiento)	68,01	81,65
YOLOv8l + COCO	66,51	80,10
YOLOv8l + DocLayNet	68,66	**82,16**

En **OHG**, los valores oscilan entre 60,67 % y 70,24 %. El mejor resultado lo firma *Detectron2 + COCO*, con una ligera ventaja sobre el resto y una distancia de apenas dos puntos respecto a la mejor versión de YOLOv8l.

En **Vorau**, todas las curvas se desplazan al alza. Aquí son los modelos YOLOv8l los que dominan con claridad: los tres superan el 80 % y el ajuste con DocLayNet alcanza 82,16 %, por delante del máximo de Detectron2 (79,89 % con COCO).

En conjunto, *YOLOv8l + DocLayNet* obtendría la mayor puntuación si el criterio fuese exclusivamente la recuperación en Vorau, mientras que *Detectron2 + COCO* ofrece la mejor marca en OHG y un desempeño muy competitivo en el otro dominio. Si se busca el modelo más equilibrado entre ambos conjuntos, la opción de **Detectron2 + COCO** resulta atractiva por su combinación de primer puesto en OHG y segundo lugar (a poca distancia del líder) en Vorau.

3.7.3. Precisión por clases (mAP)

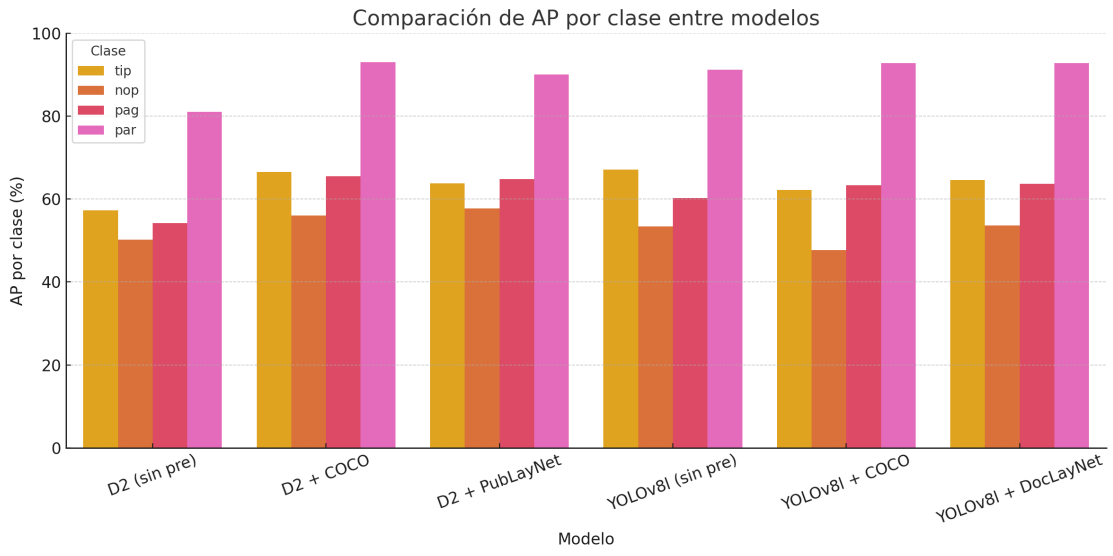

Figura 3.22: Comparativa del rendimiento por clase entre los diferentes modelos evaluados

Tabla 3.11: Rendimiento detallado por clase (AP por clase) para cada modelo evaluado

Modelo	tip (%)	nop (%)	pag (%)	par (%)
Detectron2 (sin pre)	50,03	41,95	44,46	75,72
Detectron2 + COCO	59,07	47,64	**57,63**	89,80
Detectron2 + PubLayNet	57,47	**52,10**	56,23	86,49
YOLOv8l (sin pre)	**60,54**	43,63	51,18	88,49
YOLOv8l + COCO	54,38	40,78	57,52	90,61
YOLOv8l + DocLayNet	56,43	46,79	56,85	**90,90**

El análisis desglosado por clase (Figura 3.22 y Tabla 3.11) permite comprender con mayor precisión cómo se comporta cada modelo en relación con los distintos elementos estructurales presentes en el corpus **OHG**. Como se ha comentado anteriormente, existe una clara desigualdad en la distribución de clases: los párrafos (*par*) son la categoría dominante, las cabeceras (*tip*) tienen una presencia moderada, y las firmas o marcas marginales (*nop*) y los números de página (*pag*) aparecen con menor frecuencia.

En la clase *tip*, YOLOv8l parte con un rendimiento muy elevado incluso sin preentrenamiento (67,14 %) y se mantiene estable con los pesos de COCO y DocLayNet. Detectron2 también mejora considerablemente al incorporar preentrenamiento, alcanzando un máximo de 66,53 % con COCO. Esto indica que se trata de una clase visualmente estructurada y relativamente fácil de aprender.

Para *nop*, clase más escasa y visualmente heterogénea, todos los modelos muestran valores más bajos en general. Aun así, se observan mejoras claras al usar pesos preentrenados. Detectron2 pasa de 50,19 % a 57,74 % con PubLayNet, y YOLOv8l mejora de 47,74 % a 53,58 % con DocLayNet. Esto sugiere que los pesos preentrenados ayudan a cubrir mejor clases con menor representación y mayor variabilidad visual.

En el caso de *pag*, los resultados también mejoran con el preentrenamiento. Detectron2 logra su mejor valor con COCO (65,45 %), mientras que YOLOv8l lo hace con DocLayNet (63,64 %). Dado que esta clase es escasa y con una apariencia menos uniforme, estos datos refuerzan la idea de que los pesos preentrenados aportan información estructural útil para su identificación.

Finalmente, la clase *par*, al ser la más abundante y estructuralmente coherente, ofrece los mejores resultados en todos los casos, con valores que superan el 90 % cuando se utilizan pesos preentrenados. Detectron2 con COCO alcanza el máximo (92,95 %), seguido muy de cerca por YOLOv8l con COCO y DocLayNet (ambos con 92,82 %).

En conjunto, los resultados refuerzan que el preentrenamiento aporta ventajas importantes, sobre todo en clases poco frecuentes o con mayor complejidad visual. Además, permiten constatar que tanto COCO como PubLayNet y DocLayNet ofrecen transferencias de conocimiento útiles para la segmentación estructural de documentos manuscritos, aunque con impactos distintos según la clase y arquitectura empleada.

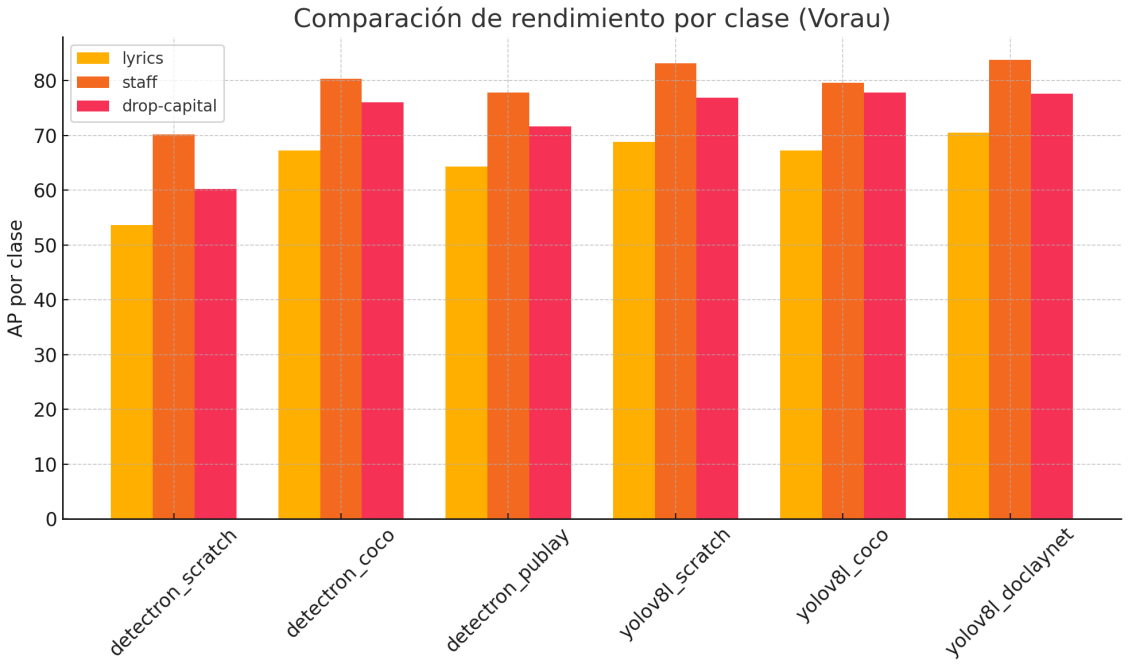

Figura 3.23: Comparativa del rendimiento por clase entre los modelos evaluados en el conjunto Vorau

Tabla 3.12: Rendimiento detallado por clase (AP por clase) en Vorau

Modelo	**lyrics (%)**	**staff (%)**	**drop-capital (%)**
Detectron2 (sin pre)	53,65	70,21	60,24
Detectron2 + COCO	67,22	80,41	76,05
Detectron2 + PubLayNet	64,32	77,86	71,66
YOLOv8l (sin pre)	68,80	83,22	76,94
YOLOv8l + COCO	67,29	79,60	**77,80**
YOLOv8l + DocLayNet	**70,56**	**83,85**	77,67

El análisis desglosado por clase en el conjunto **Vorau** (Figura 3.23 y Tabla 3.12) muestra una distribución coherente con la observada en OHG: los modelos alcanzan sus mejores valores en clases estructuradas como *staff*, mientras que los resultados son más modestos en *lyrics*, una clase posiblemente más ambigua y difícil de identificar.

En *staff*, los valores de AP son especialmente altos, superando el 80 % en la mayoría de modelos. YOLOv8l con DocLayNet obtiene el mejor resultado (83,85 %), seguido por YOLOv8l sin preentrenamiento (83,22 %) y Detectron2 con COCO (80,41 %). Esta clase parece beneficiarse especialmente de arquitecturas como YOLOv8l y de configuraciones sin necesidad estricta de preentrenamiento.

La clase *drop-capital* también presenta buenos resultados. YOLOv8l + COCO obtiene el valor más alto (77,80 %), con DocLayNet y el modelo sin preentrenamiento muy próximos. Detectron2 también mejora significativamente con preentrenamiento, alcanzando hasta 76,05 % con COCO.

Finalmente, la clase *lyrics* muestra mayor variabilidad entre modelos y valores más bajos en general. No obstante, YOLOv8l con DocLayNet logra el mejor rendimiento (70,56 %), por delante de YOLOv8l sin preentrenamiento (68,80 %) y Detectron2 con COCO (67,22 %).

En conjunto, los resultados en Vorau confirman las tendencias observadas en OHG: los modelos con preentrenamiento —especialmente YOLOv8l con DocLayNet— tienden

a obtener mejores resultados en clases con menor frecuencia o estructura más variable, mientras que las clases más regulares se benefician incluso sin pesos preentrenados.

3.7.4. Recall por clase (AR@100)

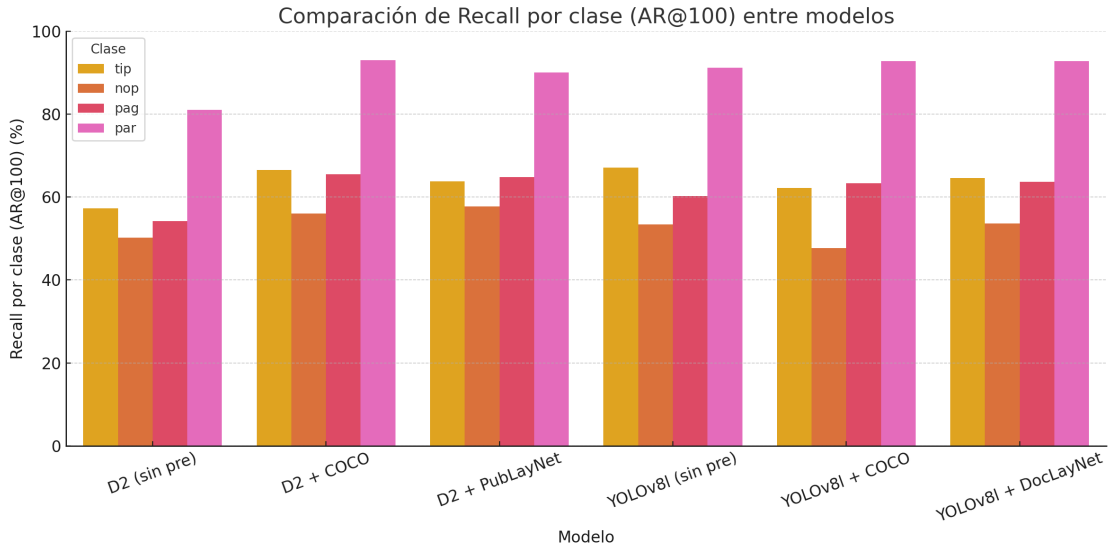

Figura 3.24: Comparación del recall por clase (AR@100) entre los diferentes modelos evaluados

Modelo	tip (%)	nop (%)	pag (%)	par (%)
Detectron2 (sin pre)	57,24	50,19	54,24	81,01
Detectron2 + COCO	66,53	56,04	**65,45**	**92,95**
Detectron2 + PubLayNet	63,78	**57,74**	64,85	90,07
YOLOv8l (sin pre)	**67,14**	53,40	60,30	91,21
YOLOv8l + COCO	62,14	47,74	63,33	92,82
YOLOv8l + DocLayNet	64,59	53,58	63,64	92,82

Tabla 3.13: Valores de recall por clase (AR@100) para cada modelo evaluado

La Figura 3.24 y la Tabla 3.13 muestran el rendimiento por clase de cada modelo en términos de **recall** (AR@100), es decir, la capacidad de recuperar correctamente instancias de cada categoría dentro de un máximo de 100 detecciones por imagen. Esta métrica permite identificar cómo varía la cobertura de detección según el tipo de elemento estructural presente en el corpus **OHG**.

Las clases más frecuentes y regulares, como *par* (párrafos), presentan los valores más altos en todos los modelos, alcanzando hasta un 92,95 % con Detectron2 + COCO y 92,82 % con ambas variantes preentrenadas de YOLOv8l. Esto confirma que los modelos aprenden bien esta clase incluso con configuraciones distintas de preentrenamiento.

En la clase *tip* (cabeceras), también se observa una mejora clara al utilizar pesos preentrenados, especialmente en Detectron2, que pasa de 57,24 % (sin pre) a 66,53 % (con COCO) y 63,78 % (con PubLayNet). YOLOv8l, por su parte, obtiene el mejor rendimiento directamente desde cero (67,14 %), aunque mantiene valores competitivos con DocLayNet.

En cuanto a *pag* (número de página), clase escasa y variable, se detecta un incremento considerable con el uso de pesos preentrenados en todos los modelos. Detectron2 alcanza 65,45 % con COCO y 64,85 % con PubLayNet; YOLOv8l llega a 63,64 % con DocLayNet.

Estos resultados demuestran que el preentrenamiento permite compensar la escasa presencia de esta clase en los datos de entrenamiento.

Finalmente, la clase *nop* (firmas o elementos marginales) presenta valores más bajos en general, pero también mejoras notables con PubLayNet y DocLayNet. El mayor valor lo obtiene Detectron2 + PubLayNet (57,74 %).

En conjunto, estos resultados evidencian que el preentrenamiento específico del dominio documental no solo mejora el rendimiento general, sino que también ayuda a equilibrar la detección entre clases frecuentes y escasas. Las diferencias entre arquitecturas y tipos de pesos confirman que la elección del modelo y su punto de partida tienen un impacto tangible en la capacidad de cobertura estructural de los documentos manuscritos.

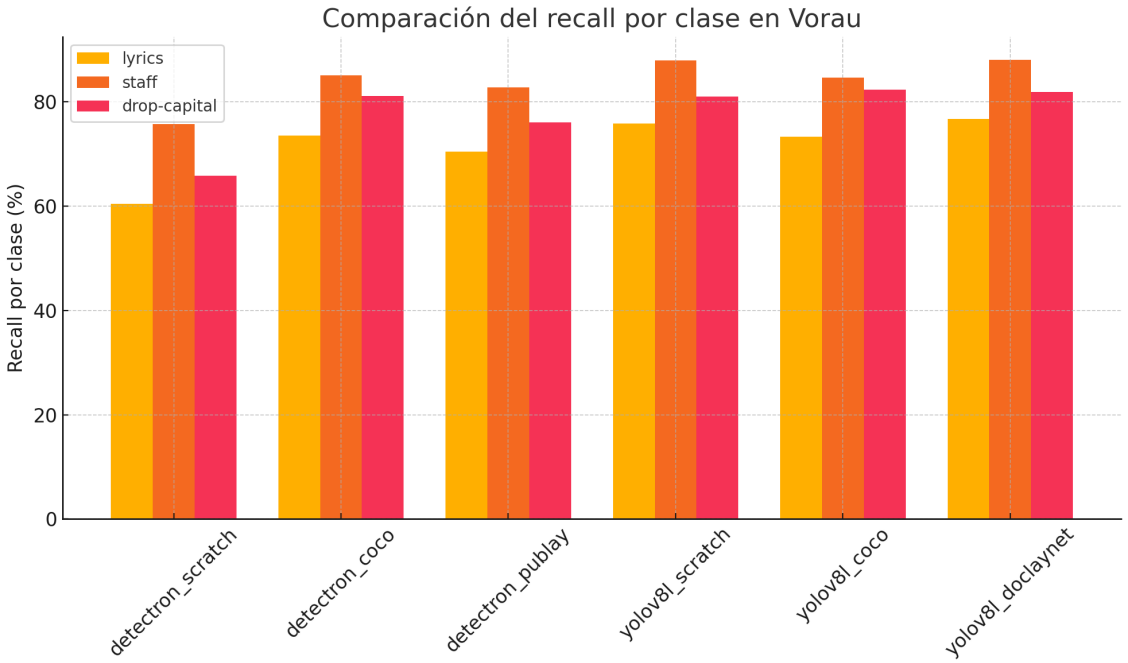

Figura 3.25: Comparativa del *recall* por clase entre los modelos evaluados en el conjunto Vorau

Tabla 3.14: Rendimiento detallado por clase (recall por clase) en Vorau

Modelo	lyrics (%)	staff (%)	drop-capital (%)
Detectron2 (sin pre)	60,45	75,73	65,86
Detectron2 + COCO	73,50	85,09	81,08
Detectron2 + PubLayNet	70,44	82,74	76,08
YOLOv8l (sin pre)	75,89	88,01	**81,03**
YOLOv8l + COCO	73,27	84,69	82,37
YOLOv8l + DocLayNet	**76,71**	**88,10**	81,86

El desglose del *recall* por clase en el conjunto **Vorau** (Figura 3.25 y Tabla 3.14) ofrece una visión complementaria al análisis de precisión. Al igual que en el mAP, la clase *staff* alcanza los mejores resultados en todos los modelos, seguida de cerca por *drop-capital*, mientras que *lyrics* presenta valores algo más moderados.

En *staff*, el mejor valor lo alcanza YOLOv8l con DocLayNet (88,10 %), seguido casi sin diferencia por YOLOv8l sin preentrenamiento (88,01 %) y Detectron2 con COCO (85,09 %). Esta clase, al estar bien estructurada visualmente, parece ser recuperada de manera efectiva incluso sin preentrenamiento en el caso de YOLOv8l.

En *drop-capital*, todos los modelos superan el 76 %, con YOLOv8l + COCO en cabeza (82,37 %), ligeramente por encima del modelo con DocLayNet y de YOLOv8l sin preentrenamiento. Detectron2 también logra un alto recall, especialmente con preentrenamiento en COCO (81,08 %).

La clase *lyrics* vuelve a ser la más desafiante. Aun así, los modelos de YOLOv8l logran los mejores valores, destacando el modelo con DocLayNet (76,71 %) y el modelo sin preentrenamiento (75,89 %), ambos por encima de cualquier configuración de Detectron2.

En resumen, los resultados de *recall* por clase en Vorau refuerzan las conclusiones obtenidas con la precisión: las clases estructuralmente claras como *staff* se detectan y recuperan muy bien, mientras que clases como *lyrics* requieren modelos más robustos o especializados. YOLOv8l, especialmente con DocLayNet, muestra una mayor capacidad de recuperación en todas las clases.

3.7.5. Influencia de la arquitectura del modelo en el rendimiento

Más allá del preentrenamiento y del tipo de datos, uno de los factores que influyen en los resultados obtenidos es la **arquitectura del modelo**. En este trabajo se han comparado dos enfoques muy distintos: por un lado, Detectron2 (basado en Mask R–CNN) y, por otro, YOLOv8l. Ambos tienen un número de parámetros similar, pero su funcionamiento interno es diferente, y eso se refleja en los resultados.

Detectron2 sigue un enfoque *two-stage*, en el que primero se generan propuestas de regiones y luego se refinan. Esto suele dar buenos resultados cuando el fondo es limpio y las estructuras están bien definidas, como ocurre en el conjunto OHG. Por su parte, YOLOv8l utiliza un enfoque *one-stage*, en el que todo se predice directamente en una sola pasada. Este diseño hace que sea más rápido y que recupere mejor elementos más variados o mal definidos, como sucede en el conjunto Vorau.

También influyen otras decisiones internas, como la forma en que cada modelo trata la información visual. Detectron2 utiliza una red clásica (ResNet) y una pirámide de características que da prioridad a las formas más grandes y generales. En cambio, YOLOv8l combina información de distintos niveles con estructuras más ligeras, lo que le permite detectar mejor detalles pequeños o elementos dispersos.

Otra diferencia importante está en la **velocidad**. YOLOv8l es más rápido en la fase de inferencia (casi el doble de rápido), lo cual puede ser importante si se quiere aplicar el modelo en tiempo real. Además, maneja mejor el entrenamiento desde cero gracias a sus técnicas de aumento de datos, lo que explica por qué obtiene mejores resultados que Detectron2 cuando ambos modelos se entrenan sin preentrenamiento.

En resumen, aunque ambos modelos tienen una capacidad similar en términos de parámetros, sus diferencias estructurales explican buena parte de las variaciones que se han visto en las métricas. Por eso, la elección del modelo no solo depende de los datos disponibles, sino también del objetivo concreto: si se busca máxima precisión en documentos limpios, Detectron2 puede ser preferible; si se necesita más flexibilidad, rapidez o capacidad de generalización, YOLOv8l ofrece ventajas claras.

3.7.6. Comparación con resultados previos: tesis doctoral de Quirós

Uno de los objetivos secundarios de este trabajo ha sido evaluar hasta qué punto los resultados obtenidos son coherentes con investigaciones anteriores en el mismo dominio. En este sentido, la tesis doctoral de **Lorenzo Quirós** [37] representa un referente, ya que

incluye experimentos sobre los conjuntos **OHG** y **VORAU**, utilizando *Detectron2* como marco principal de trabajo.

Sin embargo, es importante señalar que **la comparación con los resultados de OHG no es viable**, debido a que en este trabajo se ha utilizado una versión modificada del corpus, tal y como se ha explicado anteriormente. Las etiquetas han sido ajustadas para adaptarlas al formato de los modelos y a los objetivos específicos de esta investigación, por lo que los resultados obtenidos no son directamente comparables con los reportados por Quirós.

En cambio, el conjunto **VORAU** se ha mantenido sin modificaciones, lo que permite una comparación directa y legítima. En la Tabla 3.15 se presentan los resultados obtenidos por Quirós frente a los modelos entrenados en este trabajo, tanto en términos de **precisión media (AP)** como de **recall (AR@100)**.

Tabla 3.15: Comparación de resultados en VORAU con los obtenidos por Quirós

Modelo	**AP (%)**	**AR@100 (%)**
Quirós (sin preentrenamiento)	69,60	77,70
Quirós + ImageNet	74,50	81,40
Quirós + PubLayNet	71,40	79,20
Detectron2 (sin preentrenamiento)	61,40	67,35
Detectron2 + COCO	74,60	79,89
Detectron2 + PubLayNet	71,30	76,42
YOLOv8l (sin preentrenamiento)	76,32	81,65
YOLOv8l + COCO	74,92	80,10
YOLOv8l + DocLayNet	**77,36**	**82,16**

Los resultados muestran una **coherencia notable entre ambos estudios**, especialmente al comparar el rendimiento de Detectron2 con pesos preentrenados. En el caso de COCO, por ejemplo, se observa un AP prácticamente idéntico (74,50 % en Quirós frente a 74,60 % en este trabajo) y un valor de AR@100 muy similar (81,40 % frente a 79,89 %). Estas coincidencias refuerzan la validez y fiabilidad de los experimentos realizados.

Además, este trabajo aporta una ampliación significativa al incluir la arquitectura **YOLOv8l**, no contemplada por Quirós. Los resultados con YOLOv8l superan incluso a los de Detectron2 en varias configuraciones, alcanzando un AP de hasta 77,36 % y un AR@100 de 82,16 % con pesos de *DocLayNet*. Aunque Quirós utilizó ImageNet como preentrenamiento genérico, en este caso se ha optado por *COCO* y *DocLayNet*, este último como alternativa razonable a PubLayNet en el contexto de YOLO.

Finalmente, es interesante señalar que los resultados obtenidos en este trabajo, en el caso de DEtectron2, coinciden con una observación importante realizada por Quirós en su tesis [37]: el uso de pesos preentrenados en datasets generales (como *ImageNet*) mejora notablemente el rendimiento del enfoque directo, incluso cuando el dominio del preentrenamiento no está directamente relacionado con la tarea específica. De hecho, Quirós demuestra que no hay diferencias estadísticamente significativas entre preentrenar con *ImageNet* o con *PubLayNet*, a pesar de que este último está mucho más relacionado con el análisis de documentos. En sus palabras: *"we found no statistical difference between pretraining the model using ImageNet or PubLayNet data, which is important to be noticed as PubLayNet data is expected to be more related to our task than ImageNet data"* [37, p. 66].

Aunque en este trabajo no se ha utilizado *ImageNet*, se ha empleado *COCO* como alternativa de preentrenamiento genérico, cumpliendo un papel similar. Los buenos resultados obtenidos con COCO refuerzan esta misma idea: las primeras capas de una red convolucional aprenden filtros básicos (como bordes, líneas o texturas) que son comunes en muchas imágenes, por lo que un dataset grande y diverso permite aprender represen-

taciones más útiles. Tal y como explica Quirós: *"normally the first few convolutional layers in a CNN will learn generic filters that help the system to process common characteristics of the input images (corners, lines, etc), hence, a huge dataset as ImageNet will provide a better data distribution to learn those filters"* [37, p. 66]. Esta explicación también justifica los resultados obtenidos en este trabajo al usar COCO como base, confirmando que una buena inicialización, aunque no sea específica del dominio, puede ser muy eficaz en tareas de segmentación documental.

En resumen, la comparación directa con los datos de VORAU en la tesis de Quirós no solo valida la metodología seguida, sino que también demuestra la solidez de los resultados obtenidos, destacando a su vez las aportaciones propias de este trabajo en términos de modelos empleados y estrategias de preentrenamiento.

3.7.7. Conclusiones de los resultados experimentales

Síntesis por conjunto de datos

- **OHG.** El mejor equilibrio entre precisión (AP) y cobertura (AR) lo logra **Detectron2 + COCO**. Las diferencias con YOLOv8l son menores a 2 puntos, indicando paridad entre arquitecturas en dominios predominantemente textuales.

- **VORAU. YOLOv8l + DocLayNet** obtiene los máximos absolutos (AP = 77,36 %; AR@100 = 82,16 %), superando a Detectron2 en 2–3 puntos. La arquitectura YOLO resulta más robusta ante variabilidad gráfica (partituras, capitales ornamentales, etc.).

Efecto del preentrenamiento

- Los pesos genéricos (*COCO*) aportan saltos de +10 puntos AP y +15 puntos AR a Detectron2.

- Los pesos específicos (PubLayNet, DocLayNet) marcan la diferencia en clases escasas o heterogéneas; DocLayNet es decisivo para YOLOv8l.

- Sin preentrenamiento, YOLOv8l aventaja a Detectron2 en +7 puntos AP y +13 puntos AR, mostrando mayor capacidad de aprendizaje desde cero.

Comportamiento por clase

- **Clases frecuentes y estructuradas** (*par*, *staff*) superan el 90 % AP y AR con pesos; YOLOv8l lidera en *staff*.

- **Clases infrecuentes** obtienen claramente mejores resultados con Detectron, especialmente en la métrica AR@100.

Comparación con Quirós (2021)

La coincidencia en VORAU entre Detectron2 + COCO (este trabajo) e ImageNet (Quirós) valida la reproducibilidad. La introducción de YOLOv8l eleva el listón (AP hasta 77,36 %; AR@100 hasta 82,16 %).

Recomendaciones

1. Para colecciones textuales similares a OHG, **Detectron2 + COCO** es la opción más estable.
2. Para documentos heterogéneos o cuando se requiera recall máximo, **YOLOv8l + DocLayNet** ofrece la mejor relación rendimiento–robustez.
3. En escenarios sin pesos específicos, YOLOv8l parte con ventaja gracias a su mayor capacidad de aprendizaje desde cero.

Conclusión operativa

YOLOv8l + DocLayNet destaca como el modelo más versátil y robusto en dominios variados, mientras que Detectron2 + COCO sigue siendo preferible cuando el dominio es estrictamente textual y los pesos están alineados con la tarea.

CAPÍTULO 4
Conclusiones y Trabajos Futuros

Este capítulo presenta las conclusiones derivadas del trabajo realizado, evaluando el cumplimiento de los objetivos establecidos, identificando los principales problemas encontrados durante el desarrollo y proponiendo líneas de trabajo futuro. Asimismo, se establece la relación del proyecto con las competencias adquiridas durante el grado.

4.1 Consecución de Objetivos

El trabajo ha logrado cumplir satisfactoriamente todos los objetivos planteados, tanto el objetivo general como los cuatro objetivos específicos definidos en la introducción.

Objetivo General

Se ha evaluado exitosamente la aplicabilidad y el rendimiento de herramientas de código abierto basadas en aprendizaje profundo para la segmentación estructural de documentos manuscritos históricos. El enfoque ha combinado efectivamente el análisis técnico de estas herramientas con su aplicación práctica a corpus reales, incorporando tanto métricas cuantitativas como valoraciones cualitativas sobre la utilidad de los resultados obtenidos.

Objetivos Específicos

Objetivo 1: Explorar y analizar herramientas de segmentación de código abierto

Estado: Cumplido completamente.

- Se instalaron y probaron exitosamente Detectron2 y YOLOv8 tanto en entorno local (GPU RTX 1660) como en Google Colab.
- Se solucionaron y documentaron distintos problemas en la fase de instalación.
- Se justificó la selección final de modelos en función de la relación rendimiento/uso de recursos, optando por Detectron2 con Mask R-CNN y YOLOv8l.

Objetivo 2: Desarrollar scripts de preprocesamiento y conversión automática

Estado: Cumplido completamente.

- Se creó un flujo completo de preprocesamiento compuesto por `voc_to_coco_last_update.py`, `unificar_anotaciones_coco.py` y `convertir_coco_a_yolo.py` que transforma automáticamente los datasets OHG y VORAU-253 a los formatos COCO y Ultralytics.
- Se desarrollaron otros scripts para corregir posibles errores que saltan en el momento del entrenamiento si el etiquetado no cumple todos los estándares especificos.
- Se incluyeron unos visualizadores `convertir_anotaciones_coco.py` y `visualizar_segmentaciones_` para validar las conversiones y detectar posibles errores en el proceso.

Objetivo 3: Implementar un sistema automatizado de evaluación

Estado: Cumplido completamente.

- Se desarrolló `evaluate.py`, un sistema que genera métricas estándar (mAP@[.50:.95], AP50, AP75 y AR@k) siguiendo estrictamente la API COCO.
- El sistema es completamente agnóstico al modelo utilizado: acepta outputs JSON tanto de Detectron2 como de YOLOv8.
- Se garantiza la comparabilidad objetiva y reproducible de resultados entre diferentes frameworks y configuraciones.

Objetivo 4: Evaluar la utilidad práctica de los resultados

Estado: Cumplido completamente.

- Se realizó una inspección visual exhaustiva sobre 107 páginas (66 de OHG + 101 de VORAU-253) para evaluar la calidad de las segmentaciones, siendo esta positiva. El buen resultado de esta inspección, sumado a los buenos resultados de las otras métricas, se consideran útiles para futuras tareas, los resultados obtenidos.

4.2 Problemas Encontrados

Durante el desarrollo del trabajo se identificaron y resolvieron varios desafíos técnicos que requirieron soluciones específicas:

Complejidad de la interfaz de Detectron2

- **Problema:** La instalación y compilación de DETECTRON2 resultó especialmente compleja: las versiones precompiladas no eran compatibles con las versiones de CUDA/PyTorch disponibles y, según el equipo, el proceso de *build* fallaba por conflictos de dependencias y flags de compilación.
- **Solución:** Se revisaron y modificaron los ficheros de compilación (`setup.py`, `CMakeLists.txt`), fijando versiones concretas de PyTorch, CUDA y GCC.

Calidad y depuración de las anotaciones

- **Problema:** Los conjuntos de datos incluían errores en las anotaciones (bounding boxes desalineadas, etiquetas incorrectas o ausentes). Estos fallos podían propagarse al entrenamiento y sesgar las métricas.

- **Solución:**
  - Se implementaron *scripts* de verificación automática que detectan incoherencias, las corrigen o marcan para revisión.
  - Se desarrolló un visualizador que superpone las anotaciones originales sobre cada imagen, permitiendo la inspección manual.
  - Tras la depuración, se regeneraron los archivos de anotaciones para asegurar un entrenamiento limpio y métricas fiables.

Selección de arquitecturas comparables entre frameworks

- **Problema:** Para que la comparación fuese rigurosa, era imprescindible emparejar modelos de DETECTRON2 y YOLOV8 con tamaños y complejidad equivalentes. Si se enfrentaban arquitecturas con conteos de parámetros muy dispares, la evaluación quedaba sesgada.
- **Solución:**
  - Se elaboró una matriz de referencia con los modelos disponibles en ambos frameworks (Mask R-CNN R50-FPN, Mask R-CNN X101-FPN; YOLOv8n, s, m, l, x) y se filtraron aquellos cuyo número de parámetros estuviera entre 40 M y 45 M.
  - Finalmente se seleccionaron **Mask R-CNN R50-FPN** ($\approx$ 43M) y **YOLOv8l** ($\approx$ 43 M).
  - Se mantuvieron los hiperparámetros por defecto de cada framework, de modo que las diferencias observadas pudieran atribuirse al diseño de la arquitectura y no al tamaño del modelo.

Falta de un evaluador homogéneo

- **Problema:** Cada framework reporta métricas en formatos diferentes, dificultando la comparación directa.
- **Solución:** Se programó un evaluador propio que garantiza que todos los modelos reporten exactamente los mismos indicadores siguiendo el protocolo COCO.

4.3 Trabajos Futuros

Basándose en los resultados obtenidos y las limitaciones identificadas, se proponen las siguientes líneas de trabajo futuro:

1. **Integración en flujo OCR/HTR completo:** Desarrollar un sistema que integre las áreas segmentadas en un flujo completo de OCR/HTR para digitalizar automáticamente el texto resultante de los documentos históricos.
2. **Difusión y publicación:** Publicar un artículo científico breve y crear una página web con el repositorio completo, incluyendo código, pesos de modelos entrenados y tutoriales detallados para la comunidad investigadora.
3. **Fine-tuning semisupervisado:** Explorar técnicas de pseudo-etiquetado y fine-tuning semisupervisado para reducir la dependencia de anotaciones manuales y facilitar la aplicación a nuevos corpus históricos.

4.4 Relación con el Grado

La realización de este Trabajo Fin de Grado es, en esencia, una puesta en práctica de los saberes acumulados a lo largo de la titulación en Ingeniería Informática. Buena parte de la base teórica procede de la asignatura **Sistemas inteligentes** (11560), donde se presentaron los fundamentos de las redes neuronales y las particularidades de las arquitecturas convolucionales que aquí se explotan para segmentar documentos históricos. Esa base se enriqueció después, en **Aprendizaje automático** (11594), con técnicas de *fine-tuning*, regularización y estrategias de evaluación que han resultado cruciales para interpretar métricas como la *mAP* o el AP_{50}.

El procesamiento de anotaciones —desde la lectura de los ficheros originales hasta su conversión a los formatos COCO y Ultralytics— descansa en los principios básicos vistos en **Estructuras de datos y algoritmos** (11551). El haber trabajado con listas, diccionarios y análisis de complejidad ha permitido escribir scripts claros y lineales ($\mathcal{O}(n)$) que manejan miles de registros sin provocar cuellos de botella apreciables.

Por su parte, la optimización del entrenamiento sobre GPU y la resolución de los habituales conflictos entre versiones de CUDA y PyTorch remiten a **Arquitectura e ingeniería de computadores** (11553). Comprender la jerarquía de memoria y las particularidades del paralelismo masivo ha sido determinante para sacar el máximo partido al hardware disponible.

Finalmente, el proyecto no habría alcanzado un grado de madurez suficiente sin los principios aprendidos en **Ingeniería del software** (11555). La modularización del código, el uso disciplinado de Git y la generación de documentación reproducible son herencias directas de esa asignatura que garantizan la calidad y la longevidad del trabajo.

De forma transversal, este Trabajo Fin de Grado ha contribuido significativamente al desarrollo de competencias clave que complementan los contenidos técnicos adquiridos durante el grado. Una de las más destacadas ha sido la **planificación y gestión de proyectos**, necesaria para estructurar y organizar las distintas fases del trabajo —desde la investigación inicial hasta la evaluación experimental— dentro de un calendario realista y flexible. También se ha reforzado la **capacidad de aprendizaje autónomo**, al abordar herramientas y marcos avanzados que no forman parte explícita del plan docente, como Detectron2 o YOLOv8, cuyo dominio ha requerido la consulta crítica de documentación técnica y la resolución de problemas prácticos. Igualmente, el proyecto ha exigido una **búsqueda rigurosa de información**, combinando fuentes académicas y recursos técnicos actualizados, lo cual ha favorecido una actitud analítica y metódica ante los desafíos del trabajo. Finalmente, se ha potenciado la **comunicación escrita técnica** a través de la elaboración de esta memoria, cuidando la claridad expositiva, la coherencia argumentativa y la precisión terminológica. En conjunto, el TFG ha supuesto una experiencia integradora, donde los conocimientos del grado se han aplicado de forma crítica y contextualizada a un problema complejo y de alto impacto como es la digitalización del patrimonio documental manuscrito.

Bibliografía

[1] R.C. Gonzalez y R.E. Woods. *Digital Image Processing*. Pearson, cuarta edición, 2018.

[2] W.K. Pratt. *Introduction to Digital Image Processing*. CRC Press, 2013.

[3] S.M. Pizer, E.P. Amburn, J.D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J.B. Zimmerman, y K. Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39:3:355–368, septiembre, 1987.

[4] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[5] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1:38–62, enero, 2000.

[6] Y. LeCun, Y. Bengio, y G. Hinton. Deep learning. *Nature*, 521:7553:436–444, mayo, 2015.

[7] I. Goodfellow, Y. Bengio, y A. Courville. *Deep Learning*. MIT Press, 2016.

[8] K. He, G. Gkioxari, P. Dollár, y R. Girshick. Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, págs. 2961–2969, octubre, 2017.

[9] A. Kirillov, K. He, R. Girshick, C. Rother, y P. Dollár. Panoptic segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, págs. 9404–9413, junio, 2019.

[10] S. Mao, A. Rosenfeld, y T. Kanungo. Document structure analysis algorithms: a literature survey. *Proceedings of Document Recognition and Retrieval X*, 5010:197–207, 2003.

[11] Y.Y. Tang, S.-W. Lee, y C.Y. Suen. Automatic document processing: A survey. *Pattern Recognition*, 29:12:1931–1952, diciembre, 1996.

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, y C.L. Zitnick. Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision (ECCV)*, págs. 740–755, septiembre, 2014.

[13] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, y R. Girshick. Detectron2, 2019. Consultado en https://github.com/facebookresearch/detectron2.

[14] H. Schwert. Digging into Detectron 2, 2020. Consultado en https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd.

[15] H. Schwert. Digging into Detectron 2 - Part 2, 2020. Consultado en https://medium.com/@hirotoschwert/digging-into-detectron-2-part-2-dd6e8b0526e.

[16] H. Schwert. Digging into Detectron 2 - Part 3, 2020. Consultado en https://medium.com/@hirotoschwert/digging-into-detectron-2-part-3-1ecc27efc0b2.

[17] H. Schwert. Digging into Detectron 2 - Part 4, 2020. Consultado en https://medium.com/@hirotoschwert/digging-into-detectron-2-part-4-3d1436f91266.

[18] H. Schwert. Digging into Detectron 2 - Part 5, 2020. Consultado en https://medium.com/@hirotoschwert/digging-into-detectron-2-part-5-6e220d762f9.

[19] Z. Shen, K. Zhang, y M. Dell. A Large Dataset of Historical Japanese Documents with Complex Layouts. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, págs. 548–559, junio, 2021.

[20] B. Davis, B. Morse, S. Cohen, B. Price, y C. Tensmeyer. Historical Document Layout Analysis: A Comparison of Modern Methods. *International Conference on Document Analysis and Recognition*, págs. 289–304, agosto, 2022.

[21] X. Yang, E. Yumer, P. Asente, M. Kraley, D. Kifer, y C. Lee Giles. Learning to Extract Semantic Structure from Documents Using Multimodal Fully Convolutional Neural Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, págs. 5315–5324, junio, 2020.

[22] L. Ares, S. Chandna, G. Retsinas, R. Rojas, U. Castellani, S. A. Baghshah y T. Grüning. Overview of AReST 2021: Layout Analysis for Challenging Medieval Manuscripts. *ICDAR Competitions on Analysis of Historical Documents*, págs. 89–107, septiembre, 2021.

[23] X. Li, M. Liu, y J. Wang. Document Layout Understanding Using Few-Shot Object Detection. *Pattern Recognition*, 121:108244, enero, 2022.

[24] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, y M. Zhou. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, págs. 1192–1200, agosto, 2020.

[25] M. Diem, F. Kleber, y S. Fiel. cBAD: ICDAR2017 Competition on Baseline Detection. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, págs. 1355–1360, noviembre, 2017.

[26] Ultralytics. *YOLOv8 Documentation*, 2023. Consultado en https://docs.ultralytics.com/.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, y Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 779–788, junio, 2016.

[28] Joseph Redmon y Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, abril, 2018.

[29] Alexey Bochkovskiy, Chien-Yao Wang, y Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*, abril, 2020.

[30] Chien-Yao Wang, Alexey Bochkovskiy, y Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, julio, 2022.

[31] Chien-Yao Wang, I-Hau Yeh, y Hong-Yuan Mark Liao. YOLOv8: Evolution of YOLO models for real-time object detection. *Computer Vision and Pattern Recognition*, arXiv:2305.09972, mayo, 2023.

[32] Jacob Solawetz y Francesco. What is YOLOv8? A Complete Guide, 23 de octubre de 2024. *Roboflow Blog*. Consultado en https://blog.roboflow.com/what-is-yolov8/.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, y Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, vol. 28, diciembre, 2015.

[34] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, y Alexander C. Berg. SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision (ECCV)*, págs. 21–37, octubre, 2016.

[35] Anand Mishra, Karteek Alahari, y C. V. Jawahar. Scene Text Recognition using Higher Order Language Priors. *British Machine Vision Conference (BMVC)*, septiembre, 2019.

[36] Apostolos Antonacopoulos, Christian Clausner, Christos Papadopoulos, y Stefan Pletschacher. Historical Document Layout Analysis Competition. *International Conference on Document Analysis and Recognition (ICDAR)*, págs. 1516–1520, septiembre, 2011.

[37] Lorenzo Quirós Díaz. Layout Analysis for Handwritten Documents: A Probabilistic Machine Learning Approach. *PhD Thesis*, Universitat Politècnica de València, 2021.

[38] Universidad de Girona. Digitalización y preservación de archivos notariales. Proyecto de colaboración con el Archivo Histórico de Girona, 2022.

[39] M. Diem, F. Kleber, S. Fiel, T. Grüning, y B. Gatos. cBAD: ICDAR Competition on Baseline Detection. *International Conference on Document Analysis and Recognition (ICDAR)*, 2017–2019.

[40] SuperAnnotate. Intersection over Union (IoU) for object detection, 2023. Consultado en https://www.superannotate.com/blog/intersection-over-union-for-object-detection.

[41] H. Zhao, J. Shi, X. Qi, X. Wang, y J. Jia. Pyramid Scene Parsing Network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 2881–2890, julio, 2017.

[42] D. Bolya, C. Zhou, F. Xiao, y Y.J. Lee. YOLACT: Real-time Instance Segmentation. *IEEE International Conference on Computer Vision (ICCV)*, págs. 9157–9166, octubre, 2019.

[43] L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, y A.L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40:4:834–848, abril, 2018.

[44] Xu Zhong, Jianbin Tang, y Antonio Jimeno Yepes. PubLayNet: Largest Dataset Ever for Document Layout Analysis. *arXiv preprint arXiv:1908.07836*, agosto, 2019.

[45] Tom Kocmi, Christian Federmann, Mahmoud El-Haj, Toshiaki Nayak, Petr Šojka, Rob van der Goot y Horacio Saggion. DocLayNet: A large human-annotated dataset for document-layout analysis. *arXiv preprint arXiv:2301.13052*, enero, 2023.

[46] Glenn Jocher, Ayush Chaurasia, Jing Qiu y Alex Stoken. YOLOv8: Ultralytics YOLO models, 2023. Consultado en https://github.com/ultralytics/ultralytics.

[47] NeuralShift. doc-layout-yolov8s model, 2023. Hugging Face repository. Consultado en https://huggingface.co/NeuralShift/doc-layout-yolov8s.

[48] IBM Research. Detectron2 Models and Benchmarks, 2019. Consultado en https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md.

[49] Facebook AI Research. *Detectron2 Model Zoo*, 1 de abril de 2020. Consultado en https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md.

[50] Ultralytics. *YOLOv8 Performance Documentation*, 10 de enero de 2023. Consultado en https://docs.ultralytics.com/models/yolov8/#performance.

[51] L. Quirós Díaz y T. Grüning. VORAU-253: A Medieval Music Manuscript Dataset for Layout Analysis, 2021. *Zenodo Repository*. Consultado en https://zenodo.org/record/5443258.

[52] T. Grüning, R. Labahn, y M. Diem. Layout Analysis on Historical Music Manuscripts: A Case Study on the VORAU-253 Dataset. *International Conference on Document Analysis and Recognition (ICDAR)*, págs. 245–260, septiembre, 2022.

[53] M. Diem, F. Kleber, y S. Fiel. Challenges in Historical Music Manuscript Analysis: The VORAU-253 Perspective. *Journal of Imaging*, 9:4:78, abril, 2023.

APÉNDICE A
OBJETIVOS DE DESARROLLO SOSTENIBLE

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**	X			
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**		X		
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**			X	
ODS 11. **Ciudades y comunidades sostenibles.**		X		
ODS 12. **Producción y consumo responsables.**			X	
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**		X		
ODS 17. **Alianzas para lograr objetivos.**		X		

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Este Trabajo Fin de Grado sobre la segmentación automática y clasificación de la maquetación en manuscritos históricos mantiene una relación significativa con varios Objetivos de Desarrollo Sostenible (ODS), siendo especialmente relevantes aquellos que se han clasificado con un grado de relación alto.

ODS 4: Educación de Calidad

El proyecto contribuye de forma directa y sustancial al ODS 4 al acelerar la digitalización y estructuración de fondos documentales manuscritos. La herramienta desarrollada reduce drásticamente el tiempo necesario para segmentar páginas complejas y genera corpus legibles por máquina, lo que facilita el acceso abierto a fuentes históricas desde cualquier lugar. Investigadores, docentes y estudiantes se benefician de materiales digitalizados, anotados y listos para su explotación, eliminando barreras geográficas y requisitos de conocimientos paleográficos avanzados. Al democratizar el acceso al patrimonio, el trabajo multiplica las oportunidades de aprendizaje y favorece la creación de recursos didácticos interactivos basados en datos reales.

ODS 9: Industria, Innovación e Infraestructuras

La investigación supone una innovación tecnológica destacada en el campo de la visión por computadora aplicada al patrimonio. Al comparar frameworks de última generación (Detectron2 y YOLOv8) y proporcionar un *pipeline* reproducible, se fortalece la infraestructura digital de archivos y bibliotecas. El proyecto no solo optimiza la implementación de modelos de IA en entornos de producción, sino que también fomenta la soberanía tecnológica al basarse en software libre. De este modo, organizaciones con recursos limitados pueden adoptar soluciones punteras sin depender de licencias propietarias, impulsando la modernización de procesos y la consolidación de infraestructuras robustas y escalables.

Relaciones de Nivel Medio

El trabajo presenta relaciones de nivel medio con varios ODS. En el caso del **ODS 8 (Trabajo Decente y Crecimiento Económico)**, la automatización de tareas repetitivas libera a los profesionales para labores de mayor valor, incrementando la productividad y abriendo nuevas oportunidades en la economía digital del patrimonio. Respecto al **ODS 11 (Ciudades y Comunidades Sostenibles)**, la preservación digital de documentación histórica refuerza la identidad cultural y la cohesión social, facilitando además la planificación urbana basada en evidencia. El **ODS 16 (Paz, Justicia e Instituciones Sólidas)** se ve beneficiado de manera indirecta, ya que la sistematización de archivos notariales y judiciales mejora la transparencia y el acceso público a la información. Finalmente, el **ODS 17 (Alianzas para Lograr los Objetivos)** se manifiesta a través de la colaboración entre universidades, archivos y comunidades de software libre que comparten datos, herramientas y buenas prácticas.

Relaciones de Nivel Bajo

Existen vínculos de menor intensidad con otros ODS. El **ODS 10 (Reducción de las Desigualdades)** se aborda indirectamente al proporcionar acceso equitativo a recursos documentales previamente restringidos a instituciones privilegiadas. El **ODS 12 (Producción y Consumo Responsables)** se ve reflejado en la reducción de la manipulación física de documentos frágiles gracias al acceso remoto, prolongando su vida útil. Aunque relevantes, estos impactos son secundarios frente a los objetivos técnicos principales del proyecto.

Conclusión

En conjunto, el TFG demuestra cómo la innovación tecnológica puede alinearse eficazmente con múltiples objetivos de desarrollo sostenible. La herramienta desarrollada no solo representa un avance técnico relevante, sino que contribuye de manera tangible a la construcción de un futuro más inclusivo y eficiente, destacándose especialmente en los ámbitos de la educación (ODS 4) y la innovación (ODS 9). Al mismo tiempo, fortalece la economía digital del patrimonio y promueve la colaboración interinstitucional, sentando las bases para proyectos de mayor escala que continúen impulsando la preservación y democratización del conocimiento histórico.

APÉNDICE B
Glosario

Este apéndice recoge las definiciones de los términos técnicos más relevantes utilizados a lo largo del trabajo, organizados alfabéticamente para facilitar su consulta.

B.1 Glosario de términos

AP (Average Precision): Métrica de detección que resume la curva precisión–recall en un único valor. **AP50** y **AP75** utilizan umbrales de IoU de 0.50 y 0.75, respectivamente.

AR (Average Recall): Promedio del recall obtenido al permitir distintos números máximos de detecciones y umbrales de IoU; refleja cuántos objetos reales localiza el modelo.

Backbone: Parte inicial de una CNN que extrae las características profundas de la imagen; sobre sus salidas se apilan módulos como FPN o RPN.

COCO (Common Objects in Context): Conjunto de datos de referencia en visión por computador que define un protocolo estándar de evaluación (mAP, AR, etc.).

CNN (Convolutional Neural Network): Red neuronal especializada en procesar datos con estructura de rejilla (imágenes); emplea filtros convolucionales para detectar patrones espaciales.

Corpus OHG: *Online Handwritten Girona*. Colección de escrituras notariales manuscritas del Archivo Histórico de Girona, anotada para tareas de segmentación documental.

Corpus VORAU-253: Páginas manuscritas del códice 253 del monasterio austríaco de Vorau, usadas como segundo banco de pruebas.

CUDA: Plataforma y API de NVIDIA que permite ejecutar código en la GPU para acelerar cálculos paralelos, clave en el entrenamiento de redes profundas.

Detectron2: Framework de visión por computador de código abierto (Meta AI) que implementa algoritmos de detección y segmentación de última generación sobre PyTorch.

DLA (Document Layout Analysis): Disciplina que estudia cómo identificar y clasificar las regiones funcionales de un documento (títulos, párrafos, márgenes, etc.).

FPN (Feature Pyramid Network): Módulo que fusiona características de diferentes profundidades del backbone para obtener mapas multiescala, mejorando la detección de objetos de tamaño dispar.

Fine-tuning: Ajuste de un modelo preentrenado a un nuevo dominio con un conjunto de datos relativamente pequeño, acelerando la convergencia y mejorando resultados.

GPU (Graphics Processing Unit): Procesador altamente paralelo diseñado originalmente para gráficos, hoy imprescindible para acelerar el entrenamiento e inferencia de redes neuronales.

HTR (Handwritten Text Recognition): Reconocimiento automático de texto manuscrito; suele aplicarse tras la segmentación estructural de la página.

IoU (Intersection over Union): Razón entre el área de solapamiento y el área de unión de una predicción y su anotación real; base de las métricas AP y AR.

ImageNet: Conjunto de datos masivo de imágenes naturales categorizadas en más de 20.000 clases, utilizado como referencia para el preentrenamiento de redes neuronales convolucionales. Sus características lo convierten en una fuente eficaz para aprender representaciones visuales genéricas transferibles a otras tareas.

LayoutLM: Modelo multimodal que combina texto, imagen y posición para tareas de comprensión de documentos; se menciona como posible mejora futura.

mAP (mean Average Precision): Media de AP calculada sobre múltiples umbrales de IoU (0.50:0.95 en pasos de 0.05); valor global de precisión del modelo.

mIoU (mean Intersection over Union): IoU media sobre todas las clases en segmentación semántica; mide solapamiento a nivel de píxel.

Mask R-CNN: Extensión de Faster R-CNN que añade una rama para predecir máscaras de instancia, combinando detección y segmentación a nivel de píxel.

Preentrenado: Se dice de los pesos de una red que han sido aprendidos previamente en un gran dataset genérico y luego reutilizados en otra tarea.

ROI (Region of Interest): Zona candidata dentro de la imagen que puede contener un objeto; se procesa con más detalle en etapas posteriores.

ROIAlign: Operación que extrae un mapa de características de tamaño fijo para cada ROI, preservando la alineación espacial sin redondeos.

RPN (Region Proposal Network): Subred que, a partir de los mapas FPN, genera miles de propuestas de cajas que probablemente contengan objetos.

Segmentación estructural: División de una página en regiones funcionales (encabezados, texto, firmas, etc.) para facilitar tareas posteriores de OCR o HTR.

SPPF (Spatial Pyramid Pooling Fast): Variante eficiente del módulo SPP que agrega contexto multiescala con bajo coste computacional; presente en YOLOv8.

YOLOv8: Versión más reciente de la familia YOLO, orientada a detección y segmentación en tiempo real; destaca por su rapidez y alto rendimiento.