Detección de Tablas de Calificaciones en Expedientes Académicos mediante Modelo YOLO y Transcripción a Formato CSV

Autor: Álvaro Portugal Muñoz

Director: José Portela González

15 de junio de 2025

Índice

Abstract

Actualmente, la Universidad Pontificia Comillas realiza un proceso de revisión de expedientes académicos manual, proceso que supone una carga administrativa considerable y una inversión significativa de tiempo. Además, al ser una tarea manual repetitiva, está sujeta a errores humanos. Este Trabajo de Fin de Máster propone una solución automatizada basada en el uso de modelos de inteligencia artificial y técnicas de aprendizaje profundo. Para ello, se han entrenado modelos que identifican y extraen tablas de calificaciones de expedientes en formato PDF. La aplicación convierte las páginas de cada documento en imágenes, aplica el modelo de detección a cada imagen del expediente, y recorta las regiones detectadas como tablas para su posterior procesamiento y transcripción a formato CSV. Los resultados obtenidos demuestran el potencial de este enfoque para optimizar y agilizar los procesos de revisión y validación académica, reduciendo significativamente la carga manual.

1. Introducción

Revisar expedientes académicos es una tarea fundamental en los procesos de admisión universitaria, ya que permite verificar que los candidatos cumplen con los requisitos académicos exigidos para acceder a determinadas titulaciones. Sin embargo, en muchas instituciones, incluida la Universidad Pontificia Comillas, este proceso sigue realizándose de forma manual, lo que implica una gran carga de trabajo administrativo, un consumo considerable de tiempo y una mayor probabilidad de cometer errores humanos. A medida que aumenta el volumen de solicitudes, esta situación puede generar cuellos de botella y retrasar los tiempos de respuesta, afectando tanto a los departamentos administrativos como a los propios solicitantes.

En este contexto, surge la necesidad de explorar soluciones tecnológicas que permitan automatizar parcial o totalmente la revisión de estos documentos. Este Trabajo de Fin de Máster, desarrollado en el marco del Máster en Big Data y Analítica Avanzada, aborda esta problemática mediante el desarrollo de un sistema capaz de detectar automáticamente las tablas de calificaciones de los expedientes académicos que entrega un solicitante, habitualmente en formato PDF. La revisión de esta información, que se encuentra estructurada en forma de tabla, es esencial para evaluar el rendimiento académico del candidato. Por este motivo, la identificación automática de dichas tablas representa un paso clave hacia un proceso de revisión más ágil y eficiente.

La solución propuesta se basa en el uso de un modelo de inteligencia artificial, concretamente en técnicas de aprendizaje profundo aplicadas a la detección de objetos en imágenes. Para ello, se ha empleado el modelo YOLOv8 (You Only Look Once), una arquitectura ampliamente utilizada en tareas de detección en imágenes por su equilibrio entre precisión y velocidad de inferencia [1]. El sistema desarrollado convierte cada página de los expedientes PDF en imágenes, aplica el modelo entrenado para localizar las regiones de las imágenes que contienen las tablas de calificaciones, y recorta automáticamente dichas regiones para su posterior tratamiento. Tras este primer paso de detección de tablas, los recortes de las imágenes de las tablas detectadas se envían a un modelo de lenguaje multimodal (GPT-4o) a través de la API de OpenAI. Este modelo identifica el curso académico correspondiente y transcribe automáticamente el contenido de las tablas en un formato estructurado (CSV).

El sistema desarrollado ofrece resultados muy satisfactorios en la detección y transcripción automática de tablas de calificaciones, demostrando un alto nivel de rendimiento en los casos evaluados. El objetivo principal de este proyecto es demostrar la viabilidad de utilizar estos modelos como herramienta integrada en el proceso de revisión académica. Para que el sistema pueda utilizarse de manera efectiva y fiable en un entorno de producción, será necesario perfeccionar su rendimiento mediante un entrenamiento del modelo más exhaustivo. De este modo, se busca que la solución alcance un nivel de fiabilidad y consistencia que le permita integrarse plenamente en los flujos del proceso de admisión.

Este documento está estructurado de la siguiente manera: en primer lugar, se presenta el marco teórico necesario para comprender las tecnologías utilizadas. A continuación, se describe en detalle la

metodología que se ha seguido para preparar los datos, entrenar los modelos y desarrollar el sistema y la aplicación final. Posteriormente, se exponen los resultados obtenidos y se analizan sus implicaciones. Por último, se exponen las conclusiones principales y se plantean posibles líneas de mejora y desarrollo futuro.

2. Marco Teórico

Esta sección introduce los conceptos fundamentales necesarios para comprender la tecnología subyacente al sistema desarrollado. Se revisan brevemente las bases del *Deep Learning* (*DL*) y de la *visión por computador* (*CV*), con especial atención a las *redes neuronales convolucionales* (*CNNs*) y a los algoritmos de detección de objetos. Asimismo, se describe en detalle la arquitectura *YOLO*, empleada como base del sistema desarrollado, y se presentan las principales métricas utilizadas para evaluar el rendimiento de los modelos de detección.

2.1. Deep Learning y Computer Vision

El *Deep Learning* (*DL*) ha transformado el desarrollo de sistemas inteligentes para tareas complejas de percepción y análisis de datos no estructurados. Este área de la inteligencia artificial se centra en la utilización de redes neuronales con múltiples capas para la resolución de tareas específicas, tales como el reconocimiento de imágenes, el procesamiento de lenguaje natural y la detección de objetos, entre muchas otras. A diferencia de los métodos convencionales de aprendizaje automático, en los cuales se requiere la definición manual de las características que el modelo debe analizar, el *DL* posibilita que el propio sistema adquiera de manera automática la capacidad de discernir patrones a partir de los datos de entrada. Estas redes han demostrado una notable capacidad para identificar patrones simples y complejos a medida que se progresa a través de las capas del modelo [2].

Uno de los campos donde esta tecnología ha tenido mayor impacto es en la *visión por computador* (*CV*), disciplina que se encarga de la interpretación de imágenes y vídeos de forma similar a como lo hacen los seres humanos. Tareas como la clasificación de imágenes, el reconocimiento de rostros, la segmentación semántica o la detección de objetos han sido tradicionalmente difíciles de resolver mediante técnicas algorítmicas clásicas. No obstante, con el avance del *Deep Learning*, se han alcanzado niveles de precisión comparables, e incluso superiores, a los de los seres humanos en ciertas tareas [3].

Este progreso ha sido impulsado por varios factores, como la disponibilidad de grandes volúmenes de datos etiquetados, el aumento de la capacidad computacional (especialmente mediante *GPUs*) y el desarrollo de arquitecturas de redes neuronales profundas optimizadas para el procesamiento de información visual. Entre estas arquitecturas, las *redes neuronales convolucionales* (*CNN*) han desempeñado un papel central al explotar las relaciones espaciales entre píxeles en las imágenes, lo que les permite identificar patrones visuales como bordes, texturas o formas complejas. Esta capacidad es clave para el procesamiento eficaz de información visual, como se abordará con mayor detalle en secciones posteriores [4].

En este trabajo, el *Deep Learning* y *Computer Vision* se emplean como herramientas para abordar el problema de la extracción automática de información desde documentos académicos en formato PDF. En particular, se implementan técnicas de detección de objetos fundamentadas en redes profundas para la localización de tablas de calificaciones en imágenes generadas a partir de los expedientes en PDF,automatizando un proceso que actualmente es manual.

2.1.1. Redes neuronales convolucionales (CNN)

Las *CNNs* constituyen una clase de red neuronal profunda, especializada en el procesamiento de datos con estructura espacial como imágenes. En contraste con las *redes neuronales tradicionales total-*

mente conectadas o *red densa* (*fully connected networks*), en las que cada neurona recibe información de todos los píxeles de la imagen, en las CNNs cada neurona se conecta exclusivamente con una pequeña región local de la imagen, denominada campo receptivo (*receptive field*). Esto se debe a que, en una imagen, los píxeles cercanos suelen estar relacionados y formar patrones visuales, como bordes o texturas. Por ejemplo, un borde en una figura aparece cuando hay un cambio local entre diferentes colores o intensidades.

Estas conexiones locales permiten a las CNN detectar patrones simples —como líneas, esquinas o texturas— en las primeras capas. Posteriormente, al combinar múltiples capas, pueden aprender patrones más complejos, como formas o estructuras completas. Además, las CNN usan un pequeño conjunto de filtros que se aplican repetidamente sobre toda la imagen. Este mecanismo se llama peso compartido, y permite al modelo detectar los mismos tipos de patrones en cualquier parte de la imagen. Gracias a esto, las CNN necesitan muchos menos parámetros que una red densa, lo que las hace más rápidas de entrenar y menos propensas a sobreajustar [3].

Una *CNN* está compuesta por una secuencia de capas que transforman progresivamente la entrada (por ejemplo, una imagen) en representaciones de mayor nivel de abstracción. Las capas más importantes son las capas convolucionales (que extraen patrones locales), las funciones de activación (que introducen no linealidad), las capas de agrupamiento (*pooling*, que reducen la dimensión espacial), y, en algunos casos, las capas totalmente conectadas al final de la red (que realizan la clasificación o regresión final)

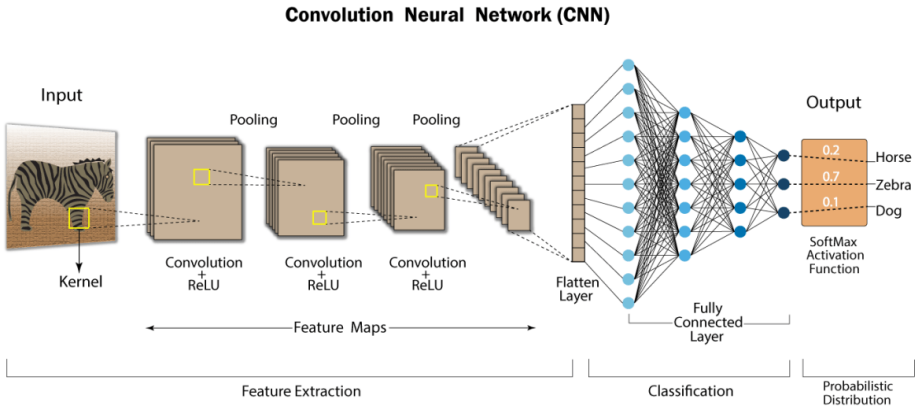


Figura 1: Arquitectura típica de una red neuronal convolucional (CNN). Figura extraida de [5].

Capas convolucionales

La operación convolucional consiste en aplicar un pequeño filtro (también llamado *kernel*) que se desplaza localmente por la imagen de entrada y genera un nuevo mapa de activaciones. Este mapa indica en qué posiciones de la imagen se detectan determinados patrones visuales, como bordes, texturas o esquinas. A medida que se apilan capas convolucionales, la red es capaz de aprender y combinar patrones cada vez más complejos, como formas, estructuras u objetos completos [4]. Este proceso permite construir representaciones jerárquicas de la información visual, fundamentales para tareas como la clasificación o la detección de objetos.

Uno de los principales beneficios de esta operación es la reducción significativa del número de parámetros del modelo, gracias al uso de filtros compartidos. En lugar de aprender parámetros independientes para cada región de la imagen, la CNN aplica el mismo conjunto de filtros a lo largo de toda la imagen. Esto permite entrenar redes mucho más profundas y eficientes, reduciendo el riesgo de sobreajuste al limitar la capacidad del modelo para memorizar el conjunto de entrenamiento. En comparación, las redes densas clásicas (*fully connected networks*) requieren un número de parámetros mucho mayor [6].

Funciones de activación

Después de cada operación convolucional, se aplica una función de activación no lineal. La más común es la ReLU (*Rectified Linear Unit*), definida como

$$f(x) = \max(0, x),$$

donde x es un valor numérico resultado de cada convolución. Esta función transforma el resultado de la convolución de la siguiente manera: todos los valores negativos se sustituyen por 0, mientras que los valores positivos se mantienen sin cambios.

El uso de funciones de activación no lineales es fundamental en las redes neuronales profundas, ya que permite que la red aprenda y represente relaciones complejas entre las variables de entrada y salida. Si no se introdujera esta no linealidad, la red completa sería simplemente una combinación lineal de sus entradas, lo que limitaría enormemente su capacidad para aprender funciones complejas.

Además, la ReLU tiene la ventaja de ser extremadamente simple y eficiente de calcular, lo que no incrementa significativamente la carga computacional. Su uso permite optimizar de manera eficaz el entrenamiento de redes profundas, facilitando el aprendizaje incluso en arquitecturas con muchas capas [3].

Pooling

El *pooling* es una operación que suele aplicarse tras las capas convolucionales, con el objetivo de reducir la resolución espacial de los mapas de activación. De este modo, se disminuye progresivamente el tamaño de las representaciones intermedias a medida que la red se hace más profunda. El tipo más habitual es el *max pooling*, que divide el mapa de activación en regiones locales y selecciona el valor máximo dentro de cada región.

Esta operación ofrece múltiples ventajas: reduce la carga computacional de las capas posteriores, introduce invariancia a pequeñas traslaciones o deformaciones en la imagen, y ayuda a evitar el sobreajuste al forzar a la red a aprender patrones más generales y no dependientes de la posición exacta de los elementos en la imagen [7].

Capas totalmente conectadas y clasificación

En tareas como la clasificación de imágenes, al final de la red se suelen añadir capas totalmente conectadas que toman las representaciones extraídas y las transforman en una salida interpretable (por ejemplo, una probabilidad por cada clase, es decir, un porcentaje). En el caso de la detección de objetos, estas capas suelen sustituirse por cabezales especializados *head* que predicen las coordenadas de los bounding boxes que identifican las coordenadas en la imagen de las clases de objetos, como veremos más adelante con nuestro modelo YOLO.

Aplicación en visión por computador

En este trabajo, las *CNN* permiten procesar imágenes generadas a partir de documentos PDF académicos y detectar visualmente la ubicación de tablas de calificaciones. Gracias a su capacidad para generalizar desde los datos de entrenamiento y adaptarse a distintos formatos visuales, son una tecnología fundamental para el desarrollo del sistema propuesto.

En particular, este proyecto utiliza una arquitectura optimizada de detección de objetos basada en *CNN*, el modelo *YOLO*, que permite realizar detecciones rápidas y precisas incluso en documentos donde el formato y la disposición visual de los elementos varían entre diferentes expedientes.

2.2. Detección de objetos en imágenes

La detección de objetos en imágenes es una tarea clave en visión por computador que consiste en identificar qué objetos aparecen en una imagen y en qué lugar de la imagen se encuentran. Para ello,

cada objeto a detectar se marca con una caja rectangular (bounding box) y se le asigna una etiqueta de clase.

A diferencia de la clasificación de imágenes, que etiqueta la imagen completa en su totalidad, la detección permite localizar varios elementos distintos dentro de la misma imagen, indicando sus posiciones exactas. Esto resulta útil en muchos campos, como la conducción autónoma, la seguridad o, como en este proyecto, para localizar automáticamente tablas de calificaciones en documentos académicos.

Los modelos actuales de detección se basan en arquitecturas de *CNN*, que se entrenan mediante conjuntos de imágenes anotadas. Estas anotaciones incluyen las *bounding boxes* y las clases de los objetos presentes, lo que permite a la red aprender a reconocer y localizar dichos elementos de manera automática [3].

2.2.1. Tipos de algoritmos de detección y representación de objetos

Existen distintos enfoques para resolver la tarea de detección de objetos mediante deep learning. La mayoría de los métodos actuales se basan en redes convolucionales, pero difieren en su arquitectura y en cómo combinan la extracción de características con la predicción de objetos. Podemos clasificar estos algoritmos en dos grandes grupos: algoritmos en dos etapas (two-stage) y algoritmos en una sola etapa (one-stage).

Algoritmos en dos etapas

En los modelos *two-stage*, como *R-CNN* y *Faster R-CNN*, la detección de objetos se realiza en dos fases. En primer lugar, el modelo genera un conjunto de regiones candidatas que probablemente contengan objetos, proceso conocido como *region proposal*. A continuación, estas regiones se analizan en detalle mediante una segunda etapa, que predice tanto la clase de cada objeto como una estimación más precisa de su *bounding box*.

Este enfoque suele ofrecer una mayor precisión en la detección, ya que permite refinar las predicciones en dos pasos. Sin embargo, este beneficio está acompañado de un mayor costo computacional y de tiempos de inferencia más elevados, lo que limita su aplicación en escenarios que requieren procesamiento en tiempo real o en los que no se dispone de grandes recursos computacionales [3].

Algoritmos en una sola etapa

Los modelos *one-stage*, como *YOLO* (*You Only Look Once*) o *SSD* (*Single Shot MultiBox Detector*), realizan la detección de objetos directamente en una sola pasada por la red, sin necesidad de una fase intermedia de propuestas. Estos modelos dividen la imagen en una cuadrícula y, para cada celda, predicen múltiples posibles *bounding boxes* junto con la clase correspondiente para cada una de ellas.

Este enfoque permite obtener modelos mucho más rápidos y computacionalmente más eficientes, lo que los hace especialmente adecuados para aplicaciones que requieren detección en tiempo real [8].

Representación mediante bounding boxes

En todos estos algoritmos, los objetos detectados se representan mediante cajas delimitadoras (*bounding boxes*), que son rectángulos ajustados a cada objeto. Una *bounding box* se define por las coordenadas de sus cuatro esquinas, o bien por las coordenadas del centro de la caja delimitadora y su ancho y alto.

Formatos de anotación

Para que un modelo de detección de objetos aprenda a identificar y localizar elementos dentro de una imagen, es necesario proporcionarle ejemplos correctamente etiquetados. Esto implica que cada imagen de entrenamiento debe ir acompañada de información que indique qué objetos contiene, a qué clase pertenecen y en qué lugar se encuentran. Esa información se almacena en archivos complementarios,

conocidos como formatos de anotación.

En todos los formatos, el elemento esencial es la bounding box: un rectángulo que delimita la región de la imagen donde se encuentra un objeto. Junto con ella, se incluye también la etiqueta de clase correspondiente. Durante el entrenamiento, esta información permite al modelo aprender a realizar predicciones; y durante la validación, se usa para comparar las predicciones con las anotaciones reales y calcular métricas de rendimiento.

Existen diferentes formatos de anotación, como VOC, COCO o YOLO, que varían en su estructura y en la forma de expresar las coordenadas de las cajas delimitadoras. La elección del formato depende normalmente del modelo o framework que se utilice, ya que cada uno espera los datos en un formato concreto.

2.3. La arquitectura YOLO

La arquitectura *YOLO* (*You Only Look Once*) introduce un enfoque innovador en la detección de objetos al tratar esta tarea como un problema de regresión único, a diferencia de los métodos tradicionales basados en múltiples etapas. Su principal aportación consiste en procesar la imagen completa en una sola pasada a través de la red, eliminando así la necesidad de generar previamente regiones de interés, como lo hacen los modelos *two-stage*. De este modo, el modelo predice simultáneamente las coordenadas de las cajas delimitadoras y las probabilidades de clase para cada objeto detectado. Este enfoque permite que *YOLO* alcance una velocidad de inferencia excepcionalmente alta, lo que lo hace especialmente adecuado para aplicaciones en tiempo real [9].

El modelo *YOLO* divide la imagen de entrada en una cuadrícula de $S \times S$ celdas. Para cada celda, predice B posibles *bounding boxes*, cada una con cinco valores: las coordenadas de la caja (x, y, w, h) y una puntuación de confianza que refleja la probabilidad de que la caja contenga efectivamente un objeto y la precisión de su localización. Además, para cada celda se predicen C probabilidades de clase, correspondientes al número de categorías de objeto que puede reconocer el modelo [9]. La salida final del modelo es un tensor de dimensiones $S \times S \times (B \times 5 + C)$.

La red neuronal de *YOLO* emplea una arquitectura convolucional profunda, inspirada en redes como *GoogLeNet*, pero adaptada específicamente para la detección de objetos. Las capas convolucionales extraen características relevantes de la imagen, que luego son procesadas por capas totalmente conectadas para generar las predicciones finales de las cajas delimitadoras y las clases en una sola pasada.

Una característica clave de *YOLO* es el uso de anclajes (*anchors*) [10], o cajas prior en versiones posteriores (a partir de *YOLOv2*). Estos anclajes consisten en un conjunto predefinido de formas y tamaños de cajas que se optimizan durante el entrenamiento, ayudando al modelo a predecir cajas delimitadoras más precisas y a manejar objetos con diferentes relaciones de aspecto. Para cada anclaje, el modelo estima pequeños ajustes (*offsets*) que permiten adaptar la caja final a la forma y tamaño del objeto detectado.

El entrenamiento de *YOLO* se basa en una función de pérdida que combina diferentes tipos de error: localización (precisión de las coordenadas de las cajas), confianza (presencia o ausencia de un objeto en la celda) y clasificación (correcta identificación de la clase del objeto). Para contrarrestar el desequilibrio entre celdas que contienen objetos y las que no, se asigna un peso mayor a los errores cometidos en las celdas que sí contienen objetos, favoreciendo así un aprendizaje más eficaz en escenarios con objetos poco frecuentes.

2.4. Evaluación de modelos de detección

La evaluación del rendimiento de un modelo de detección de objetos es esencial para comprender tanto su capacidad de localizar correctamente los objetos en la imagen como de clasificarlos adecua-

damente. A diferencia de las tareas de clasificación de imágenes, donde solo se predice una etiqueta por imagen, en la detección es igualmente importante evaluar con qué precisión se predicen las posiciones de los objetos (mediante cajas delimitadoras) y si el modelo es capaz de identificar las clases correctamente.

En este contexto, se utilizan métricas específicas que permiten medir el equilibrio entre aciertos y errores, tanto en la localización como en la clasificación. Las métricas más fundamentales en este campo son: *IoU* (Intersection over Union), la precisión, el *recall* y el *mean Average Precision* (*mAP*) [11].

2.4.1. IoU (Intersection over Union)

La *IoU* (*Intersection over Union*) es una métrica fundamental para evaluar el grado de coincidencia entre una caja delimitadora predicha por el modelo y la caja real (*ground truth*) de un objeto. Se calcula dividiendo el área de intersección de ambas cajas entre el área total de su unión:

$$IoU = \frac{\text{Área de Intersección}}{\text{Área de Unión}}$$

El valor de *IoU* varía entre 0 y 1, donde 1 indica una superposición perfecta entre la predicción y la caja real. Esta métrica permite no solo comprobar si el modelo detecta el objeto correcto, sino también evaluar la precisión con la que localiza su posición dentro de la imagen, que es fundamental en este tipo de tareas.

En la práctica, se establece un umbral mínimo de *IoU* (por ejemplo, 0.5 o 0.75) para determinar cuándo una detección se considera correcta. Si la *IoU* entre la caja predicha y la caja real excede este umbral, y la clase es correcta, la detección se contabiliza como un *Verdadero Positivo* (TP). Si no se alcanza el umbral o la clase es incorrecta, se considera un *Falso Positivo* (FP). Por otro lado, si una caja real no es detectada por ninguna predicción válida, se contabiliza como un *Falso Negativo* (FN).

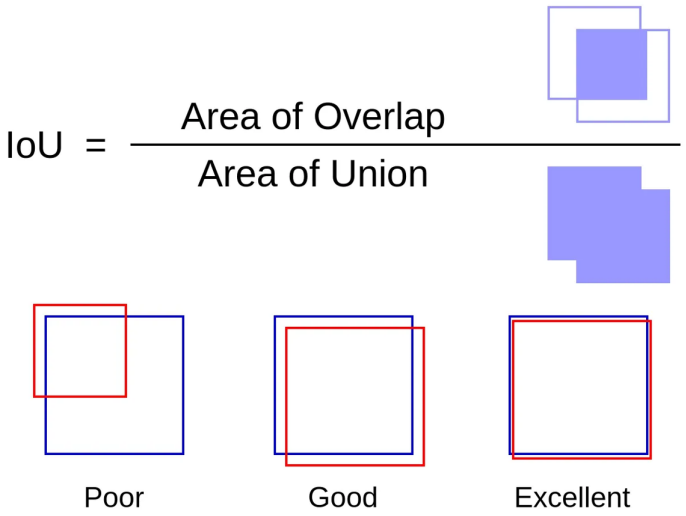


Figura 2: Visualización de la métrica *IoU* (*Intersection over Union*), que mide la superposición entre la caja predicha y la caja real de un objeto. Imagen extraida de [12].

2.4.2. Precisión (Precision)

La *precisión* mide la proporción de detecciones que son correctas. Es decir, indica qué porcentaje de las cajas delimitadoras predichas por el modelo corresponden realmente a una tabla de calificaciones presente en el expediente.

Matemáticamente, se define como el cociente entre el número de verdaderos positivos (detecciones correctas de una tabla de calificación) y la suma de verdaderos positivos y falsos positivos (detecciones erróneas, el modelo predice una tabla de calificaciones donde no la hay):

$$\text{Precisión} = \frac{\text{Verdaderos Positivos (TP)}}{\text{Verdaderos Positivos (TP)} + \text{Falsos Positivos (FP)}}$$

Una alta precisión indica que el modelo comete pocos errores a la hora de predecir la presencia de tablas (es decir, genera pocos falsos positivos).

2.4.3. Recall (Exhaustividad)

El *recall*, también conocido como *exhaustividad*, mide la proporción de objetos reales presentes en la imagen que han sido correctamente detectados por el modelo. Es decir, indica la capacidad del sistema para identificar todos los objetos que realmente existen en la imagen.

En el contexto de este proyecto, un alto *recall* implica que el sistema es capaz de localizar la mayoría de las tablas de calificaciones presentes en los documentos, minimizando el número de tablas que pasan desapercibidas.

Matemáticamente, se define como el cociente entre el número de verdaderos positivos (tablas correctamente detectadas) y la suma de verdaderos positivos y falsos negativos (tablas reales que el modelo no ha detectado):

$$\text{Recall} = \frac{\text{Verdaderos Positivos (TP)}}{\text{Verdaderos Positivos (TP)} + \text{Falsos Negativos (FN)}}$$

Un alto *recall* indica que el modelo consigue detectar la mayoría de los objetos presentes (pocos falsos negativos).

Existe un equilibrio natural entre *precisión* y *recall*: al aumentar la *precisión* (siendo más estricto al aceptar detecciones), el *recall* puede disminuir (dejando sin detectar algunos objetos reales), y viceversa.

2.4.4. mAP (mean Average Precision)

El *mean Average Precision* (mAP) es la métrica más utilizada y robusta para evaluar el rendimiento global de los modelos de detección de objetos. Ofrece una visión completa de la capacidad del modelo para localizar correctamente los objetos y clasificarlos.

Para cada clase, se construye una *Curva Precisión-Recall*, variando el umbral de confianza de las predicciones del modelo. A partir de esta curva, se calcula el *Área Bajo la Curva* (AP, por *Average Precision*), que resume en un solo valor el equilibrio entre precisión y recall a lo largo de todos los umbrales.

El mAP se obtiene promediando las AP de todas las clases detectadas:

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

donde N es el número total de clases.

Es habitual reportar el *mAP* a diferentes umbrales de IoU. Por ejemplo, *mAP@0.5* (donde se requiere un IoU $\geq$ 0,5 para considerar una detección como *True Positive*), o *mAP@[0.5:0.95]*, que promedia el *mAP* calculado a lo largo de varios umbrales de IoU, desde 0.5 hasta 0.95 en incrementos de 0.05. Esta última variante es más estricta y proporciona una evaluación más completa del rendimiento del modelo en condiciones diversas de localización.

El *mAP* es especialmente útil en proyectos como este, ya que permite cuantificar de forma global tanto la capacidad del sistema para localizar correctamente las tablas de calificaciones como su precisión en la clasificación.

3. Metodología

3.1. Visión general del sistema

El sistema desarrollado en este trabajo tiene como objetivo automatizar la detección y digitalización de tablas de calificaciones contenidas en documentos académicos en PDF. Para ello, se ha diseñado un flujo de trabajo estructurado en tres fases principales que se desarrollan de forma secuencial.

En la **primera fase**, se realiza la preparación de los datos necesarios para entrenar el modelo YOLO de detección de tablas de calificaciones. Este entrenamiento se realiza usando expedientes académicos reales de alumnos. Esta primera fase incluye la separación aleatoria de los expedientes académicos en conjuntos de entrenamiento (80 %) y validación (20 %), la conversión de los documentos en imágenes, la anotación y clasificación manual de las tablas de calificaciones en el conjunto de entrenamiento, y el entrenamiento de varios modelos basados en la arquitectura *YOLOv8* con diferentes hiperparámetros.

En la **segunda fase**, se evalúa el rendimiento de los modelos sobre el conjunto de validación y se escoge el mejor de ellos para la integración con la aplicación final. Este modelo se vuelve a evaluar sobre nuevos expedientes no vistos anteriormente para asegurarnos de su correcto funcionamiento. Se crea un script que carga el modelo con mejor rendimiento, se le pasa un expediente en formato PDF en su versión original y se detectan automáticamente las tablas de calificaciones presentes en cada página. El script devuelve la imagen (o imágenes en caso de que haya más de una) de la tabla con la bounding box en rojo y un recorte de la bounding box (tabla de calificaciones).

La **tercera fase** consiste en la transcripción automática del contenido de las tablas detectadas. Antes de realizar dicha transcripción, el sistema identifica automáticamente el curso académico al que pertenece cada tabla, ya que esta información es imprescindible para interpretar correctamente su estructura y contenido. A partir de las imágenes recortadas y del curso identificado, un componente adicional del sistema es capaz de interpretar visualmente las notas y generar una tabla estructurada en formato digital.

Estas tres fases se integran finalmente en una aplicación interactiva que permite al usuario cargar un documento PDF y ejecutar de forma automática todo el proceso: detección de tablas, identificación del curso académico correspondiente y transcripción de las calificaciones. El sistema procesa todas las tablas encontradas en el documento sin requerir intervención manual adicional por parte del usuario.

3.2. Fase 1: Entrenamiento del modelo de detección con YOLOv8

3.2.1. Recolección y división del conjunto de datos

El entrenamiento de un modelo de detección de objetos requiere un conjunto de datos previamente anotados. Para garantizar una evaluación objetiva del rendimiento del modelo, es fundamental dividir este conjunto de datos en dos subconjuntos: uno para el entrenamiento y otro para la validación.

El conjunto de entrenamiento se utiliza para ajustar los parámetros del modelo, permitiéndole aprender a identificar las regiones correspondientes a las tablas de calificaciones. Por su parte, el conjunto de validación permite evaluar la capacidad del modelo para generalizar a datos no vistos, y es especialmente útil para detectar problemas de sobreajuste o insuficiencia de datos.

Los modelos se han entrenado con 108 expedientes de alumnos. La partición se ha realizado de forma aleatoria, utilizando un 80 % de los documentos PDF para el entrenamiento y el 20 % restante para la validación. Se decidió emplear esta proporción con el objetivo de disponer de una cantidad suficiente de datos para el aprendizaje del modelo, pero al mismo tiempo disponer de un conjunto representativo e independiente para evaluar su rendimiento.

3.2.2. Conversión de documentos PDF a imágenes

Dado que los modelos de detección de objetos operan sobre imágenes y no directamente sobre archivos PDF, el primer paso del procesamiento consiste en convertir cada página de los documentos académicos a un formato de imagen. Esta transformación permite aplicar los modelos sobre los datos originales.

Para llevar a cabo esta conversión se ha utilizado la biblioteca *PyMuPDF* (*fitz*), que permite renderizar cada página de un archivo PDF en una imagen. En concreto, para cada página se genera una imagen en formato RGB, aumentando la resolución mediante un factor de escala configurable (zoom), con el objetivo de preservar la legibilidad de las tablas y maximizar la calidad de detección.

El resultado de este proceso es un conjunto de imágenes en formato estándar (PNG), cada una correspondiente a una página de un documento académico. Estas imágenes son posteriormente empleadas tanto en la fase de anotación manual como en el entrenamiento y evaluación del modelo de detección.

3.2.3. Anotación manual de tablas de calificaciones

Una vez convertidas las páginas de los documentos a imágenes, es necesario indicar manualmente la ubicación exacta de las tablas de calificaciones que el modelo deberá aprender a detectar. Este proceso se conoce como *anotación*, y es fundamental en cualquier enfoque de aprendizaje supervisado aplicado a la detección de objetos.

La anotación consiste en dibujar, sobre cada imagen del conjunto de entrenamiento, una o varias cajas delimitadoras (*bounding boxes*) que encierren las tablas de calificaciones relevantes. Además de las coordenadas de cada caja, se asigna a cada una de ellas la etiqueta de clase que identifica el tipo de objeto contenido. En este caso, dado que solo se detecta un tipo de objeto (la tabla de calificaciones), todas las anotaciones pertenecen a una única clase.

Para facilitar esta tarea se ha utilizado la herramienta *LabelImg*, una aplicación de etiquetado visual ampliamente utilizada en proyectos de visión por computador. Esta herramienta permite seleccionar interactivamente las regiones de interés sobre cada imagen y genera automáticamente los archivos de anotación correspondientes. Las anotaciones se guardan en formato XML siguiendo el estándar Pascal VOC, lo que asegura su compatibilidad con los procesos de conversión y entrenamiento utilizados en

fases posteriores.

Este paso, aunque manual, es crítico para garantizar que el modelo reciba ejemplos precisos y representativos durante el entrenamiento. La calidad y coherencia de estas anotaciones influyen directamente en el rendimiento del sistema.

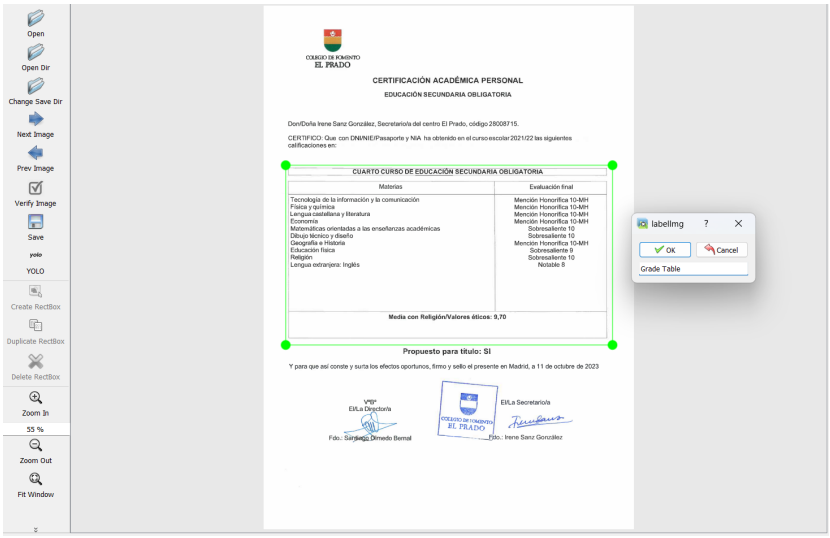


Figura 3: Ejemplo del proceso de anotación manual de una tabla de calificaciones. Se ha realizado este procedimiento para cada tabla presente en las imágenes del conjunto de entrenamiento, utilizando la herramienta *LabelImg*.

3.2.4. Entrenamiento y ajuste de hiperparámetros del modelo YOLOv8

El entrenamiento del modelo se integra en un proceso experimental diseñado para identificar la combinación óptima de hiperparámetros, con el objetivo de maximizar la precisión en la detección de tablas de calificaciones manteniendo una eficiencia computacional razonable.

Este proceso de búsqueda de hiperparámetros óptimos se organiza en dos fases. En la primera, se fijan la tasa de aprendizaje inicial (*lr0*) y el tamaño de las imágenes de entrada (*imgsz*). La tasa de aprendizaje define la velocidad con la que el modelo ajusta sus pesos durante el entrenamiento, mientras que el tamaño de imagen determina la resolución con la que se procesan los datos visuales. A partir de estos valores iniciales, se varían el número de épocas (*epochs*), que indica cuántas veces el modelo recorre completamente el conjunto de entrenamiento, y el tamaño del lote (*batch*), que especifica cuántas imágenes se procesan simultáneamente en cada iteración.

En la segunda fase, se toman los valores óptimos obtenidos para *epochs* y *batch*, y se experimenta con distintas combinaciones de *lr0* e *imgsz*. Esta etapa permite afinar el modelo y mejorar su capacidad de generalización, especialmente ante imágenes con estructuras o calidades variables.

Durante cada experimento, se entrena un modelo basado en la arquitectura *YOLOv8*, una versión ligera optimizada para velocidad y eficiencia. En cada iteración se monitorizan métricas clave como la pérdida total, la precisión y el *mAP* (*mean Average Precision*), lo que permite evaluar el progreso del modelo y detectar posibles casos de sobreajuste.

3.2.5. Evaluación con métricas de detección

Una vez completado el entrenamiento de los distintos modelos, se procede a su evaluación sobre un conjunto de validación, compuesto por documentos que no han sido utilizados durante la fase de

entrenamiento. Esta evaluación permite medir la capacidad del sistema para generalizar a nuevos datos y detectar con precisión las tablas de calificaciones en documentos no vistos.

Para ello, se emplean métricas estándar en tareas de detección de objetos, que permiten evaluar tanto la calidad de la localización como la precisión de la clasificación:

- *Precisión (Precision)*: proporción de predicciones correctas entre todas las predicciones realizadas. Indica la capacidad del modelo para evitar falsos positivos, es decir el modelo predice una tabla donde no la hay.
- *Exhaustividad (Recall)*: proporción de objetos reales que han sido correctamente detectados. Mide la capacidad del modelo para no omitir objetos relevantes (minimizar falsos negativos).
- *Precisión media (mAP)*: valor promedio del área bajo la curva Precisión–Recall para todos los umbrales de confianza. Es la métrica más utilizada para evaluar el rendimiento global del modelo en detección.
- *Pérdida de localización*: error en la predicción de las coordenadas de las cajas delimitadoras, que evalúa la precisión de las posiciones de los objetos detectados.

Estas métricas permiten comparar objetivamente el rendimiento de las distintas configuraciones evaluadas y proporcionan una guía clara para seleccionar el mejor modelo. Además, permiten identificar posibles áreas de mejora, como errores sistemáticos en la localización o una elevada tasa de falsas detecciones. Los resultados detallados de estas métricas se presentan y analizan en profundidad en el apartado de *Resultados*.

Una vez completada la evaluación, se selecciona como modelo final aquel que alcanza el mejor equilibrio entre *precisión*, *exhaustividad* y *mAP* sobre el conjunto de validación. Los pesos del modelo seleccionado se guardan en un archivo que será utilizado en las fases posteriores de detección sobre documentos una vez el sistema se despliega.

3.3. Fase 2: Procesamiento automático de expedientes con IA generativa

3.3.1. Aplicación del modelo YOLOv8 y extracción de regiones detectadas

Una vez completado el entrenamiento, el modelo *YOLOv8* se guarda en un archivo con los pesos aprendidos. En la fase de predicción, este modelo se aplica automáticamente sobre documentos PDF no utilizados durante el entrenamiento ni la validación.

Para cada documento, el sistema convierte sus páginas en imágenes, que se introducen en el modelo previamente entrenado. *YOLOv8* detecta las tablas de calificaciones presentes en cada página, devolviendo para cada una una *bounding box* con sus coordenadas, una puntuación de confianza y la clase correspondiente.

A partir de estas predicciones, se generan dos salidas por tabla detectada:

- Una versión de la imagen original con las cajas delimitadoras dibujadas, que permite una verificación visual rápida.
- Una imagen recortada que contiene únicamente la región de la tabla detectada, que será utilizada como entrada para la fase de transcripción automática.

Ambos tipos de imágenes se almacenan organizadas por documento y página, permitiendo trazabilidad, inspección manual e integración con el sistema de extracción textual. Este paso intermedio es esencial para aislar visualmente las tablas y facilitar su procesamiento posterior mediante IA generativa.

3.3.2. Identificación del curso académico mediante un *prompt* a GPT

Para que el sistema pueda transcribir correctamente una tabla de calificaciones, es imprescindible conocer de antemano el curso académico al que pertenece dicha tabla. Esto se debe a que la estructura, el tipo de asignaturas y el formato de las notas varían significativamente entre cursos.

La identificación del curso se realiza mediante un modelo de lenguaje (*GPT-4o*), utilizando como entrada una imagen de la página completa del expediente. En dicha imagen se ha detectado una tabla concreta haciendo uso de nuestro modelo, marcada visualmente con un rectángulo rojo, y es únicamente sobre esta tabla sobre la que el modelo debe centrar su análisis.

El proceso sigue los siguientes pasos:

1. Se codifica la imagen completa de la página (en formato base64) para su envío a la API de OpenAI.
2. Se construye un *prompt* claro y restrictivo, en el que se indica explícitamente que solo debe analizarse la tabla marcada con el rectángulo rojo.
3. Se interroga al modelo con esta imagen y el *prompt*, solicitándole que devuelva uno de los siguientes cursos válidos:
   - Tercero ESO
   - Cuarto ESO
   - Primero Bachillerato
4. Si el modelo identifica con claridad el curso académico, devuelve su nombre exacto. En caso contrario, responde con el valor literal *No identificado*.

La salida del modelo se procesa automáticamente para mapearla al formato esperado y se guarda en un archivo de texto (*_curso.txt*) asociado a la imagen. Este archivo actúa como condición imprescindible para transcribir la tabla en pasos siguientes: si no existe o contiene *No identificado*, la transcripción de la tabla se cancela automáticamente.

Este enfoque asegura que solo se transcriban aquellas tablas cuyo curso académico ha sido identificado con claridad. Se da prioridad a la precisión, evitando errores que puedan surgir si se intenta transcribir una tabla sin saber a qué curso pertenece.

3.3.3. Transcripción automática de notas usando GPT

La transcripción del contenido de cada tabla detectada se realiza mediante un modelo de lenguaje multimodal (*GPT-4o*), que recibe como entrada una imagen recortada de la tabla y el curso académico previamente identificado. Solo se procesan aquellas tablas para las que se haya podido determinar el curso, ya que esta información es imprescindible para interpretar correctamente las asignaturas y su clasificación.

El objetivo de este paso es convertir una tabla visual en una representación estructurada en formato digital (CSV), organizada por materias, categorías y calificaciones. El proceso que se ha seguido es el siguiente:

1. Se comprueba si existe un archivo *_curso.txt* asociado a la imagen de la tabla y generado en el paso previo. Si el archivo no existe o contiene el valor *No identificado*, la transcripción se para automáticamente. De lo contrario, se procede a la conversión.

2. La imagen se codifica en base64 y se envía al modelo junto con el curso detectado usando la API de *openAI*. El sistema le pasa un prompt claro que indica al modelo que debe generar un archivo CSV utilizando el carácter *;* como separador, con las siguientes tres columnas:
   - *Materia*: nombre de la asignatura.
   - *Categoría*: categoría de evaluación (por ejemplo, Evaluación continua, Final, etc.), elegida de un conjunto cerrado definido por curso.
   - *Nota*: calificación numérica entre 0 y 10.
3. Las posibles categorías se incluyen en el prompt, ajustadas a cada curso (Tercero ESO, Cuarto ESO o Primero de Bachillerato). Si el modelo no puede asignar correctamente una categoría, se utiliza el valor *Desconocido*.
4. La respuesta generada se limpia de posibles marcas de formato (por ejemplo, delimitadores de código en Markdown) y se intenta cargar como un CSV. Si el archivo es válido, se guarda con el sufijo *_transcripcion.csv*. En caso de error de formato, el contenido se almacena como *_error.txt* para su revisión posterior.

Este proceso permite transformar automáticamente imágenes en tablas digitales estructuradas, facilitando la extracción masiva de datos académicos sin necesidad de intervención humana directa.

3.3.4. Manejo de errores e incertidumbre en la transcripción

El proceso de transcripción automática conlleva ciertos riesgos de error, tanto por la calidad visual de las imágenes como por ambigüedades presentes en el formato de las tablas. Para mitigar estos riesgos, el sistema incorpora medidas específicas de control de errores y gestión de incertidumbre.

En primer lugar, se adopta una política conservadora: solo se transcriben aquellas tablas para las que se ha identificado previamente el curso académico. Esto garantiza que el modelo de lenguaje cuente con el contexto adecuado para clasificar correctamente las asignaturas en las categorías definidas.

Además, una vez generada la transcripción en formato CSV, el sistema valida que el contenido sea correcto y legible. Para ello, se intenta cargar el texto resultante como un archivo CSV estructurado. Si este paso falla (por errores de formato, separadores incorrectos, columnas incompletas o inconsistencias en los datos), se considera que la transcripción no es válida.

Este enfoque permite alcanzar un equilibrio entre automatización y fiabilidad, priorizando la calidad de los resultados y minimizando el riesgo de generar datos incorrectos o incompletos.

3.4. Fase 3: Desarrollo de la aplicación interactiva

El objetivo de esta fase es construir una aplicación accesible y funcional que permita al usuario procesar expedientes académicos de forma completamente automática. La interfaz ofrece un entorno sencillo para cargar documentos en formato PDF, realizar la transcripción y visualizar los resultados obtenidos.

3.4.1. Diseño de la interfaz con Streamlit

La interfaz de usuario del sistema se desarrolla utilizando la biblioteca *Streamlit*, que permite construir aplicaciones web interactivas de forma sencilla y rápida a partir de scripts en Python. Esta interfaz sirve como punto de entrada para el usuario y como visor del resultado del procesamiento.

El diseño se centra en la simplicidad y la claridad. Al abrir la aplicación, el usuario puede seleccionar un archivo PDF desde su sistema local a través de un panel de carga. Una vez seleccionado, la aplicación lanza automáticamente el proceso completo:

- Detección de tablas,
- Identificación del curso académico,
- Transcripción de calificaciones, y
- Generación de resultados en formato estructurado.

Después del procesamiento, la interfaz muestra al usuario los elementos más relevantes:

- Las páginas analizadas,
- Las tablas detectadas y marcadas visualmente con su *bounding box*,
- Las imágenes recortadas correspondientes a cada tabla, y
- El contenido transcrito en formato tabular.

Esta presentación clara y estructurada permite validar fácilmente los resultados y detectar posibles errores o inconsistencias. Además, la aplicación incluye mensajes informativos en caso de que alguna tabla no pueda ser transcrita, por ejemplo, si no se logra identificar el curso académico.

El uso de *Streamlit* garantiza que el sistema sea accesible para usuarios sin conocimientos técnicos, permitiendo ejecutar todo el pipeline sin necesidad de trabajar directamente con código.

3.4.2. Exportación organizada de resultados por documento

Tras completar el procesamiento de los documentos PDF, el sistema genera una estructura de salida organizada que facilita la gestión y el análisis posterior de los resultados. Para cada documento procesado, se crea un conjunto de archivos asociados que se almacenan de forma estructurada en el sistema de archivos.

Los elementos exportados incluyen:

- Las imágenes originales con las tablas detectadas marcadas mediante *bounding boxes*.
- Los recortes individuales de cada tabla detectada.
- Un archivo *.txt* con el curso académico identificado para cada tabla.
- Un archivo *.csv* con la transcripción estructurada de cada tabla, cuando el curso ha sido identificado con claridad.
- Un archivo de error alternativo (*_error.txt*), en caso de que el modelo no haya podido generar una transcripción válida.

Estos archivos se nombran sistemáticamente a partir del nombre del documento y del índice de la tabla, lo que permite mantener la trazabilidad entre las distintas etapas del procesamiento. Además, la organización en carpetas por documento facilita su posterior revisión, exportación o integración con otros sistemas.

Esta estrategia de exportación clara y estructurada permite al usuario acceder fácilmente a los resultados.

4. Resultados

En esta sección se presentan los resultados obtenidos durante el desarrollo y validación del sistema propuesto para la detección automática de tablas de calificaciones en documentos académicos. Dado el carácter aplicado del proyecto, los resultados se dividen en dos bloques complementarios.

En primer lugar, se exponen los resultados cuantitativos del entrenamiento de los modelos *YOLOv8*, incluyendo las métricas de *precisión*, *recall* y *precisión media* (*mAP*) evaluadas sobre el conjunto de validación.

Posteriormente, se presentan los resultados cualitativos obtenidos al aplicar el sistema completo —detección, identificación del curso académico y transcripción— sobre un conjunto de documentos reales no utilizados durante el entrenamiento ni la validación.

4.1. Resultados del entrenamiento del modelo YOLOv8

El proceso de entrenamiento del modelo YOLOv8 se llevó a cabo en dos fases experimentales. En la Fase 1, se mantuvieron constantes la tasa de aprendizaje ($lr0 = 0.01$) y el tamaño de imagen de entrada ($imgsz = 640$ px), mientras se variaban el tamaño del *batch* y el número de épocas de entrenamiento. El objetivo de esta fase era determinar la configuración óptima en términos de número de épocas y tamaño de batch para maximizar el rendimiento del modelo.

Posteriormente, en la Fase 2, se tomó el mejor modelo obtenido en la Fase 1 (M7: 200 *epochs*, *batch size* 16) y se exploraron distintas combinaciones de tasa de aprendizaje y tamaño de imagen para evaluar su impacto sobre la precisión y el mAP.

En ambas fases se monitorizaron las métricas estándar de evaluación para tareas de detección de objetos: *precisión* (*Precision*), *exhaustividad* (*Recall*), *mAP@0.5* y *mAP@0.5:0.95*.

Los resultados completos del entrenamiento se resumen en la tabla 1.

Modelo	Epochs	Batch	Learning Rate	Imgsz	Precision	Recall	mAP@50	mAP@50-95
Fase 1: lr0 = 0.01, imgsz = 640								
M1	50	4	0.01	640	0.997	0.976	0.989	0.915
M2	100	4	0.01	640	0.999	0.976	0.994	0.916
M3	100	8	0.01	640	0.999	1.000	0.995	0.923
M4	150	8	0.01	640	0.995	0.976	0.991	0.928
M5	100	16	0.01	640	0.999	0.976	0.990	0.931
M6	150	16	0.01	640	0.999	0.976	0.992	0.937
M7	200	16	0.01	640	1.000	0.999	0.995	0.940
M8	262	16	0.01	640	0.999	0.976	0.989	0.924
Fase 2: Variación de Learning Rate								
M9	200	16	0.01	640	1.000	0.999	0.995	0.940
M10	200	16	0.02	640	1.000	0.999	0.995	0.933
M11	200	16	0.005	640	1.000	0.999	0.995	0.933

Tabla 1: Resultados del entrenamiento de los modelos YOLOv8.

El análisis de los resultados muestra que el modelo *YOLOv8* alcanza métricas de rendimiento consistentemente elevadas en todas las configuraciones probadas. En la Fase 1 se observa una tendencia clara de mejora a medida que se incrementan el número de *epochs* y el tamaño del *batch*, destacando el modelo *M7* (200 épocas, *batch size* 16) como el mejor de esta fase, con una *precision* perfecta (1.000), un *recall* de 0.999, un *mAP@0.5* de 0.995 y un *mAP@0.5:0.95* de 0.940.

En la Fase 2 se evaluaron variaciones en la tasa de aprendizaje, manteniendo la configuración de *M7*. Los resultados indican que el valor inicial de $lr0 = 0.01$ seguía siendo óptimo, ya que las modificaciones

exploradas (0.02 y 0.005) no lograron mejorar el rendimiento del modelo. De hecho, se observó una ligera disminución en el *mAP@0.5:0.95* en ambos casos.

Además, en esta segunda fase el coste computacional se volvió notablemente alto. Considerando que el modelo ya ofrecía un rendimiento muy elevado y estable en todas las métricas evaluadas, se decidió no continuar explorando nuevas combinaciones de hiperparámetros y adoptar el modelo *M7* como modelo final para su integración en el sistema.

En conjunto, los resultados confirman que la configuración seleccionada proporciona un modelo robusto y fiable para la detección de tablas de calificaciones, permitiendo pasar con seguridad a la fase de aplicación sobre documentos reales.

4.1.1. Análisis detallado del modelo seleccionado (M7)

El modelo *M7*, entrenado durante 200 épocas con un *batch size* de 16 y una tasa de aprendizaje *lr0* = 0.01, fue seleccionado como modelo final tras los experimentos realizados. A continuación se analizan las principales métricas obtenidas durante su proceso de entrenamiento.

La Figura 4 muestra la evolución de las pérdidas y métricas clave a lo largo del entrenamiento. Se observa una rápida convergencia de las diferentes componentes de la pérdida (*box loss*, *classification loss* y *distribution focal loss*), junto con una mejora progresiva y estable de las métricas de *precision* y *recall*, que alcanzan valores cercanos a 1.0 en las últimas fases del entrenamiento. Asimismo, las métricas *mAP@0.5* y *mAP@0.5:0.95* alcanzan valores finales de 0.995 y 0.940, respectivamente.

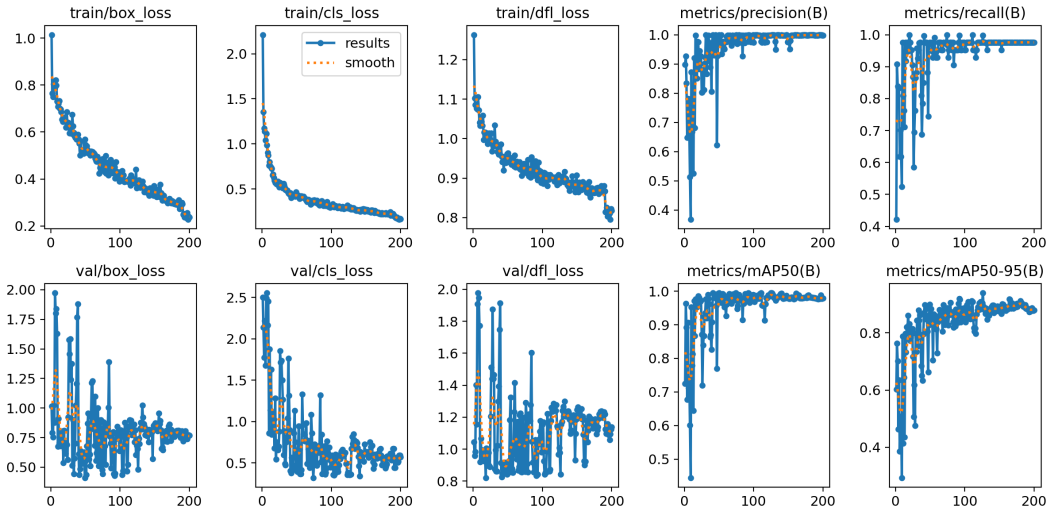


Figura 4: Evolución de las pérdidas y métricas de evaluación durante el entrenamiento del modelo *M7*.

El comportamiento del modelo en términos de la relación *precision-recall* se muestra en la Figura 5. Se aprecia que el modelo mantiene valores de *precision* cercanos a 1.0 a lo largo de prácticamente todo el rango de *recall*, lo que evidencia un excelente equilibrio entre ambas métricas.

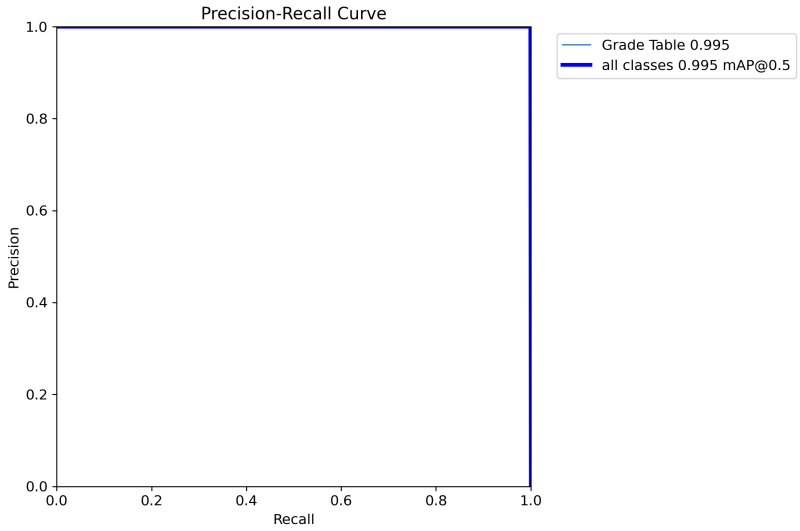
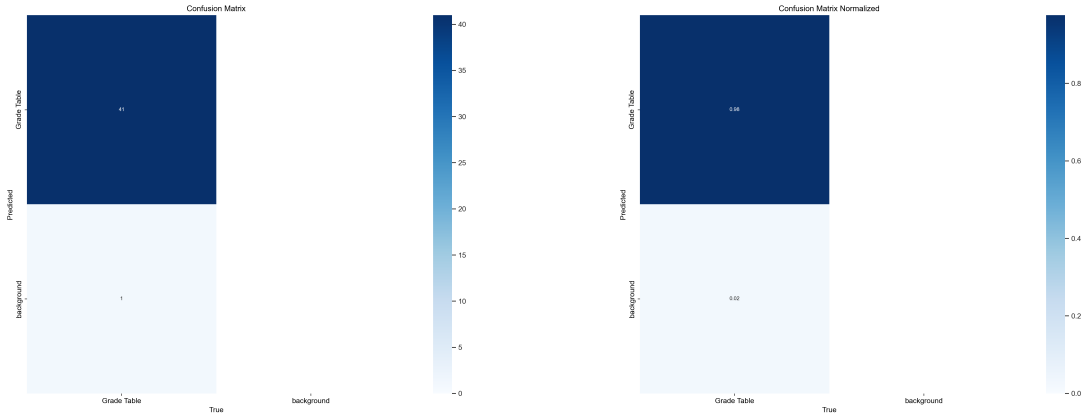


Figura 5: Curva *precision-recall* del modelo *M7*.

Por otro lado, la Figura 6 muestra las matrices de confusión del modelo, tanto en valores absolutos como normalizados. Los resultados confirman la capacidad del sistema para distinguir con gran precisión las tablas de calificaciones del resto del contenido del documento. La tasa de verdaderos positivos para la clase *Grade Table* alcanza el 98 %, con un porcentaje muy reducido de falsos positivos.



(a) Matriz de confusión (valores absolutos)

(b) Matriz de confusión normalizada

Figura 6: Matrices de confusión del modelo *M7*.

Finalmente, la curva F1-Confidence, mostrada en la Figura 7, evidencia que el modelo mantiene un valor máximo de F1-score próximo a 1.0 para un amplio rango de valores de confianza. Esto permite aplicar umbrales de detección conservadores sin pérdida de rendimiento, lo que resulta especialmente relevante en aplicaciones prácticas donde se desea minimizar la aparición de falsas detecciones.

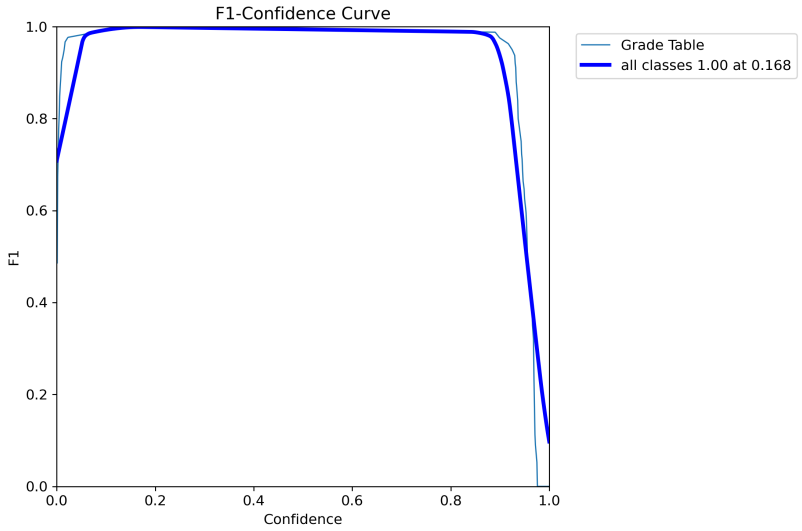


Figura 7: Curva F1-Confidence del modelo M7.

En conjunto, los resultados obtenidos confirman que el modelo M7 presenta un comportamiento altamente robusto, sin evidencias de sobreajuste y con un rendimiento excelente tanto en la localización como en la clasificación de las tablas de calificaciones. Estas evidencias justifican su selección como modelo final para su integración en el sistema.

4.1.2. Ejemplos de predicciones en el conjunto de validación

Para complementar el análisis del modelo M7, en esta sección se presentan ejemplos visuales que permiten evaluar gráficamente el comportamiento del modelo sobre el conjunto de validación.

Las Figuras 8 y 9 muestran una comparación entre las etiquetas reales (*ground truth*) y las predicciones obtenidas por el modelo. La Figura 8 representa las posiciones anotadas manualmente de las tablas de calificaciones en las imágenes del conjunto de validación, es decir, las *bounding boxes* que el modelo debería detectar. Por su parte, la Figura 9 muestra las predicciones generadas por el modelo sobre esas mismas imágenes, indicando para cada caja predicha tanto su localización como el nivel de confianza asignado.

La comparación de ambas imágenes permite apreciar visualmente el grado de acierto del modelo. En este caso, se observa que las predicciones coinciden en gran medida con las etiquetas reales, con un alineamiento espacial muy preciso y niveles de confianza elevados, lo que confirma la correcta capacidad de detección en el conjunto de validación.

A continuación se presentan también ejemplos adicionales de predicciones realizadas por el modelo M7 sobre documentos completos representativos de casos reales de uso. La Figura 9 muestra una selección de páginas en las que el modelo ha detectado correctamente las tablas de calificaciones. Cada caja azul indica la localización predicha para la tabla, junto con la probabilidad estimada por el modelo. Se aprecia que las predicciones son consistentes y robustas incluso en documentos con formatos y estructuras visuales muy variados, lo que refuerza la validez del modelo para su aplicación práctica.

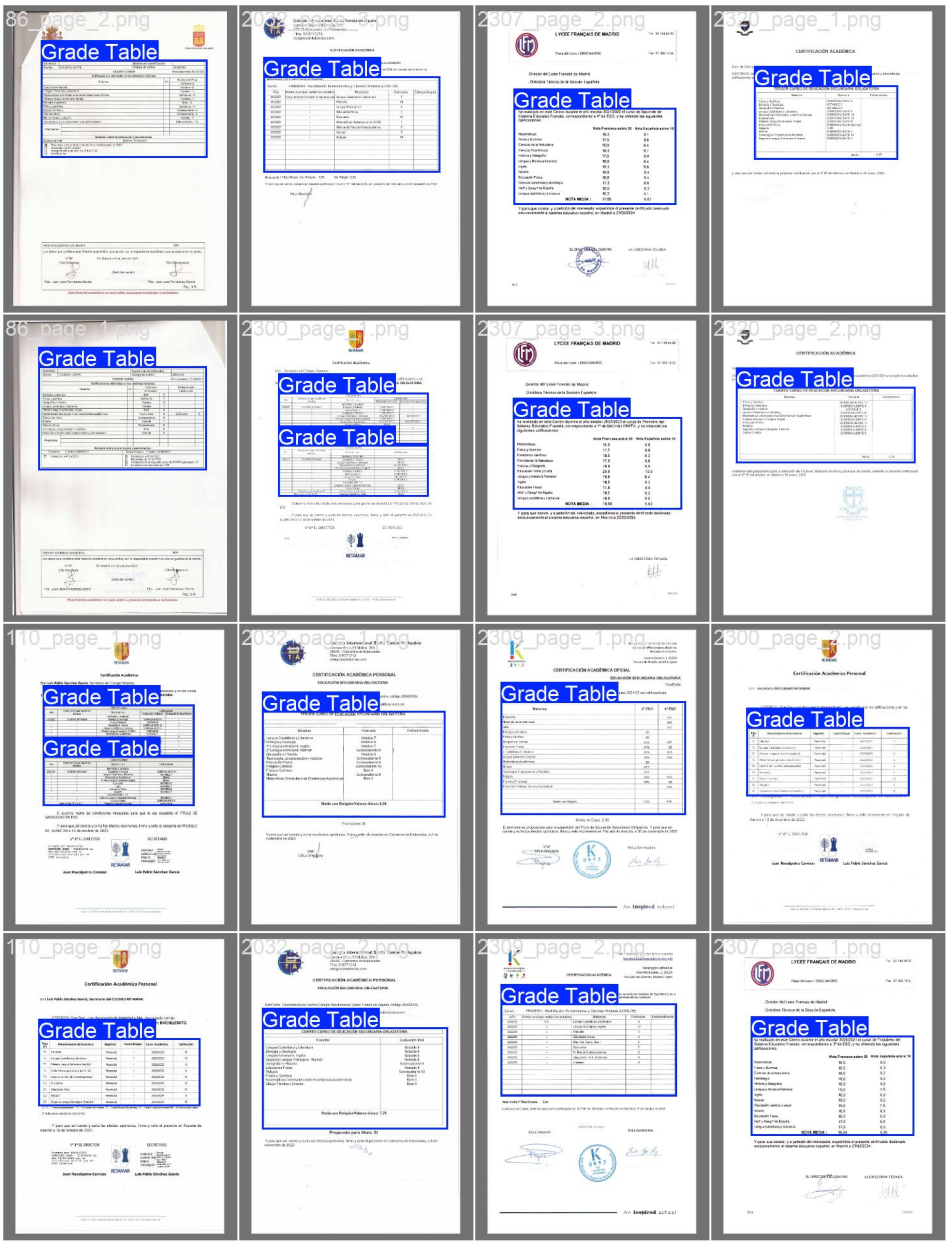


Figura 8: Ejemplo de etiquetas reales (ground truth) en el conjunto de validación.

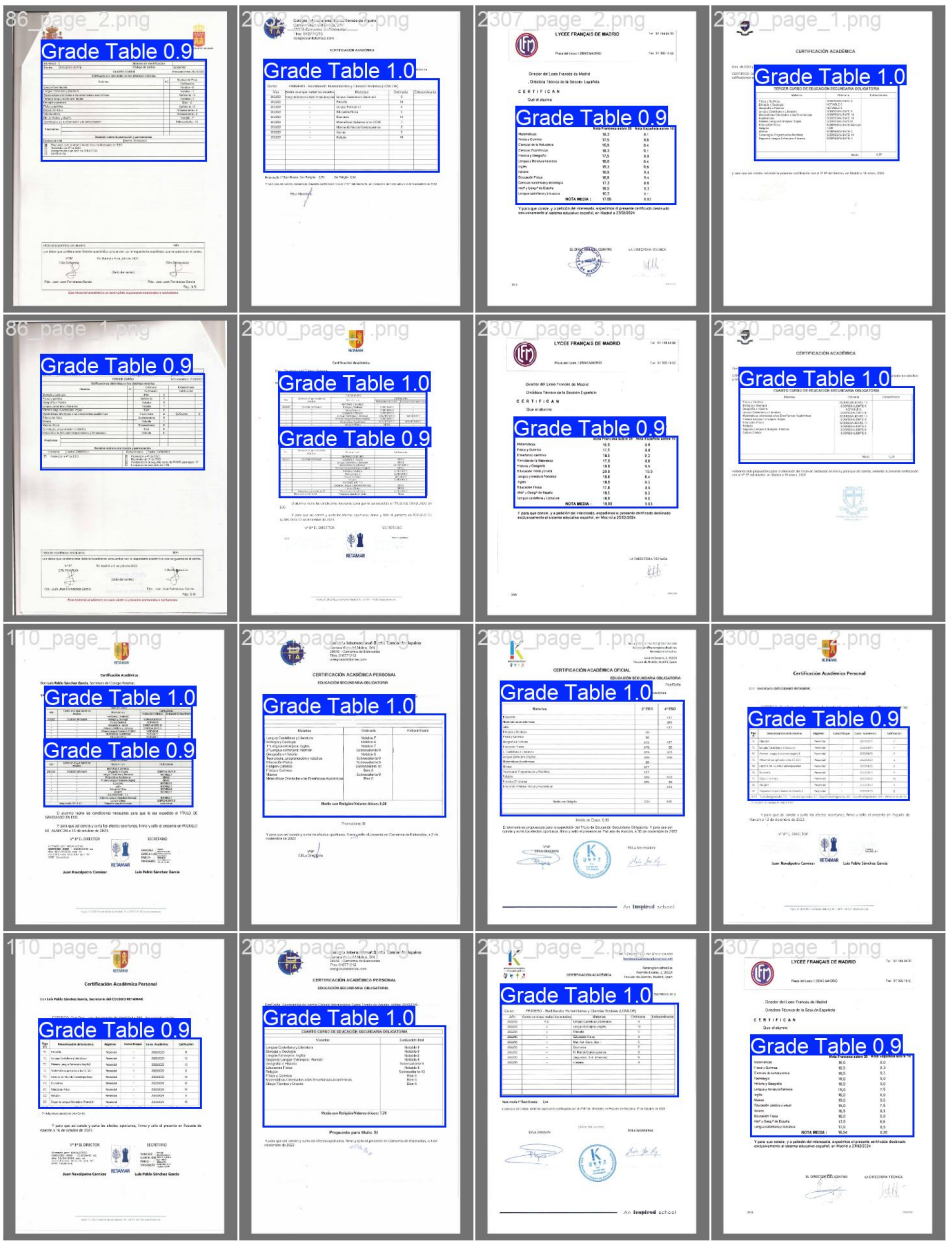


Figura 9: Ejemplo de predicciones del modelo M7 en el conjunto de validación.

En resumen, el modelo YOLOv8 entrenado en el proyecto ofrece un rendimiento muy alto y fiable en la detección automática de tablas de calificaciones. En la sección 4.2, se presentan los resultados obtenidos al integrar este modelo en el sistema completo de procesamiento y transcripción de documentos académicos.

4.2. Resultados del sistema completo de transcripción

El sistema completo, compuesto por el modelo YOLOv8 para la detección de tablas de calificaciones, el componente de identificación automática del curso académico y la transcripción de las tablas a CSV se ha aplicado a un conjunto de documentos reales no utilizados durante las fases de entrenamiento ni de validación.

Los resultados obtenidos muestran que la detección de las tablas en los documentos funciona de

manera consistente y robusta en la gran mayoría de los casos, logrando localizar correctamente las regiones de interés incluso en documentos con variaciones de formato y calidad visual. La identificación automática del curso académico también ha demostrado un comportamiento muy fiable, permitiendo contextualizar correctamente la transcripción en la mayoría de los documentos procesados.

En cuanto a la transcripción automática de las tablas, el sistema es también capaz de generar resultados válidos en la gran mayoría de los casos. No obstante, se han observado algunos errores puntuales, fundamentalmente en documentos con tablas especialmente complejas o mal estructuradas, o en aquellos casos en los que la calidad de la imagen extraída no es del todo óptima. Los errores detectados incluyen principalmente la asignación incorrecta de alguna categoría o materia, y errores en la lectura de calificaciones aisladas.

En conjunto, los resultados obtenidos confirman que el sistema desarrollado permite automatizar de forma efectiva y con un grado de fiabilidad elevado el proceso de detección y transcripción de tablas de calificaciones, si bien existe margen de mejora en la fase de transcripción para alcanzar un rendimiento plenamente satisfactorio en todos los casos.

Se observa que el sistema detecta las tablas de forma bastante precisa, incluso en documentos con variaciones de formato o calidad, lo que indica que esta parte del proceso es bastante fiable. Aunque una detección incorrecta podría afectar a la transcripción, en la mayoría de los casos los errores observados no provienen de esta fase, sino de la interpretación del contenido de las tablas. Por tanto, el principal margen de mejora se encuentra en la fase de transcripción. Una línea de trabajo prometedora sería combinar la información visual con técnicas OCR (Optical Character Recognition) especializadas para extraer el texto de cada celda de la tabla de calificaciones antes de enviarlo al modelo de lenguaje, reduciendo así la posibilidad de errores de interpretación estructural. Además, se podrían introducir mecanismos automáticos de validación que detecten incoherencias en las materias, categorías o calificaciones, permitiendo corregir o marcar para revisión los casos dudosos. Por último, ampliar la base de pruebas con documentos más diversos, incluyendo aquellos con estructuras irregulares o calidad visual reducida, puede contribuir a mejorar la robustez general del sistema frente a situaciones reales más complejas.

Figura 10: Interfaz final de la aplicación desarrollada con Streamlit. Permite cargar un PDF de calificaciones y visualizar automáticamente las tablas detectadas y transcritas.

Página 3 - Tabla 1

Curso:	PRIMERO - Bachillerato: Ciencias y Tecnología (LOMLOE)			
Año	Centro en el que realizó los estudios	Materias	Ordinaria	Extraordinaria
2022/23	Aldovea	Lengua castellana y literatura I	10MH	
2022/23	"	Primera Lengua Extranjera I	10MH	
2022/23	"	Filosofía	10	
2022/23	"	Matemáticas I	8	
2022/23	"	Educación física	10	
2022/23	"	Religión	10	
2022/23	"	Economía	10	
2022/23	"	Física y química	9	
2022/23	"	Dibujo técnico I	8	

Recorte de la tabla

	Lengua castellana y literatura I	Lengua Castellana	10
0	Primera Lengua Extranjera I	Inglés	10
1	Filosofía	Filosofía	10
2	Matemáticas I	Matemáticas	8
3	Educación física	Educación Física	10
4	Religión	Religión	10
5	Economía	Economía	10
6	Física y química	Física y Química	9
7	Dibujo técnico I	Dibujo Técnico	8

Figura 11: Ejemplo de tabla de calificaciones transcrita a formato estructurado (CSV) a partir de la imagen detectada.

5. Discusión

Los resultados obtenidos en este trabajo demuestran la viabilidad del enfoque propuesto para automatizar la detección y transcripción de tablas de calificaciones en documentos académicos. El modelo YOLOv8 ha mostrado un rendimiento excelente en la detección de tablas, incluso en documentos reales con formatos variados, mientras que el sistema completo es capaz de proporcionar transcripciones correctas en la gran mayoría de los casos.

Aunque la detección de tablas por el sistema es ya bastante precisa, la transcripción automática presenta todavía algunos retos. En particular, se han identificado errores puntuales en documentos con tablas complejas, estructuras irregulares o baja calidad visual. Estos fallos no comprometen el funcionamiento general del sistema, pero sí limitan su uso en entornos donde se requiere una fiabilidad absoluta.

El proyecto confirma que el sistema puede ser de gran utilidad en tareas donde se requiere procesar grandes volúmenes de documentos de forma rápida y eficiente. Aunque aún no es perfecto, especialmente en casos complejos, el enfoque desarrollado sienta una base sólida sobre la que seguir construyendo. Con futuras mejoras en la transcripción y una mayor cobertura de casos diversos, el sistema tiene potencial para integrarse en flujos de trabajo reales y reducir de forma significativa la intervención manual.

Referencias

[1] Ultralytics. *Explorar Ultralytics YOLOv8*. Disponible en: https://docs.ultralytics.com/es/models/yolov8/

[2] Y. LeCun, Y. Bengio, G. Hinton. *Deep learning. Nature, 521(7553), 436–444.*. May 2015. https://doi.org/10.1038/nature14539

[3] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning. MIT Press.* . https://www.deeplearningbook.org/

[4] *Stanford University. CS231n: Convolutional Neural Networks for Visual*. https://cs231n.github.io/

[5] Nafiz Shahriar. *What is Convolutional Neural Network (CNN)? Deep Learning*. Medium. Disponible en: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5.

[6] H. Ide, T. Kurita. *Improvement of learning for CNN with ReLU activation by sparse regularization*. 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017. https://ieeexplore.ieee.org/document/7966185

[7] M. D. Zeiler, R. Fergus. *Visualizing and Understanding Convolutional Networks*. European Conference on Computer Vision (ECCV), 2014. https://arxiv.org/abs/1311.2901

[8] Sharkyun. *Computer Vision — Object Detection: One-stage vs Two-stage*. Medium, 2021. Disponible en: https://sharkyun.medium.com/computer-vision-object-detection-one-stage-vs-two-stage-b05dbff88195

[9] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. https://arxiv.org/abs/1506.02640

[10] J. Redmon, A. Farhadi. *YOLO9000: Better, Faster, Stronger*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. https://arxiv.org/abs/1612.08242

[11] Ultralytics. *YOLO Performance Metrics*. Ultralytics Documentation, 2024. https://docs.ultralytics.com/es/guides/yolo-performance-metrics/

[12] A. Gupta, S. Manandhar, S. Mishra, D. Das, P. P. Roy. *Evaluation metrics for object detection: A survey and insights*. Multimedia Tools and Applications, Springer, 2023. Disponible en: https://link.springer.com/article/10.1007/s11042-023-16736-5