UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Grado en Ingeniería Informática

Impacto de algoritmos de mejora en el rendimiento de redes neuronales de detección

Impact of enhancement algorithms on the performance of detection neural networks

Realizado por
María Jesús García Bravo

Tutorizado por
Miguel Ángel Molina Cabello

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, septiembre de 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Grado en Ingeniería Informática

Impacto de algoritmos de mejora en el rendimiento de redes neuronales de detección

Impact of enhancement algorithms on the performance of detection neural networks

Realizado por
María Jesús García Bravo

Tutorizado por
Miguel Ángel Molina Cabello

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, septiembre de 2025

Fecha defensa

Resumen

El procesamiento de imágenes mediante técnicas de inteligencia artificial ha experimentado un avance significativo en los últimos años, transformando sectores como la seguridad, la conducción autónoma, el diagnóstico médico y la visión por computador. Estas tecnologías, al permitir que las máquinas analicen y comprendan información visual, modifican la forma en que interactuamos con los sistemas automatizados y cómo estos interpretan su entorno.

Dentro de este contexto, las redes neuronales convolucionales (CNN) han demostrado ser fundamentales para tareas de detección y clasificación de objetos, debido a su capacidad para identificar patrones visuales con elevada precisión. Entre los modelos más destacados, YOLO ha sobresalido por su rapidez y capacidad de realizar predicciones en tiempo real, constituyendo una herramienta eficaz en una amplia variedad de aplicaciones.

No obstante, a pesar de su efectividad en condiciones ideales, la aplicación de estos modelos en entornos del mundo real presenta desafíos importantes. Uno de los factores más determinantes que puede afectar su rendimiento son las variaciones en la iluminación, ya sea por cambios en las condiciones ambientales o por limitaciones técnicas en los sistemas de captura de imágenes. Dichas variaciones alteran las características visuales de las imágenes, lo que puede reducir la precisión de los modelos de detección de objetos.

Este estudio se centra en evaluar cómo las variaciones de iluminación impactan en el desempeño de los modelos de detección de objetos y en explorar estrategias para mejorar su robustez mediante la aplicación de algoritmos de mejora de imagen, tales como la máscara de enfoque (unsharp mask en inglés), ecualización de histograma, ecualización limitada adaptativa por contraste (CLAHE) y transformación logarítmica. Se analizarán los efectos de estas técnicas sobre las imágenes y se interpretarán los resultados obtenidos, con el objetivo de proporcionar conclusiones significativas.

Keywords: aprendizaje profundo, detección de objetos, YOLO, variabilidad de iluminación, algoritmos de mejora de imagen

Abstract

Image processing through artificial intelligence techniques has experienced significant progress in recent years, transforming sectors such as security, autonomous driving, medical diagnosis, and computer vision. These technologies, by enabling machines to analyze and understand visual information, are changing the way we interact with automated systems and how these systems interpret their environment.

Within this context, convolutional neural networks (CNNs) have proven to be fundamental for object detection and classification tasks, due to their ability to identify visual patterns with high accuracy. Among the most prominent models, YOLO has stood out for its speed and ability to perform real-time predictions, making it an effective tool in a wide variety of applications.

Nevertheless, despite their effectiveness under ideal conditions, applying these models in real-world environments presents significant challenges. One of the most determining factors that can affect their performance is variations in lighting, either due to changes in environmental conditions or technical limitations in image capture systems. Such variations alter the visual characteristics of images, which can reduce the accuracy of object detection models.

This study focuses on evaluating how lighting variations impact the performance of object detection models and exploring strategies to improve their robustness through the application of image enhancement algorithms, such as unsharp mask, histogram equalization, contrast-limited adaptive histogram equalization (CLAHE), and logarithmic transformation. The effects of these techniques on the images will be analyzed and the results interpreted, with the aim of providing meaningful conclusions.

Keywords: deep learning, object detection, YOLO, lighting variability, image enhancement algorithms

Índice

1 Introducción

1.1. Motivación

La repercusión que ha tenido la inteligencia artificial y varios de sus componentes como el aprendizaje profundo y las redes neuronales en la actualidad es indiscutible. En el ámbito del procesamiento de imágenes, las redes neuronales convolucionales o CNN en inglés destacan por su capacidad para detectar y clasificar diversos objetos en imágenes. Sus aplicaciones pueden llegar a ser muy variadas, entre las que destacan su uso en el análisis de imágenes médicas y diagnóstico, así como en la conducción autónoma o en los sistemas de vigilancia.

Una de las arquitecturas de modelos y algoritmos de detección que ha ganado relevancia en los últimos años es YOLO (You Only Look Once), un algoritmo de inteligencia artificial especializado en la detección de objetos y segmentación de imágenes en tiempo real, utilizando para ello cuadros delimitadores (bounding boxes) que asignan a cada objeto una posición o región específica en la imagen, junto con una categoría y una probabilidad asociada a cada predicción. Además, YOLO se distingue por su capacidad para clasificar y localizar múltiples objetos de manera eficiente y rápida, empleando únicamente una red neuronal convolucional, a diferencia de otros modelos que requieren múltiples etapas de procesamiento, ya que divide la imagen en regiones y predice simultáneamente las probabilidades de las clases y las coordenadas de los cuadros delimitadores.

Comprender el impacto de la calidad de la imagen en el rendimiento de las arquitecturas de redes neuronales convolucionales es fundamental para el desarrollo de nuevas arquitecturas eficientes y la optimización de las ya existentes. Un tratamiento adecuado de este aspecto es esencial para garantizar un rendimiento óptimo de las CNN y generar confianza en los resultados obtenidos, ya que fallos en las predicciones podrían tener consecuencias críticas, como un diagnóstico médico erróneo.

Aunque en la mayoría de los estudios, las redes se entrenan y prueban con conjuntos de datos de imágenes de alta calidad, en aplicaciones del mundo real no siempre se tiene control sobre la calidad de las imágenes, las cuales pueden verse afectadas por factores externos o imprevistos. Por lo tanto, es crucial identificar estos posibles problemas e investigar cómo se podría optimizar el rendimiento de las redes neuronales en condiciones adversas. Esto implica también diseñar estrategias de entrenamiento y evaluación que simulen escenarios del mundo real para anticipar posibles fallos.

Uno de los factores clave que afecta significativamente a la calidad de las imágenes es el brillo, el cual influye en el nivel de iluminación de las mismas, modificando de manera directa la visibilidad de detalles y el contraste general de la escena. La variación en el brillo puede alterar significativamente la calidad visual y, por ende, el rendimiento de los modelos, especialmente cuando se enfrentan a tareas de detección y clasificación de objetos. Este aspecto es especialmente relevante en situaciones cotidianas, ya que las imágenes pueden capturarse en entornos con iluminación variable o insuficiente, por ejemplo, en sistemas de vigilancia, debido a factores como la hora del día, las condiciones meteorológicas o los cambios en la iluminación artificial, lo que dificulta la capacidad de los modelos para realizar predicciones precisas. Por ello, comprender cómo la luminosidad afecta a los resultados permite ajustar los modelos y preprocesar los datos de manera más efectiva.

Al simular condiciones de baja luminosidad, disminuyendo manualmente el brillo de una imagen, se puede observar cómo el rendimiento del modelo varía dependiendo de la cantidad de información visual que se conserve. Además, la aplicación de algoritmos de mejora de imágenes durante el preprocesamiento de datos puede tener un efecto crucial en la compensación de estos problemas de luminosidad. Entre los métodos utilizados para mejorar la calidad de las imágenes se encuentran técnicas como la ecualización de histograma, la cual redistribuye los niveles de intensidad de la imagen para mejorar el contraste, así como la transformación logarítmica, que permite resaltar detalles en zonas oscuras sin saturar las más iluminadas.

El impacto de estas técnicas de mejora es un aspecto fundamental a investigar, ya que puede influir en la precisión de los modelos, especialmente cuando son entrenados con imágenes de calidad variable. La correcta elección de estos algoritmos de preprocesamiento puede marcar la diferencia en el rendimiento del modelo final, asegurando que se aproveche al máximo la información visual disponible.

1.2. Objetivos

Este estudio tiene como objetivo analizar el impacto que la variabilidad de brillo en una imagen tiene sobre el rendimiento de redes neuronales convolucionales preentrenadas, concretamente en tareas de detección de objetos realizadas por el modelo YOLO. En particular, se estudiará cómo diferentes algoritmos de mejora de imágenes, aplicados a imágenes con condiciones de iluminación variables, pueden influir en la precisión y efectividad de la detección.

Los algoritmos de mejora que se usarán incluyen la ecualización del histograma (HE), una técnica que tiene como objetivo optimizar el contraste de una imagen mediante la redistribución de los valores de intensidad más frecuentes en ella y la ecualización del histograma adaptativa limitada por contraste (CLAHE), una variante de dicho algoritmo que, en lugar de aplicar un único histograma para toda la imagen, emplea varios correspondientes a distintas secciones de la misma, evitando un aumento excesivo del ruido.

Por otro lado, el conjunto de datos en el que se basará este estudio será COCO, en concreto la versión de 2017, que es un conjunto ampliamente utilizado para la evaluación comparativa de modelos de visión por computador, aunque se hablará del mismo más tarde.

Para concretar, este trabajo evaluará el rendimiento de la red YOLO, en este caso usando una red ya preentrenada, en las imágenes originales, analizando posteriormente cómo este varía al aplicar una disminución aleatoria acotada del brillo. El objetivo es que esta reducción sea significativa, pero no crítica, ya que una disminución excesiva impediría distinguir las características de la imagen. Finalmente, se examinará el impacto de diferentes algoritmos de mejora de imágenes sobre el rendimiento del modelo, simulando su aplicación como un paso de preprocesamiento en las imágenes con brillo reducido.

Además, para garantizar un análisis riguroso y confiable, se emplearán métricas de evaluación estandarizadas que permitan medir de forma objetiva el desempeño del modelo en cada uno de los escenarios planteados. Estas métricas no solo facilitarán la comparación directa entre los distintos resultados obtenidos, sino que también permitirán identificar patrones de comportamiento y posibles áreas de mejora. De esta manera, será posible evaluar en qué medida las técnicas de preprocesamiento aplicadas influyen en la precisión, la robustez y la efectividad general del sistema de detección de objetos, proporcionando así una base cuantitativa sólida para extraer conclusiones significativas.

1.3. Estructura del documento

Este trabajo se organiza de la siguiente manera:

- En primer lugar, se presenta la introducción, donde se expone la motivación del estudio, se formulan los objetivos del trabajo y se describe brevemente la estructura seguida a lo largo del documento.
- En segundo lugar, se desarrolla el estado del arte, ofreciendo una visión general de la inteligencia artificial aplicada a la visión por computador. Se explican los fundamentos de Machine Learning y Deep Learning, así como el funcionamiento de las redes neuronales y redes convolucionales, finalizando con la presentación de YOLO como modelo de referencia.
- A continuación, se detallan las tecnologías utilizadas para llevar a cabo el proyecto, incluyendo lenguajes de programación, librerías, plataformas de software y entornos de trabajo empleados durante el desarrollo.
- Después, se describen los algoritmos de mejora de imágenes considerados en este estudio, explicando su funcionamiento y usos.
- Seguidamente, se presenta la metodología de trabajo, abordando el enfoque científico adoptado, la organización de las iteraciones de desarrollo y el flujo experimental del proyecto.
- Más adelante, se muestran los resultados obtenidos. Se explican los métodos aplicados, el conjunto de datos empleado, las métricas de evaluación y se incluyen tanto resultados cualitativos como cuantitativos.
- Por otro lado, los resultados del estudio concluyen en una discusión donde se interpretan los hallazgos obtenidos y se señalan sus limitaciones y aportes principales.
- Finalmente, se exponen las conclusiones generales derivadas del estudio y se plantean posibles líneas de investigación futura, destacando las aportaciones más relevantes del trabajo.

2

Estado del arte

2.1. Inteligencia artificial

Antes de comenzar a hablar sobre la Inteligencia Artificial, es necesario entender primero qué es la inteligencia en sí misma. La inteligencia humana ha sido definida tradicionalmente como la capacidad de adquirir conocimientos, razonar, resolver problemas, planificar, pensar de manera abstracta, comprender ideas complejas y aprender de la experiencia. En esencia, se trata de la habilidad que poseemos los humanos para percibir, interpretar y adaptarnos a nuestro entorno mediante el análisis, la síntesis y la generalización de información.

A lo largo de la historia, el deseo de imitar esta capacidad ha llevado a científicos, ingenieros y filósofos a desarrollar sistemas capaces de replicar, al menos en parte, estos procesos cognitivos. Esta ambición dio origen a la Inteligencia Artificial (IA), un campo interdisciplinar que combina informática, matemáticas, lógica, filosofía, neurociencia y lingüística, entre otras disciplinas.

Por su parte, la Inteligencia Artificial, es el intento por parte de la informática de reproducir dicha inteligencia humana en las máquinas. A grandes rasgos, se entiende como la rama de la ciencia computacional que se encarga de crear sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como la percepción visual, el reconocimiento de voz, la toma de decisiones, la traducción entre idiomas, el razonamiento lógico o el aprendizaje autónomo [1].

Este término fue acuñado por John McCarthy en 1956 durante la Conferencia de Dartmouth, un evento histórico que reunió a algunos de los científicos más destacados de la época para discutir la posibilidad de crear una máquina que pudiera pensar y razonar como un ser humano. Este evento es considerado el punto de partida formal de la disciplina. Sin embargo, los conceptos e ideas que están detrás de la inteligencia artificial se remontan mucho más

atrás. En la década de 1940, matemáticos e ingenieros como Norbert Wiener y John von Neumann desarrollaron las teorías de los sistemas de control y la computación que sentaron las bases técnicas y conceptuales para la IA. Wiener, en particular, es considerado el padre de la cibernética, una ciencia que estudia los sistemas autorregulados, fundamental para entender cómo las máquinas pueden aprender y adaptarse.

Además de las bases matemáticas y técnicas, la cultura popular y la ciencia ficción jugaron un papel importante en la inspiración y difusión de la idea de máquinas inteligentes. Desde la década de 1920, la literatura y el cine presentaron robots y autómatas con capacidades cognitivas, lo que ayudó a popularizar la idea de máquinas pensantes y generó un interés social amplio que ha perdurado hasta hoy [2].

En la actualidad, la IA ha evolucionado hasta convertirse en una tecnología clave para el desarrollo científico, económico y social. Sus aplicaciones abarcan múltiples sectores, como se puede distinguir en la Figura 1.

Figura 1: Aplicaciones de la IA en la actualidad [2].

En función de su nivel de capacidad y desarrollo, la inteligencia artificial suele clasificarse en tres grandes categorías:

1. **IA débil** (o estrecha): hace referencia a sistemas diseñados para realizar tareas concretas dentro de un dominio específico, como asistentes virtuales, motores de recomendación o sistemas de reconocimiento facial. No poseen consciencia ni comprensión general del mundo, y actúan únicamente dentro de los límites para los que fueron programados o entrenados [2].

2. **IA fuerte**: en cambio, aspira a reproducir las capacidades cognitivas humanas en un sentido más amplio. Este tipo de IA sería capaz de razonar, aprender, adaptarse y resolver problemas en diferentes contextos sin intervención humana directa. A día de hoy, continúa siendo un objetivo a largo plazo más que una realidad concreta [2].

3. **IA superinteligente**: representa una hipotética forma de inteligencia que superaría a la humana en todos los aspectos: desde el razonamiento hasta la creatividad y la toma de decisiones. El posible desarrollo de esta forma de IA plantea interrogantes éticos, sociales y científicos sobre el futuro de la humanidad y su relación con la tecnología [2].

Actualmente, la IA se organiza en varias ramas según el enfoque de los problemas que abordan, siendo el Machine Learning y el Deep Learning las más destacadas. Como se muestra en la Figura 2, el Deep Learning forma parte del Machine Learning, el cual, a su vez, se enmarca dentro del campo general de la IA. Esta jerarquía refleja cómo el desarrollo de técnicas más avanzadas ha dado lugar a enfoques cada vez más especializados.

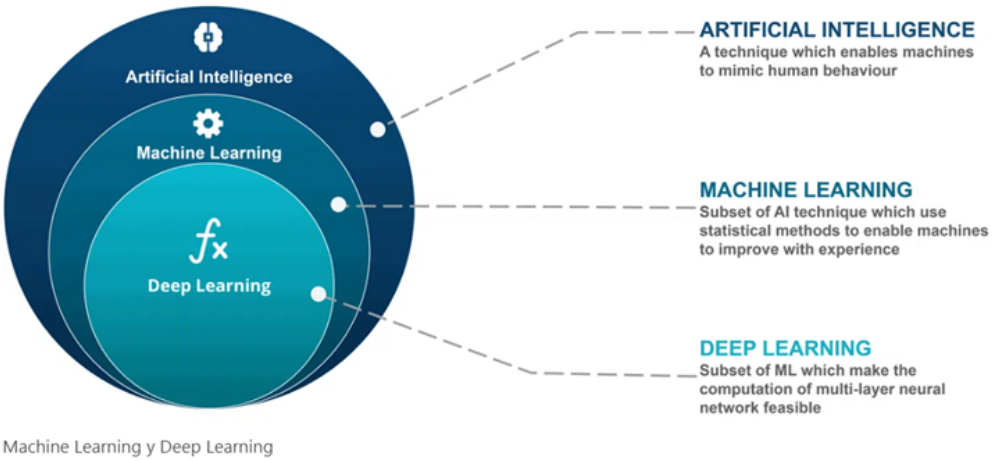


Figura 2: Relación entre Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo [3].

2.2. Deep Learning y Machine Learning

Aunque ambos términos pueden generar confusión en la comprensión general y suelen confundirse, como se ha mostrado anteriormente, el Deep Learning es una subcategoría dentro del Machine Learning, por lo que no se sitúan al mismo nivel. Aún así, es posible establecer diferencias relevantes entre ambos enfoques.

Ambos utilizan algoritmos para aprender a partir de datos, pero mientras el Machine Learning aplica modelos más simples que requieren intervención humana para definir las características relevantes, el Deep Learning organiza los algoritmos en múltiples capas de redes neuronales, lo que le permite aprender representaciones más complejas de forma automática. Además, esta estructura en capas contribuye a una mayor precisión en tareas como el reconocimiento de imágenes o el procesamiento del lenguaje.

Por otro lado, el tipo de datos que utilizan también marca una diferencia importante. El Machine Learning suele requerir datos estructurados y etiquetados, lo que implica un preprocesamiento manual. En cambio, el Deep Learning puede trabajar directamente con datos no estructurados, como imágenes o texto, extrayendo por sí mismo las características necesarias para el aprendizaje.

Esto se refleja claramente en tareas de clasificación de imágenes. Por ejemplo, si se desea entrenar un modelo para identificar distintos tipos de gatos en fotografías, como se observa en la Figura 3, un algoritmo de Machine Learning tradicional requeriría que los ingenieros definieran previamente qué características debe tener en cuenta, como la forma de las orejas, el patrón del pelaje o el tamaño de los ojos. Si en una imagen aparece un gato parcialmente cubierto, en una postura inusual o con características poco comunes, es probable que el modelo no lo reconozca correctamente, ya que depende de reglas explícitas basadas en ejemplos representativos.

En cambio, un modelo de Deep Learning puede aprender a identificar patrones más abstractos y generalizar mejor. Gracias a su estructura en capas, es capaz de combinar información visual de distintas zonas de la imagen y llegar a reconocer un gato incluso si está desenfocado, parcialmente oculto por otros objetos o si presenta características atípicas, como un pelaje poco común o una expresión facial inusual [4].

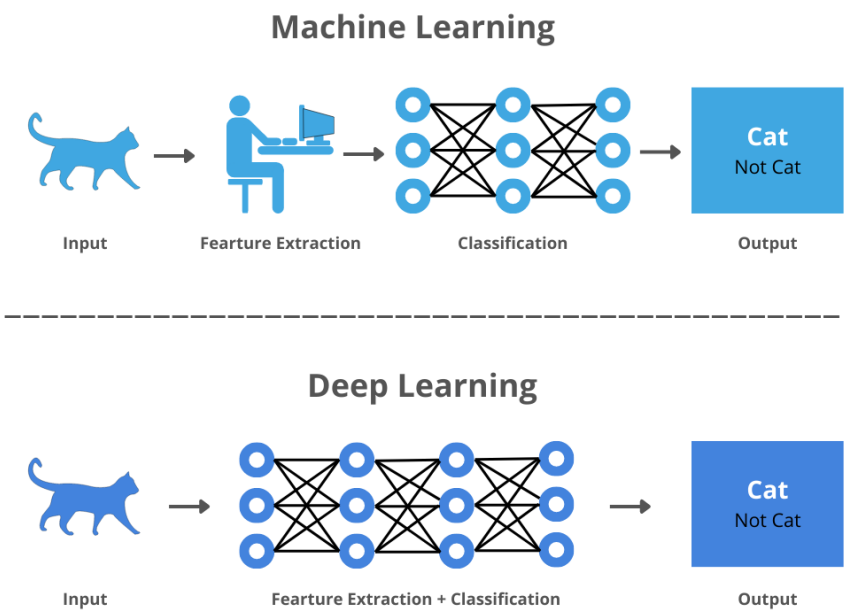


Figura 3: Ejemplo de comparación entre Machine Learning y Deep Learning en el reconocimiento de la imagen de un gato [5].

2.2.1. Machine Learning

El aprendizaje automático, conocido en inglés como Machine Learning, es una rama de la informática que se centra en identificar patrones complejos dentro de grandes volúmenes de datos. A diferencia de la programación tradicional, donde un programa sigue instrucciones fijas para realizar una tarea específica, en el aprendizaje automático los programas tienen la capacidad de aprender por sí mismos. Esto permite la extracción de información relevante de los datos sin necesidad de haber sido programados explícitamente para ello. Gracias a este proceso, el sistema mejora progresivamente su rendimiento en la tarea asignada.

El funcionamiento de un sistema de Machine Learning se comprende distinguiendo entre dos conceptos estrechamente relacionados pero distintos: el algoritmo y el modelo. Aunque a menudo se mencionan juntos, cada uno cumple una función específica dentro del proceso de aprendizaje automático.

Por un lado, un algoritmo de machine learning es un conjunto de reglas matemáticas y procedimientos estadísticos que definen cómo debe aprender un sistema a partir de los datos. Es, en esencia, el método que guía el proceso de aprendizaje: toma los datos de entrada,

los analiza, y ajusta ciertos parámetros internos para encontrar patrones o relaciones útiles. Este algoritmo no resuelve directamente el problema, sino que proporciona las instrucciones necesarias para que el sistema aprenda a resolverlo por sí mismo.

Por su parte, un modelo de machine learning es el resultado final que se obtiene tras aplicar un algoritmo a un conjunto de datos. Es una representación matemática del conocimiento adquirido, que encapsula las relaciones detectadas durante el entrenamiento. Este modelo es el que se utiliza posteriormente para hacer predicciones, clasificar información o tomar decisiones sobre nuevos datos. En otras palabras, si el algoritmo es como una receta de cocina, el modelo sería el plato ya preparado: una manifestación concreta del proceso de aprendizaje.

Esta distinción es clave para entender cómo se construyen y aplican los sistemas inteligentes. Mientras el algoritmo define el "cómo aprender", el modelo representa "lo que se ha aprendido" y se convierte en la herramienta práctica para resolver tareas reales.

Además, dependiendo del modo de aprendizaje de los modelos de aprendizaje automático, se pueden distinguir cuatro tipos de aprendizaje:

- **Aprendizaje supervisado:** El aprendizaje supervisado es una de las formas más comunes de machine learning y se basa en entrenar un modelo con datos etiquetados, es decir, ejemplos en los que se conoce de antemano la categoría o el valor que se desea predecir. Durante el entrenamiento, el modelo aprende a establecer una relación entre las características de entrada y la etiqueta asociada, lo que le permite realizar predicciones sobre nuevos datos. Este enfoque se aplica tanto a problemas de clasificación como de regresión, y emplea algoritmos como la regresión lineal, la regresión logística, los árboles de decisión o las máquinas de vectores de soporte.

- **Aprendizaje no supervisado:** El aprendizaje no supervisado es una técnica que trabaja con datos no etiquetados, permitiendo que el modelo descubra patrones o relaciones ocultas en los datos. A diferencia del aprendizaje supervisado, no se proporciona una salida conocida durante el entrenamiento, lo que lo hace útil para explorar conjuntos de datos complejos sin conocimiento previo. Se utiliza en tareas como la agrupación de elementos similares (clustering) y la reducción de dimensionalidad, con algoritmos habituales como k-means y el Análisis de Componentes Principales (PCA).

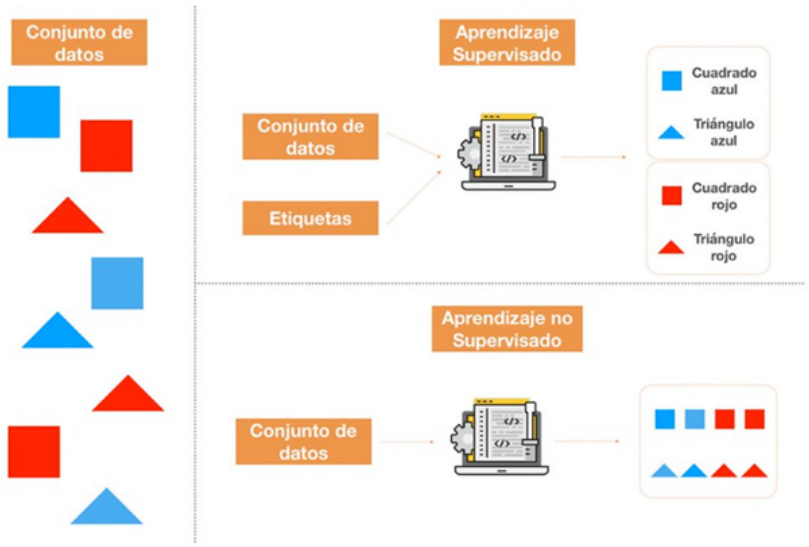


Figura 4: Diferencias entre Aprendizaje Supervisado y No Supervisado [6].

En la Figura 4 se muestra la diferencia entre aprendizaje supervisado y no supervisado. El conjunto de datos consiste en figuras geométricas con variaciones en color y forma.

En el aprendizaje supervisado, el sistema recibe tanto las figuras como las etiquetas correspondientes, por ejemplo, “cuadrado azul”, “cuadrado rojo”, “triángulo azul” o “triángulo rojo”. Esto permite que el modelo aprenda a asociar cada imagen con su etiqueta, de modo que, una vez entrenado, pueda predecir correctamente la categoría de nuevas figuras basándose en su color y forma.

En contraste, en el aprendizaje no supervisado, el sistema solo dispone de las figuras sin etiquetas. Aquí, el algoritmo identifica patrones o características comunes entre los datos para agruparlos en conjuntos homogéneos. Por tanto, su tarea se centra en agrupar aquellas figuras con características similares, como puede ser encontrar todos los cuadrados y, por otro lado, todos los círculos [6].

- **Aprendizaje semi-supervisado:** El aprendizaje semisupervisado representa un enfoque intermedio entre el aprendizaje supervisado y el no supervisado. Se utiliza cuando se dispone de una gran cantidad de datos sin etiquetar y solo una pequeña proporción de datos etiquetados, lo cual es habitual en contextos reales donde el etiquetado manual resulta costoso o inviable. En este modelo, los datos etiquetados actúan como guía inicial para que el sistema pueda inferir patrones y extender ese conocimiento al resto del

conjunto no etiquetado. Esto permite que el modelo aproveche al máximo la información disponible, mejorando su precisión y generalización sin necesidad de contar con grandes volúmenes de datos clasificados. Además, el aprendizaje semisupervisado resulta especialmente útil en aplicaciones como reconocimiento de imágenes o análisis de texto, donde etiquetar todos los datos sería demasiado costoso o laborioso [7].

- **Aprendizaje por refuerzo:** El aprendizaje por refuerzo es una técnica en la que un agente aprende a tomar decisiones a través de la interacción continua con un entorno, que es el contexto o ambiente en el que se interactúa, con limitaciones propias según el problema. A lo largo de este proceso, el agente recibe señales en forma de recompensas o penalizaciones en función de las acciones que realiza, lo que le permite ajustar su comportamiento con el objetivo de maximizar la recompensa acumulada a lo largo del tiempo. Este enfoque se inspira en el aprendizaje por ensayo y error, y no requiere datos etiquetados previamente, sino que se basa en la retroalimentación que el entorno proporciona tras cada acción. Es posible ver una representación de este funcionamiento en la Figura 5. A diferencia de otro tipo de aprendizajes, se adapta especialmente bien a situaciones donde las decisiones tienen consecuencias a largo plazo. Es decir, la acción tomada en un momento determinado puede influir en los estados futuros del entorno y, por tanto, en las recompensas que se obtendrán. Este tipo de aprendizaje es fundamental en campos como la robótica o los videojuegos [8].

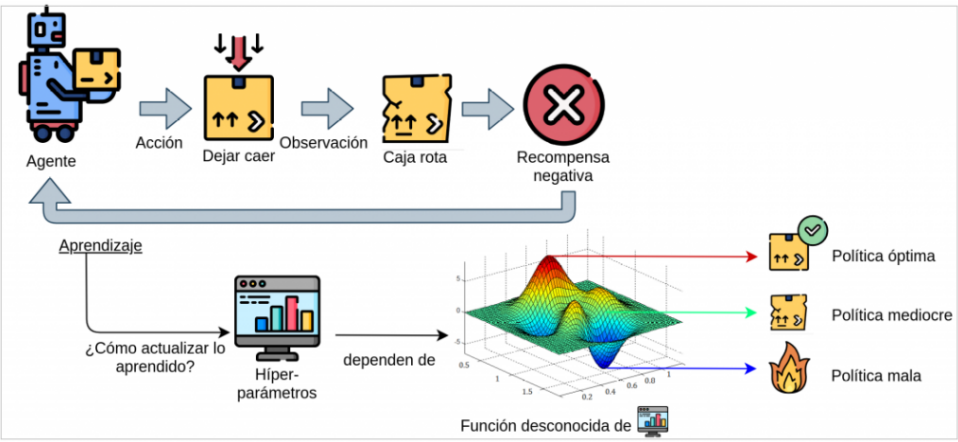


Figura 5: Esquema del Aprendizaje por Refuerzo: Interacción, Recompensa y Mejora de Políticas [9].

2.2.2. Deep Learning

Como ya se ha descrito anteriormente, el Deep Learning, o aprendizaje profundo, es una subdisciplina del aprendizaje automático (Machine Learning). Principalmente, se basa en el uso de redes neuronales artificiales para procesar grandes volúmenes de datos. Estas redes están inspiradas en el funcionamiento del cerebro humano y es una disciplina que ha avanzado bastante en los últimos años.

Actualmente, el Deep Learning está presente en muchos aspectos de nuestra vida diaria, desde los asistentes virtuales y los sistemas de recomendación en plataformas digitales, hasta aplicaciones más complejas como la conducción autónoma, el diagnóstico médico por imágenes o la traducción automática. Su capacidad para aprender de grandes volúmenes de datos lo convierte en una herramienta clave en el desarrollo de tecnologías inteligentes.

2.3. Redes Neuronales Artificiales

Una red neuronal artificial es un modelo computacional diseñado para imitar, en cierta medida, el funcionamiento del cerebro humano. Su arquitectura se inspira en el sistema nervioso biológico, donde las neuronas, que son células especializadas, se conectan entre sí formando una red altamente interconectada que transmite señales eléctricas para procesar información.

De forma análoga, una red neuronal está compuesta por unidades llamadas neuronas artificiales, que se organizan en capas y colaboran para procesar datos, transmitir información entre sí y resolver problemas complejos.

El funcionamiento básico de una red neuronal artificial se basa en la interconexión de distintas capas de neuronas. Estas capas están organizadas de forma secuencial, y cada neurona dentro de una capa está conectada a las neuronas de la capa siguiente mediante enlaces llamados nodos. Cada neurona recibe información desde la capa anterior, la procesa internamente y envía su salida a la siguiente capa. Este proceso se repite capa por capa hasta llegar a la capa de salida, que es la encargada de generar la respuesta final del sistema, ya sea una predicción, una clasificación o una decisión [10]. Durante el entrenamiento, la red ajusta los pesos de estas conexiones mediante algoritmos de aprendizaje, optimizando su capacidad para producir resultados precisos a partir de los datos de entrada.

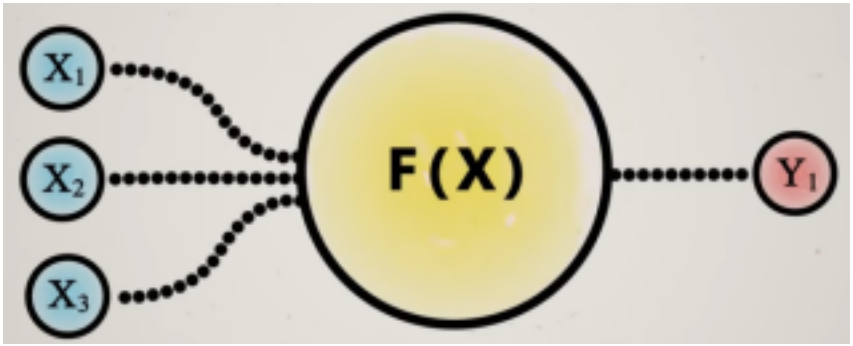


Figura 6: Estructura Neurona Artificial básica [11].

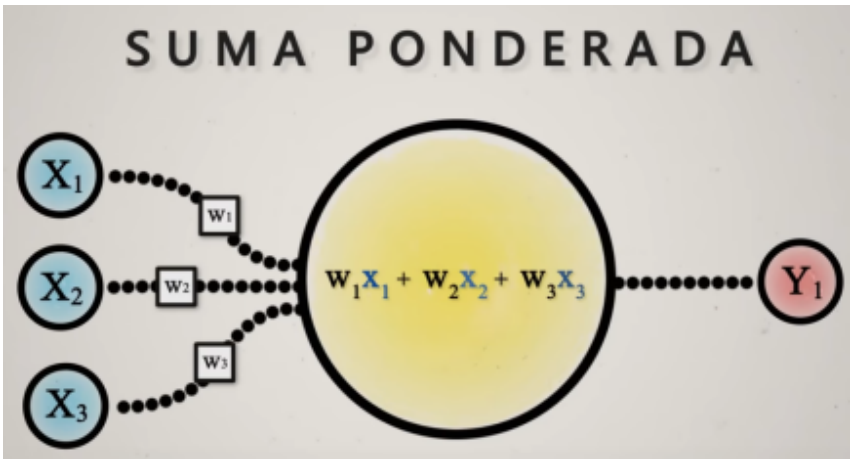


Figura 7: Cálculo de la Suma Ponderada en una Neurona Artificial [11].

Desde un punto de vista matemático, cada neurona artificial puede entenderse como una función que recibe una o varias entradas, realiza ciertos cálculos internos y genera una o más salidas. En otras palabras, transforma los datos de entrada mediante operaciones como sumas ponderadas y la aplicación de funciones de activación, cuyo resultado depende de los pesos asignados a cada entrada y de un sesgo que ajusta la salida final. Esta representación se muestra en la Figura 6.

Este comportamiento es muy similar al de una regresión lineal múltiple, donde se combinan varias variables de entrada mediante coeficientes (equivalentes a los pesos) y un término independiente (el sesgo o bias) para predecir una salida. En una neurona artificial, esta suma ponderada, que se aprecia en la Figura 7, se convierte en la entrada de una función de activación, que introduce no linealidad al modelo y permite que la red neuronal pueda aprender relaciones complejas entre los datos, más allá de lo que una simple regresión lineal podría captar.

Los pesos determinan la influencia de cada entrada sobre la salida: cuanto mayor sea el peso, mayor será su impacto. El sesgo, por su parte, se interpreta como una conexión adicional a la neurona, cuyo valor de entrada suele ser uno, aunque puede cambiar dependiendo de nuestras necesidades, como se representa en la Figura 8. Matemáticamente, actúa como el término independiente en una regresión lineal, desplazando la función de activación hacia arriba o hacia abajo y mejorando así la capacidad de ajuste del modelo.

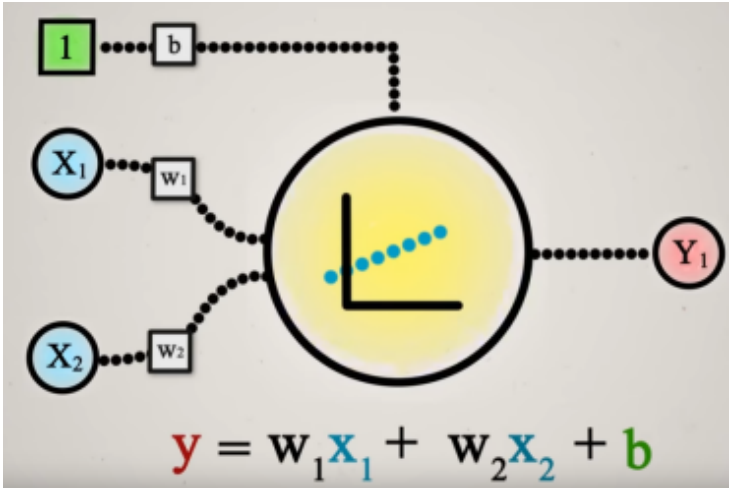


Figura 8: Modelo de Neurona Artificial: Cálculo de la Salida con Pesos y Sesgo [11].

Ambos parámetros son esenciales en el aprendizaje automático, ya que se optimizan durante el entrenamiento para reducir el error y mejorar la precisión del sistema. Gracias a esto, las redes neuronales pueden modelar relaciones complejas y abordar tareas que van desde la clasificación de imágenes hasta la predicción de series temporales.

La salida de una neurona artificial es un número que representa una respuesta o decisión basada en los datos recibidos. Su interpretación depende del tipo de problema, y para adaptarla se usan funciones de activación, que transforman el resultado interno en un valor útil. Entre las más utilizadas se encuentran:

- **Sigmoide:** Convierte cualquier número en un valor entre 0 y 1, ideal para tareas de clasificación binaria.

- **Softmax:** Transforma un conjunto de salidas en probabilidades que suman 1, útil en clasificación multiclase.

- **ReLU (Rectified Linear Unit):** Devuelve 0 si la entrada es negativa y el mismo valor si es positiva, muy común por su eficiencia y simplicidad.

Estas tres funciones se encuentran representadas en la Figura 9 a continuación.

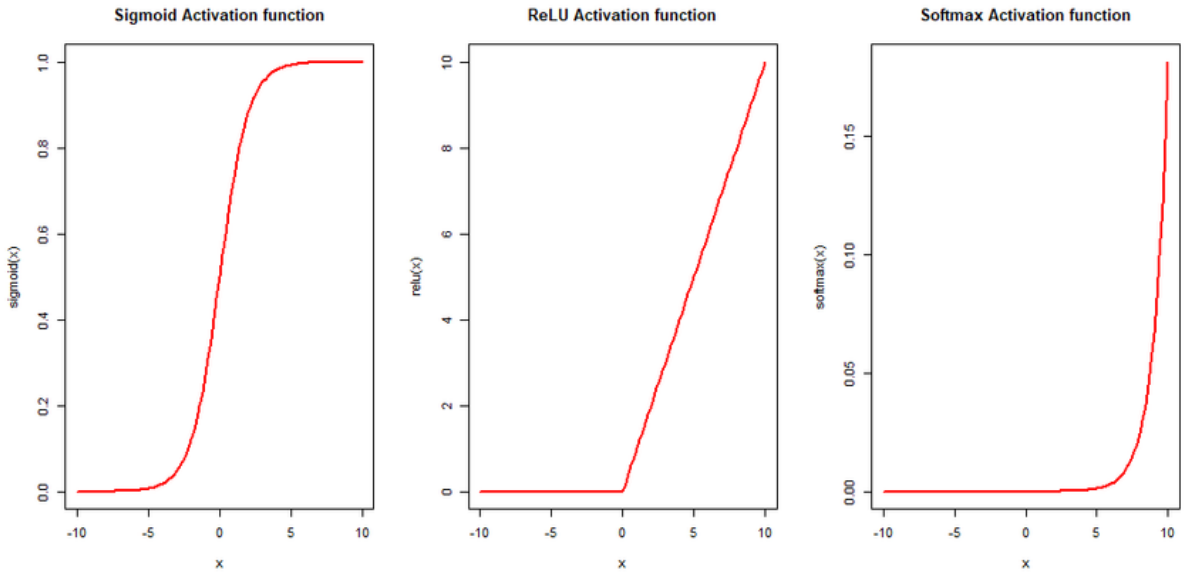


Figura 9: Funciones de Activación en Redes Neuronales: Sigmoid, ReLU y Softmax [12].

Estas funciones permiten que la red neuronal no solo procese datos, sino que también tome decisiones de forma más precisa y adaptada al tipo de tarea que se le ha asignado para llegar a resolver problemas más complejos [13].

Una vez introducido el concepto de neurona artificial, es esencial comprender cómo se organizan estas unidades dentro de una red neuronal. Toda red neuronal está estructurada en tres capas fundamentales: la capa de entrada, una o varias capas ocultas y la capa de salida. Cada una de ellas tiene una función específica en el funcionamiento de la red. La Figura 10 ilustra una posible representación.

- **Capa de entrada:** Es la primera capa de la red y su función es recibir los datos en bruto que se van a procesar. Cada neurona en esta capa representa una característica del conjunto de datos (por ejemplo, un píxel de una imagen o una variable numérica). Esta capa no realiza ningún cálculo, simplemente transmite la información a la siguiente capa. Además, el número de neuronas en esta capa depende directamente de la cantidad de características o dimensión que tenga el conjunto de datos, ya que cada neurona representará una de esas variables.

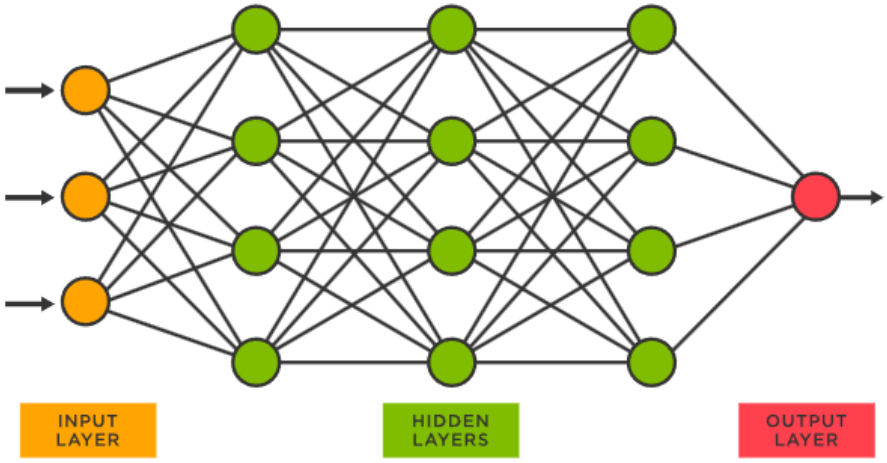


Figura 10: Estructura de una Red Neuronal Profunda [14].

- **Capa oculta:** Son una o varias capas intermedias entre la entrada y la salida. Aquí es donde ocurre el procesamiento más complejo. En cada una de estas capas encontramos un conjunto de neuronas interconectadas con las neuronas de la capa anterior y la siguiente. Así, las neuronas en estas capas reciben las señales de la capa anterior, las combinan mediante pesos y sesgos, y aplican una función de activación para generar una salida. Durante el entrenamiento de la red, los pesos y sesgos asociados a estas conexiones se ajustan automáticamente, permitiendo que la red aprenda y se adapte progresivamente a los datos de entrada. De esta forma, permiten que la red aprenda patrones, relaciones y estructuras complejas en los datos. Cuantas más capas ocultas tenga una red, más profunda y potente puede ser su capacidad de aprendizaje.

- **Capa de salida:** La capa de salida es la última capa de una red neuronal y su función es generar el resultado final del modelo, en función del tipo de problema que se desea resolver. Esta capa recibe la información procesada por las capas ocultas y la transforma en una salida interpretable. El número de neuronas en esta capa varía según la tarea: en un problema de clasificación binaria, suele haber una sola neurona que devuelve un valor entre 0 y 1, indicando la probabilidad de pertenecer a una clase u otra. En cambio, en una clasificación multiclase, la capa de salida tendrá tantas neuronas como clases posibles, y cada una devolverá una probabilidad asociada a una clase específica. Por último, como se ha mencionado anteriormente, se aplican funciones de activación específicas. [15].

2.4. Redes convolucionales

Las redes neuronales convolucionales, también conocidas como Convolutional Neural Networks (CNN o ConvNets), constituyen una arquitectura especializada de redes neuronales artificiales diseñada específicamente para el procesamiento de datos con estructura en forma de cuadrícula, como las imágenes. Para comprender cómo logra identificar patrones en una imagen, primero es importante tener en cuenta cómo se representa digitalmente una imagen. Esta representación numérica permite a la red aplicar operaciones matemáticas sobre la imagen para detectar características relevantes en distintos niveles de complejidad.

En esencia, una imagen digital se representa como una matriz de valores numéricos, donde cada valor corresponde a la intensidad de un píxel. Por ejemplo, una imagen en escala de grises puede representarse mediante una matriz cuyos elementos varían entre 0 y 255, siendo 0 el negro absoluto y 255 el blanco total. Esta estructura matricial permite que los filtros de la CNN puedan recorrer la imagen, identificando patrones para el aprendizaje del modelo.

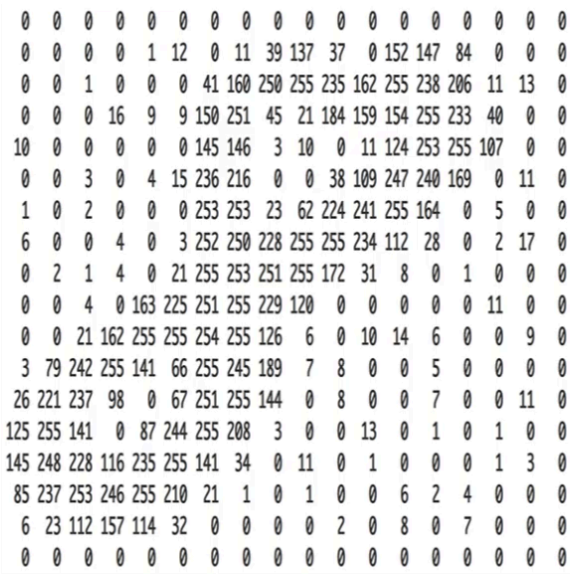


Figura 11: Representación de pixeles de una imagen digital [16].

En la Figura 11 se puede distinguir un ejemplo de una imagen que contiene el número 8, representada como una matriz con estos valores numéricos correspondientes a cada píxel. Así, este conjunto de datos digitales es el punto de partida para que una red convolucional procese la imagen. A partir de esta matriz, las CNN aplican convoluciones y operaciones de pooling para reconocer patrones y generalizar su aprendizaje a nuevas imágenes.

Estas redes han sido fundamentales en el desarrollo de la visión por computadora debido a su capacidad para extraer automáticamente características espaciales y jerárquicas a partir de los datos de entrada, sin requerir una ingeniería manual de características.

Su funcionamiento está inspirado en la corteza visual del cerebro humano. Cuando una persona observa una escena, su sistema visual no procesa la imagen en su totalidad de forma inmediata, sino que se activa mediante neuronas especializadas que responden a estímulos visuales específicos, como bordes, líneas o formas sencillas. De manera análoga, una red neuronal convolucional analiza una imagen siguiendo un proceso en cascada, donde en primer lugar se analizan los elementos más básicos o generales de una imagen, como los contornos o contrastes, que posteriormente se combinan para formar patrones más complejos. Por ejemplo, al reconocer la cara de una persona, primero se identifican rasgos simples como ojos, nariz y boca, que luego se integran para construir la imagen completa.

Si analizamos cómo una red neuronal convolucional simula este comportamiento, veremos que se basa en estudiar la estructura espacial de la imagen. Esto implica no solo considerar el valor individual de un píxel, sino también analizar la relación que este guarda con sus píxeles vecinos inmediatos. La información contextual que proporcionan estos píxeles vecinos es crucial porque las características visuales, como bordes, texturas y formas, no se definen por un solo punto, sino por un conjunto de puntos relacionados en el espacio. Por ejemplo, un borde se define por una transición abrupta entre píxeles con valores diferentes, y esta información solo se puede obtener si se considera la vecindad del píxel y no únicamente su valor aislado.

Para comprender cómo realizan su tarea, es importante destacar que las redes neuronales convolucionales se componen principalmente de tres tipos de capas. A continuación, se explicará la función de cada una de estas capas y cómo contribuyen al procesamiento y análisis de las imágenes dentro de la red:

- Capa convolucional: La capa convolucional es el componente fundamental y más complejo de una CNN. Su función principal es aplicar una operación matemática llamada convolución, que consiste en deslizar una pequeña ventana o filtro sobre la matriz de píxeles que representa una imagen. Este filtro, también conocido como kernel o núcleo, es una matriz pequeña de números (pesos) que actúa sobre una sección específica de la imagen. Su tamaño habitual es 3×3, aunque puede variar según el diseño de la red, y determina el área de la imagen que se analiza a la vez, conocido como campo receptivo.

Al colocar el kernel sobre una región de la imagen, se realiza un producto escalar entre los valores de los píxeles y los pesos del filtro. El resultado se guarda en una nueva matriz llamada mapa de características o mapa de activación, que indica dónde se detectaron ciertos patrones o características, de las mismas dimensiones que la imagen original. Esta operación se puede ver representada en la Figura 12 a continuación y se repetiría desplazando el filtro, hasta haber completado el recorrido por la imagen completa.

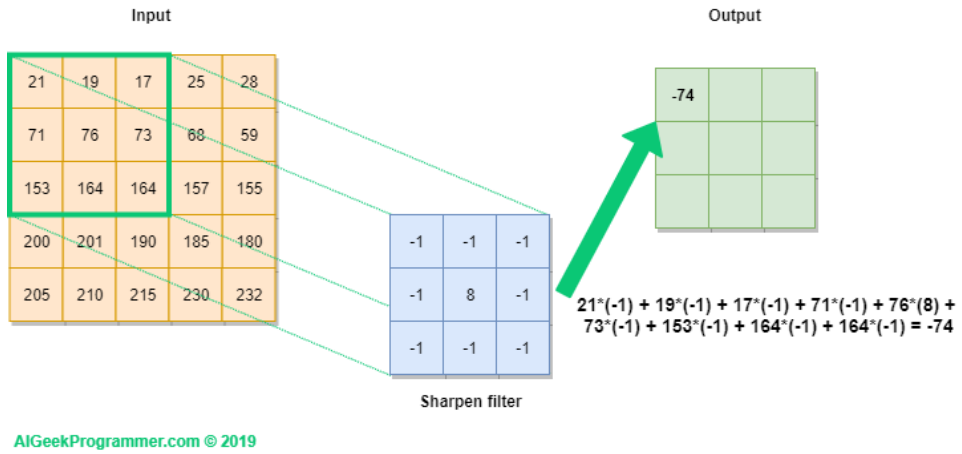


Figura 12: Ejemplo de Operación de Convolución [17].

Este proceso se repite desplazando el kernel por toda la imagen, cubriendo cada región posible hasta recorrerla completamente. Así, el mapa de características refleja la presencia y ubicación de los patrones que el filtro está diseñado para encontrar, como bordes, texturas o formas.

En imágenes a color, con tres canales (rojo, verde y azul), el filtro es tridimensional (por ejemplo, 3×3×3), con un kernel para cada canal. Cada kernel se aplica a su respectivo canal, y los resultados se combinan sumando sus valores junto con un término adicional llamado bias o sesgo, que ayuda a la red a ajustarse mejor a diferentes patrones [18].

Cada filtro funciona como una lupa especializada que resalta detalles específicos: uno puede detectar líneas rectas, otro curvas, y así sucesivamente. Al aplicar múltiples filtros, la red genera diversas representaciones de la imagen que evidencian la presencia y localización de distintos patrones. Por ejemplo, en la Figura 13 se muestra una posible transformación en una imagen que la enfoca y hace que se puedan detectar los bordes con más facilidad.

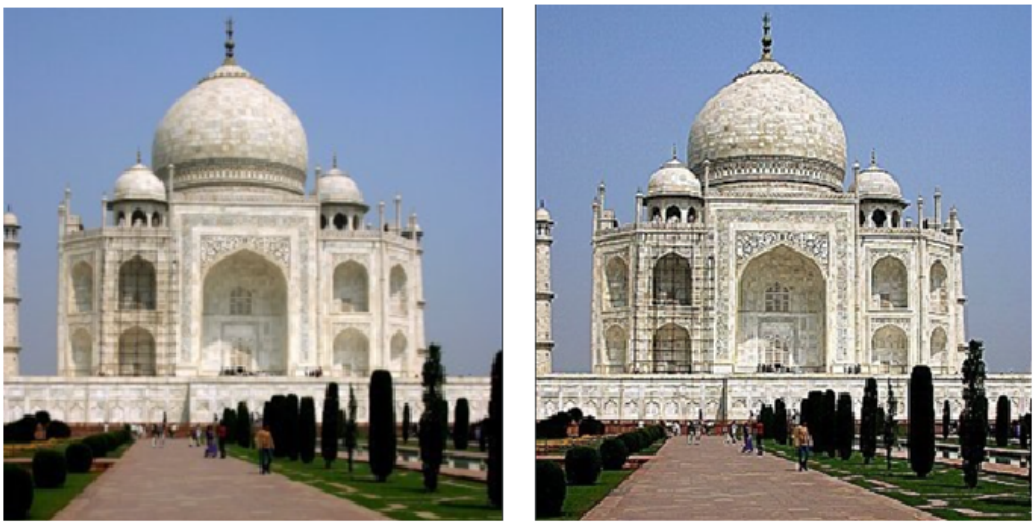


Figura 13: Ejemplo Imagen: Comparación antes y después de una operación de realce de bordes [19].

Por tanto, dos aspectos clave definen a las capas convolucionales: el número de filtros aplicados, que corresponde a la cantidad de patrones distintos que la red intenta captar; y el tamaño de estos filtros, usualmente matrices cuadradas pequeñas como 3×3 o 5×5 píxeles. Estos parámetros controlan la capacidad de la red para extraer características relevantes y construir representaciones cada vez más complejas a medida que avanza por las capas.

Por otra parte, la primera capa convolucional de una red CNN puede ir seguida de una o varias capas convolucionales adicionales. Cuando esto sucede, se establece una estructura jerárquica dentro de la red, ya que cada capa posterior analiza los llamados campos receptivos generados por las capas anteriores. En otras palabras, en lugar de trabajar directamente con los píxeles originales, las capas más profundas procesan patrones extraídos previamente, permitiendo una interpretación cada vez más abstracta y compleja de la imagen.

Por ejemplo, si deseamos identificar una bicicleta en una imagen, la red no busca el objeto completo desde el principio. Primero detecta elementos simples como bordes y esquinas; después, formas como círculos o líneas rectas que pueden corresponder a las ruedas o el cuadro; y finalmente, la combinación de estos elementos permite reconocer estructuras más complejas como el manillar, los pedales o el cuadro.

Esta división está esquematizada en la Figura 14. La suma de todos estos patrones de nivel inferior da lugar a una representación de nivel superior que permite a la red concluir que se trata de una bicicleta.

Figura 14: Proceso Jerárquico de Reconocimiento de Objetos en Redes Neuronales [18].

- Capa de agrupación:

  La función principal de la capa de agrupación, conocida también como submuestreo, es reducir el tamaño espacial de las características extraídas en una red neuronal convolucional. Esto se logra disminuyendo la cantidad de datos y parámetros que se procesan en las siguientes etapas, lo que ayuda a que el modelo sea más eficiente y menos propenso a sobreajustarse. A diferencia de una capa convolucional, la agrupación no aprende pesos ni parámetros, sino que utiliza un filtro fijo que se mueve sobre la entrada para resumir la información local. Este filtro realiza una operación de resumen o agregación sobre cada región que cubre, generando así una representación más compacta de la imagen o mapa de características original.

  Los métodos más comunes de agrupación son:

  - Max pooling (agrupación máxima): este método selecciona el valor más alto dentro del área cubierta por el filtro y lo transmite a la salida, como se muestra en la Figura 15. Es especialmente útil para resaltar las características más fuertes o activaciones más importantes en cada región.

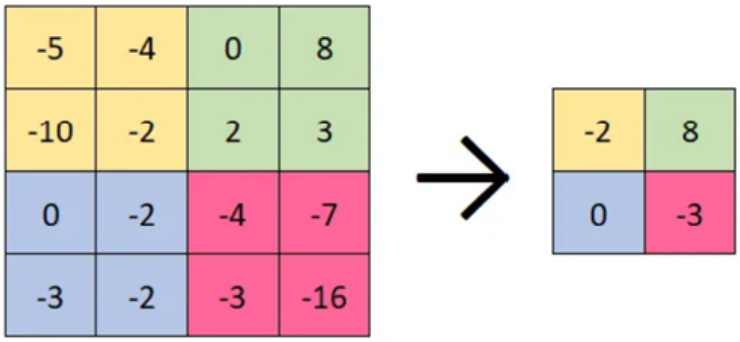


Figura 15: Ejemplo de Max Pooling sobre una Matriz de Características [20].

- Average pooling (agrupación promedio): aquí, el filtro calcula el valor promedio de los elementos dentro de su ventana, proporcionando una representación más suavizada y menos sensible a valores extremos, como se aprecia en la Figura 16.

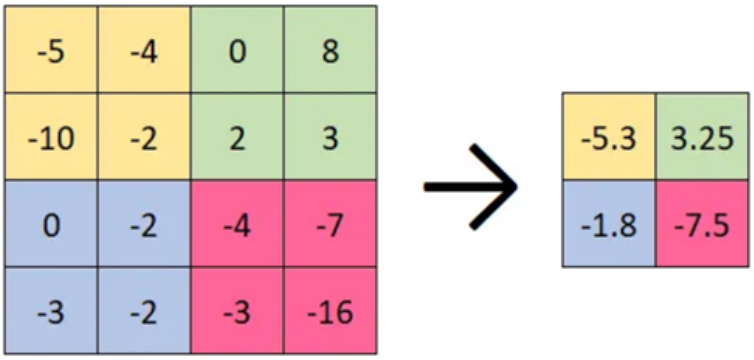


Figura 16: Ejemplo de Average Pooling sobre una Matriz de Características [20].

Aunque este proceso implica la pérdida de algunos detalles finos, la agrupación contribuye significativamente a simplificar el modelo, acelerando el entrenamiento y la inferencia. Además, al reducir la resolución espacial, también ayuda a disminuir el riesgo de sobreajuste, haciendo que la red generalice mejor a datos nuevos [18].

En conclusión, esta capa permite manejar de manera más eficiente la información, facilitando que la red se enfoque en las características más relevantes mientras controla la complejidad del modelo.

- Capa totalmente conectada: La capa totalmente conectada, también conocida como capa densa, es un tipo de capa en la que cada neurona de la capa de salida está vinculada directamente con todas las neuronas de la capa previa. Esto significa que, a diferencia de las capas convolucionales o de agrupación donde las conexiones son parciales o locales, aquí existe una conexión completa entre todas las unidades, similar a una red neuronal tradicional.

  Antes de alimentar esta capa con la salida proveniente de las capas anteriores, es necesario realizar un proceso llamado aplanamiento (flattening). Este consiste en transformar la salida tridimensional, que generalmente proviene de la última capa de agrupamiento, en un vector unidimensional. De esta forma, los datos pueden ser procesados correctamente.

  Su función principal es realizar la clasificación basándose en las características extraídas previamente por las capas convolucionales y de agrupación. Mientras que las capas anteriores suelen utilizar funciones de activación como ReLU para introducir no linealidad, las capas totalmente conectadas aplican primero funciones como ReLU para mantener la no linealidad y permitir que la red aprenda patrones complejos y finalmente utilizan la función softmax para generar probabilidades normalizadas que suman 1, facilitando la asignación de la etiqueta final al dato de entrada según la mayor probabilidad obtenida [18].

  En resumen, esta capa integra todas las características extraídas por la red para tomar la decisión final de clasificación, aprovechando la conexión completa entre neuronas para realizar una predicción robusta y precisa.

2.5. YOLO

YOLO (You Only Look Once) es un algoritmo de inteligencia artificial de código abierto ampliamente utilizado en tareas de detección de objetos. Su principal característica es que emplea una única red neuronal convolucional para analizar la imagen completa en una sola pasada, lo que lo convierte en una solución eficiente y rápida, ideal para aplicaciones en tiempo real.

A diferencia de otros enfoques que dividen el proceso en múltiples etapas (como propuestas de regiones o ventanas deslizantes), YOLO realiza simultáneamente la clasificación y localización de objetos. Esta arquitectura unificada le permite generalizar con precisión sobre objetos no vistos durante el entrenamiento, reduciendo significativamente el error en nuevas entradas.

El proceso de detección en YOLO se basa en tres componentes fundamentales:

- **Categoría del Objeto:** hace referencia a la clase del objeto que se desea identificar en una imagen. Por ejemplo, en un modelo entrenado para reconocer animales y vehículos, las categorías posibles podrían incluir *perro*, *bicicleta*, *automóvil*, entre otras. YOLO asigna una probabilidad a cada clase para cada región analizada, lo que permite determinar con qué certeza un objeto pertenece a una categoría específica.

- **Localización del Objeto:** indica la posición exacta del objeto dentro de la imagen. Para ello, YOLO utiliza las llamadas *cajas delimitadoras* (*bounding boxes*), que son rectángulos que enmarcan el área donde se predice que se encuentra el objeto y se representan como se muestra en la Figura 17 . Cada caja se define mediante cuatro parámetros:

  - x, y: coordenadas del punto superior izquierdo (o del centro) de la caja.
  - w, h: ancho y alto del rectángulo.

  Estas cajas permiten visualizar y cuantificar la ubicación de los objetos detectados, siendo fundamentales para tareas como seguimiento, segmentación o análisis espacial.

- **Probabilidad de la Predicción:** también conocida como *confidence score*, es el valor que YOLO asigna a cada detección para indicar el grado de certeza de que un objeto pertenece a una clase específica y está correctamente localizado. Esta puntuación se calcula combinando:

  - La probabilidad de que haya un objeto en la región analizada.
  - El grado de coincidencia entre la caja predicha y la caja real, medido mediante el *IoU (Intersection over Union)*, que se detallará a continuación.

Como ejemplo, en la Figura 18 se puede distinguir una imagen y las distintas detecciones que se han hecho sobre ella, reuniendo todos estos elementos.

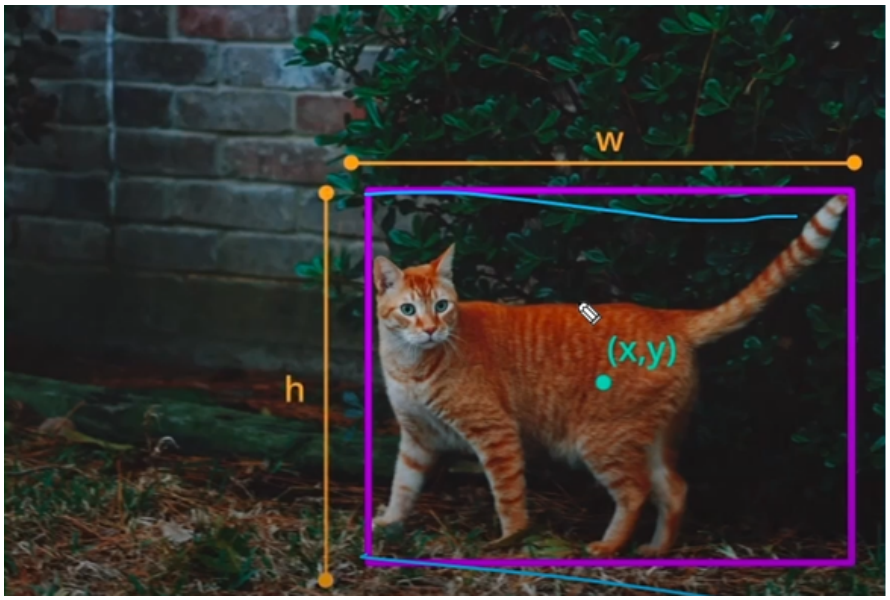


Figura 17: Representación de una Caja Delimitadora (Bounding Box) y Height y Width en Detección de Objetos [21].

Es importante conocer que el IoU es una métrica muy usada en modelos de detección de objetos, ya que permite cuantificar la precisión espacial de las predicciones. En lugar de limitarse a verificar si un objeto ha sido correctamente clasificado, evalúa el grado de solapamiento entre la región predicha por el modelo y la región real que ocupa el objeto en la imagen. Esta medida resulta especialmente útil en contextos donde la localización precisa es tan relevante como la clasificación en sí.

El valor de IoU se obtiene como el cociente entre el área de intersección (la zona común entre la predicción y la anotación real) y el área de unión (la superficie total cubierta por ambas regiones). El resultado es un valor normalizado entre 0 y 1, donde 1 indica una coincidencia perfecta y 0 implica ausencia total de solapamiento.

En modelos como YOLO, el IoU no solo se emplea como métrica de evaluación, sino que también interviene en el cálculo de la puntuación de confianza asociada a cada predicción. De este modo, se penalizan aquellas detecciones que, aunque acierten en la clase, presentan una localización deficiente [23].

Tras el paso de la imagen por la red convolucional, se obtiene una representación intermedia conocida como mapa de características, explicado anteriormente. A partir de esta representación, se genera un vector que resume las características más relevantes, que es procesado

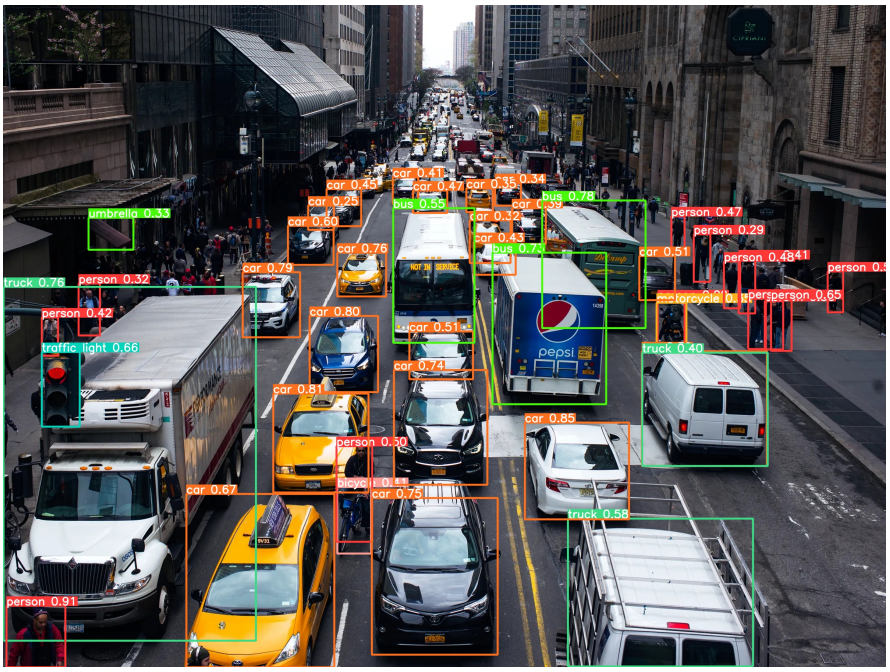


Figura 18: Ejemplo de Detección de Objetos con YOLO con Bounding Boxes, Nombre de la Clase y Confianza de la predicción [22].

por una capa densa que se encarga de asignar una probabilidad a cada clase posible, mientras que otra parte del modelo se ocupa de estimar la posición del objeto mediante una serie de regresiones. Estas regresiones permiten calcular las coordenadas necesarias para definir las cajas delimitadoras, es decir, los valores que indican la ubicación y el tamaño del objeto dentro de la imagen.

Este enfoque resulta eficaz cuando se trata de detectar un único objeto. Sin embargo, en la mayoría de los casos, las imágenes contienen múltiples elementos de diferentes clases y tamaños. Para abordar esta situación, YOLO divide la imagen en una cuadrícula de celdas, y cada una de ellas es responsable de realizar predicciones sobre los objetos cuyo centro se encuentra dentro de su área. Gracias a esta estrategia, el modelo puede realizar múltiples detecciones de forma simultánea, manteniendo la eficiencia y reduciendo el tiempo de procesamiento.

Antes de que surgieran modelos como YOLO, una técnica habitual para detectar varios objetos consistía en utilizar ventanas deslizantes. Este método implicaba recorrer la imagen con una ventana de tamaño fijo que se desplazaba por toda su superficie, generando múltiples recortes que eran analizados individualmente. Este enfoque suponía un elevado coste computacional, ya que requería evaluar una gran cantidad de regiones por imagen.

Para mejorar la eficiencia, se introdujeron métodos basados en propuestas de regiones. En lugar de analizar todas las posibles ubicaciones, este enfoque aplica una red convolucional sobre la imagen completa para identificar aquellas zonas que, por sus características visuales, tienen mayor probabilidad de contener un objeto. Estas regiones se seleccionan en función de patrones como cambios de intensidad, bordes definidos o formas geométricas particulares. Al centrarse únicamente en las áreas más prometedoras, se reduce considerablemente el número de regiones a procesar, lo que permite acelerar el análisis sin comprometer la precisión [21].

3

Tecnologías usadas

3.1. Python

Python es un lenguaje de programación potente y de alto nivel que destaca por su simplicidad y facilidad de aprendizaje. Su sintaxis clara y directa lo hace adecuado tanto para desarrolladores novatos como expertos. A pesar de su simplicidad, Python ofrece un sistema de programación orientada a objetos altamente efectivo y estructuras de datos de alto nivel, lo que lo convierte en una herramienta versátil para desarrollar desde scripts sencillos hasta aplicaciones complejas.

Una de sus grandes ventajas es la capacidad de adaptarse dinámicamente a los tipos de datos. Al ser un lenguaje interpretado, también facilita el desarrollo rápido de aplicaciones en diversos campos, incluyendo el análisis de datos y el desarrollo web, entre otros. Cuenta con un extenso ecosistema de bibliotecas y herramientas como NumPy para cálculos numéricos, Pandas para tratamiento de datos y matplotlib para creación de gráficos— que facilita tanto el análisis como la visualización de la información. Su gran comunidad activa contribuye constantemente con nuevas librerías, marcos de trabajo y recursos educativos [24].

Figura 19: Logo lenguaje de programación Python [25].

3.2. Pytorch

PyTorch es una biblioteca de código abierto desarrollada por el equipo de investigación de inteligencia artificial de Facebook (FAIR) en 2018. Diseñada principalmente para tareas de aprendizaje profundo y modelado numérico, PyTorch ha logrado un gran reconocimiento por su flexibilidad, facilidad de uso y capacidad de ejecución eficiente en GPUs.

Figura 20: Logo de PyTorch [26].

Una de las características más destacadas de PyTorch es su sistema de tensores, estructuras de datos fundamentales en esta biblioteca. La Figura 21 ilustra un ejemplo de tensor, que puede entenderse como una generalización de escalares (tensor de dimensión cero), vectores (tensores unidimensionales) y matrices (tensores bidimensionales), permitiendo representar datos en múltiples dimensiones. Por ejemplo, una imagen RGB puede representarse mediante un tensor tridimensional.

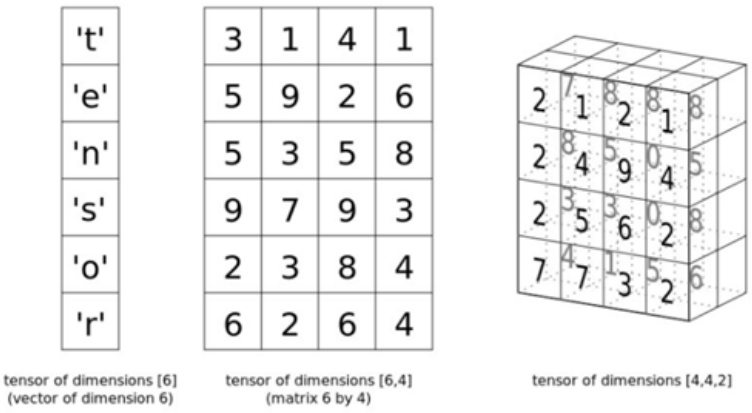


Figura 21: Ejemplo de Tensores [27].

Los tensores de PyTorch son similares a los ndarrays de NumPy, pero con la ventaja de poder ejecutarse tanto en CPU como en GPU, lo que incrementa considerablemente la velocidad de cálculo, especialmente en el entrenamiento de modelos complejos de redes neuronales.

Esta capacidad de procesamiento en paralelo mediante GPU resulta crucial al manejar grandes volúmenes de datos, como suele ser el caso en proyectos de Deep Learning.

Además, los tensores representan no solo los datos de entrada y salida de un modelo, sino también los parámetros internos como pesos y sesgos, permitiendo actualizaciones eficientes durante el aprendizaje. Destaca también la cantidad de bibliotecas de las que dispone, destacando Torchvision, que facilita la manipulación de datos de visión por computador, incluyendo carga de datos personalizados y transformaciones como redimensionado, normalización o rotación, esenciales para el preprocesamiento [28].

3.3. Google Colab

Google Colab es una plataforma gratuita de Google que permite ejecutar código Python en la nube utilizando recursos de computación de alto rendimiento como GPUs y TPUs. Basada en el concepto de cuadernos Jupyter, cuyo nombre proviene de los principales lenguajes de programación compatibles: Julia, Python y R, elimina la necesidad de configuraciones locales, permitiendo que se trabaje de manera ágil y accesible. Su funcionamiento se basa en escribir y ejecutar código Python en las celdas del cuaderno, de forma que el código se ejecuta en tiempo real en los servidores de Google y los resultados se muestran directamente debajo de las celdas.

Figura 22: Logo de Google Colab [29].

Asimismo, facilita mucho la tarea de crear gráficos o visualizaciones directamente en el cuaderno, ya que cuenta con gran soporte para ello. Otra ventaja es que ya cuenta con muchas bibliotecas populares de Python, como por ejemplo, NumPy, pandas y matplotlib. Por último, facilita almacenamiento, acceso y compartición de cuadernos y otros archivos de datos al integrarse nativamente con Google Drive [30].

3.4. COCO API

COCO API es una biblioteca escrita en Python desarrollada por el proyecto COCO (Common Objects in Context) que facilita la interacción con su conjunto de datos, ampliamente utilizado en tareas de visión por computador como detección de objetos, segmentación de instancias y puntos clave.

Esta API permite cargar y manipular fácilmente tanto las imágenes como sus anotaciones. A través de estas utilidades, es posible acceder de forma eficiente a los metadatos de cada imagen (categorías, coordenadas, máscaras, etc.), simplificando el procesamiento previo y la exploración del dataset.

Figura 23: Logo de Coco API [31].

De igual manera, ofrece herramientas para la visualización de resultados para representar gráficamente las anotaciones sobre las imágenes (cuadros delimitadores, etiquetas o segmentaciones), facilitando la validación visual de los datos o de las predicciones de un modelo.

Otro aspecto destacado es su sistema de evaluación. La clase COCOEval proporciona métricas estandarizadas como la Precisión Promedio (AP) y la Precisión Promedio Media (mAP), calculadas en distintos umbrales de Intersection over Union (IoU). Estas métricas son fundamentales para comparar de forma objetiva el rendimiento entre diferentes modelos de detección y segmentación.

Finalmente, también permite generar resultados en formatos compatibles para análisis posterior, y soporta diferentes tipos de anotaciones, como detección, segmentación o keypoints. Su versatilidad la convierte en una herramienta esencial para trabajos de investigación y desarrollo en visión artificial [31].

3.5. LaTeX

LaTeX es un sistema de composición de textos basado en el programa TeX, desarrollado inicialmente por Donald Knuth. Fue diseñado para la producción de documentos científicos y técnicos de alta calidad tipográfica y permite al autor concentrarse en el contenido mientras delega el formato al sistema. Es particularmente efectivo en la creación de documentos que requieren fórmulas matemáticas complejas, bibliografías extensas y una estructuración precisa del contenido. Su uso es muy común en disciplinas como física, matemáticas, informática e ingeniería.

Para facilitar el proceso de escritura y colaboración, se ha empleado Overleaf, un editor en línea de LaTeX que permite la edición colaborativa en tiempo real y la compilación automática de documentos. Overleaf ofrece una interfaz intuitiva y accesible desde cualquier dispositivo con conexión a Internet, eliminando la necesidad de instalar software adicional. Esta herramienta ha sido fundamental para mantener un flujo de trabajo eficiente y organizado durante la elaboración del proyecto.

Figura 24: Logo de LaTeX [32].

Figura 25: Logo de Overleaf [33].

Una de las principales ventajas de LaTeX es su sistema de paquetes extensibles, que permiten agregar funcionalidades como creación de gráficos, gestión automática de referencias, inclusión de presentaciones (Beamer) y generación de índices. A pesar de su curva de aprendizaje inicial, su potencia y flexibilidad lo hacen una herramienta insustituible para documentos de presentación [34].

3.6. YOLO Ultralytics

YOLO Ultralytics es una implementación avanzada del modelo de detección de objetos YOLO, desarrollada por la empresa Ultralytics, especializada en soluciones de visión por computador aplicadas a tareas como la detección y el seguimiento de objetos en imágenes y vídeo.

Figura 26: Logo de YOLO 11 (Ultralytics) [33].

La versión utilizada en este estudio es YOLO11, la más reciente dentro de la serie de modelos desarrollados por Ultralytics. Esta versión introduce mejoras tanto en la arquitectura del modelo como en los procesos de entrenamiento, lo que permite obtener resultados más precisos con un menor coste computacional. Además, se ha optimizado para funcionar en distintos entornos, desde dispositivos locales hasta plataformas en la nube.

YOLO11 está diseñado para abordar una variedad de tareas dentro del campo de la visión artificial. El modelo permite realizar tareas como la detección y el seguimiento de objetos, la segmentación de imágenes y la clasificación, algunas de ellas con capacidad de ejecución en tiempo real. En el contexto de este estudio, el enfoque se centrará exclusivamente en la detección de objetos en imágenes estáticas [33].

4

Algoritmos de mejora

El preprocesamiento de imágenes constituye una etapa fundamental dentro del análisis digital, ya que permite resaltar y organizar la información contenida en ellas antes de aplicar técnicas más complejas. A través de distintos métodos es posible ajustar aspectos como el brillo, el contraste o la distribución de los niveles de intensidad, lo que contribuye a que los detalles relevantes resulten más visibles y a que las imágenes adquieran una representación más adecuada para su interpretación. Estas transformaciones no solo facilitan la observación directa por parte de especialistas, sino que también ayudan a homogenizar las características de conjuntos de datos capturados bajo diferentes condiciones de iluminación o exposición.

A continuación, se presentan diversas técnicas de preprocesamiento, describiendo sus fundamentos matemáticos y el efecto que producen sobre las imágenes originales, junto con ejemplos de aplicación en distintos ámbitos.

4.1. Máscara de Enfoque (Unsharp Mask)

La máscara de enfoque o Unsharp Mask (USM) es una técnica utilizada para aumentar la nitidez de una imagen. Funciona creando una copia ligeramente desenfocada de la imagen original y combinándola con la original de manera que los bordes y detalles finos se destaquen más, haciendo que la imagen parezca más clara y definida. Su nombre proviene del hecho de que se utiliza una versión “no nítida” de la imagen para generar la máscara que refuerza los contornos.

En el procesamiento digital, este efecto se controla mediante parámetros que ajustan la intensidad del contraste en los bordes (amount), el tamaño de los bordes que se van a realzar (radius) y el umbral mínimo de diferencia de brillo necesario para aplicar el efecto (threshold).

Estos controles permiten reforzar los detalles deseados sin afectar zonas suaves de la imagen, ofreciendo flexibilidad en su aplicación y evitando la aparición de ruido no deseado [35].

La técnica digital consiste en aplicar un desenfoque gaussiano a una copia de la imagen original y restarla de la misma para generar la máscara. Esta máscara se suma nuevamente a la imagen original, resaltando los detalles de alta frecuencia y haciendo que los bordes y contornos se perciban más definidos. La Figura 27 muestra un ejemplo de aplicación de esta técnica: la imagen realzada presenta bordes más nítidos y detalles más claros en comparación con la original.

Figura 27: Ejemplo de aplicación de Unsharp Mask: imagen original vs. imagen realzada [36].

En resumen, la máscara de enfoque incrementa la nitidez aparente mediante la amplificación de las componentes de alta frecuencia de la imagen, convirtiéndose en una herramienta fundamental en el procesamiento digital de imágenes, tanto en fotografía como en otras áreas de visión por computadora.

4.2. Ecualización del Histograma (HE)

El histograma de una imagen es una representación gráfica en forma de diagrama de barras, donde el eje horizontal indica los distintos niveles de gris presentes, mientras que el eje vertical muestra la frecuencia relativa de aparición de cada nivel. Esta frecuencia se calcula como el cociente entre el número de píxeles con un determinado nivel de gris n_k y el número total de píxeles de la imagen n, es decir:

$$P_k = \frac{n_k}{n}.$$

La ecualización del histograma, también conocida como Histogram Equalization (HE), es una técnica de procesamiento digital de imágenes cuyo objetivo principal es mejorar el contraste general. Se basa en la premisa de que una imagen con una distribución uniforme de los niveles de intensidad de los píxeles presenta un contraste más alto que una cuya distribución está concentrada en un rango estrecho. Normalmente, el histograma representa la frecuencia de aparición de cada nivel de gris, comprendido entre 0 (negro) y 255 (blanco) en imágenes en escala de grises. Este método expande el rango dinámico de la imagen, redistribuyendo los niveles de gris de forma más uniforme mediante la función de distribución acumulativa del histograma original. Así, las zonas más oscuras tienden a oscurecerse más y las más claras a aclararse, realzando detalles que podrían permanecer ocultos en condiciones de bajo contraste. Como resultado, se mejora el contraste global de la imagen y se facilita su interpretación, tanto por observadores humanos como por algoritmos de procesamiento automático.

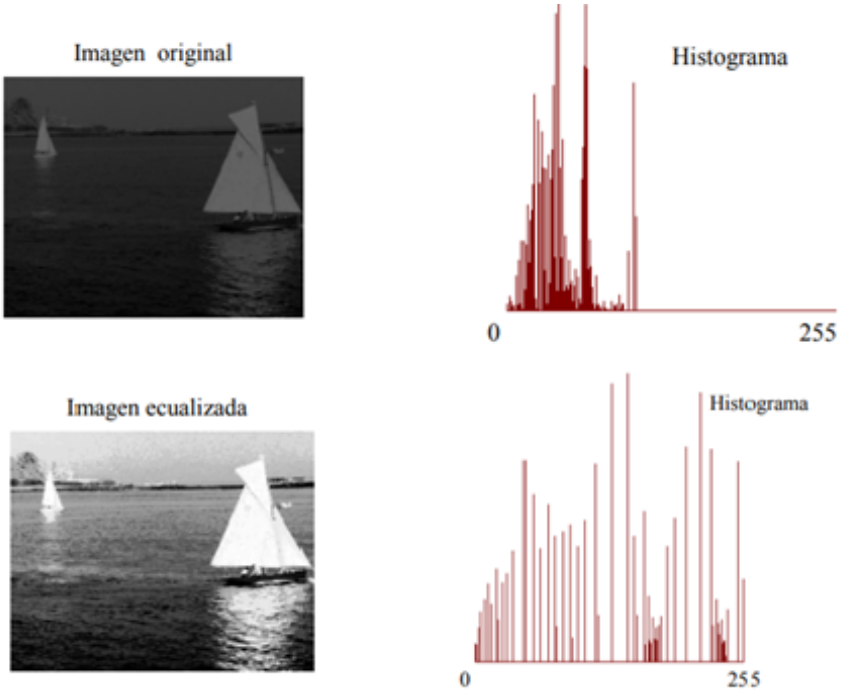


Figura 28: Comparación entre imagen original y ecualizada junto a sus histogramas [37].

Esta técnica se aplica en muchos ámbitos, como la medicina, la teledetección o la fotografía. Por ejemplo, en imágenes médicas permite resaltar ciertas estructuras internas en radiografías o resonancias, facilitando el diagnóstico y en teledetección se utiliza para mejorar la claridad de las imágenes satelitales, haciendo más visibles los detalles del terreno [38].

4.3. Ecualización Adaptativa Limitada por Contraste (CLAHE)

La ecualización adaptativa limitada por contraste (CLAHE) es una técnica utilizada para mejorar el contraste local de las imágenes, optimizando la visibilidad de detalles que podrían perderse en regiones muy claras u oscuras. A diferencia de la ecualización del histograma convencional, que aplica la misma transformación a todos los píxeles, CLAHE analiza cada píxel considerando únicamente los valores de su vecindad inmediata. Esto permite que las estructuras finas de la imagen se destaquen, facilitando la interpretación de información crítica sin comprometer otras áreas.

La amplificación del contraste se limita a un valor máximo predefinido, lo que evita la exageración de pequeñas variaciones de intensidad y reduce la aparición de ruido en zonas homogéneas. Además, CLAHE ofrece un control flexible sobre el tamaño de las regiones vecinas y el nivel máximo de contraste, permitiendo ajustar la técnica según las características específicas de cada imagen, como el nivel de iluminación o la presencia de detalles sutiles. Esta adaptabilidad hace que la técnica sea útil en entornos con condiciones de iluminación desigual y en imágenes donde se requiere resaltar tanto las zonas claras como las oscuras de manera equilibrada.

Cada píxel se ajusta según el histograma de un pequeño bloque cuadrado que lo rodea, y los píxeles situados en los bordes reciben un tratamiento especial mediante la extensión de la imagen, duplicando filas y columnas para evitar picos pronunciados en los histogramas locales. De esta manera, CLAHE garantiza que el contraste sea uniforme en toda la imagen y que los bordes y formas se definan con mayor claridad, sin generar artefactos visuales que dificulten el análisis.

La técnica se emplea ampliamente en campos como la medicina, para mejorar la visualización de estructuras en radiografías y resonancias, y en teledetección, donde la detección de detalles locales en imágenes satelitales es fundamental. CLAHE también puede integrarse con otras técnicas de preprocesamiento, como filtrado de ruido o normalización, para potenciar aún más la calidad de la información visual y facilitar la extracción de características relevantes para análisis automáticos. Por ejemplo, la Figura 29 muestra una comparación entre imágenes originales y procesadas con CLAHE, evidenciando cómo esta técnica mejora la visualización de detalles mediante el aumento del contraste local [39].

Figura 29: Comparación entre imágenes originales y procesadas con CLAHE, mostrando la mejora del contraste local [39].

4.4. Transformación Logarítmica

La transformación logarítmica, abreviada como LT en inglés, es una técnica que ajusta la escala de los datos aplicando el logaritmo a los valores originales. Cada valor x se reemplaza por $\log(x)$, siendo la base del logaritmo una decisión del analista (comúnmente base 10, base 2 o logaritmo natural). Esta transformación es especialmente útil para reducir la variabilidad extrema de los datos y facilitar su interpretación cuando los valores presentan una distribución sesgada o con un rango muy amplio. Al aplicar esta técnica, los valores bajos se expanden mientras que los valores altos se comprimen, permitiendo que los detalles presentes en zonas oscuras de la imagen se vuelvan más visibles sin saturar las áreas brillantes. Esto facilita la detección de patrones y estructuras que podrían pasar desapercibidas en la imagen original.

Como se aprecia en la Figura 30, los valores de intensidad en la imagen original se concentran mayoritariamente en el extremo izquierdo del histograma, indicando predominancia de píxeles oscuros. Tras aplicar la transformación logarítmica, la distribución de los valores se estira y redistribuye, facilitando un contraste más equilibrado y mejorando la visibilidad de detalles en sombras y áreas poco iluminadas. Este proceso ayuda también a minimizar la influencia de valores atípicos, permitiendo que las características relevantes de la imagen se destaquen de manera más clara [40]. Sin embargo, la técnica debe aplicarse con precaución: si la relación relativa entre valores es crítica para el análisis, la transformación logarítmica puede distorsionarla, y no es adecuada para resaltar valores atípicos o extremos.

Otros ejemplos del uso de transformaciones logarítmicas se muestran en las Figura 31 y Figura 32, aplicando la técnica a imágenes con iluminación diferente, una oscura y otra clara, respectivamente. Esto permite observar cómo la transformación adapta el contraste según la intensidad predominante de la escena, mejorando la visibilidad de detalles que podrían no ser perceptibles en las imágenes originales.

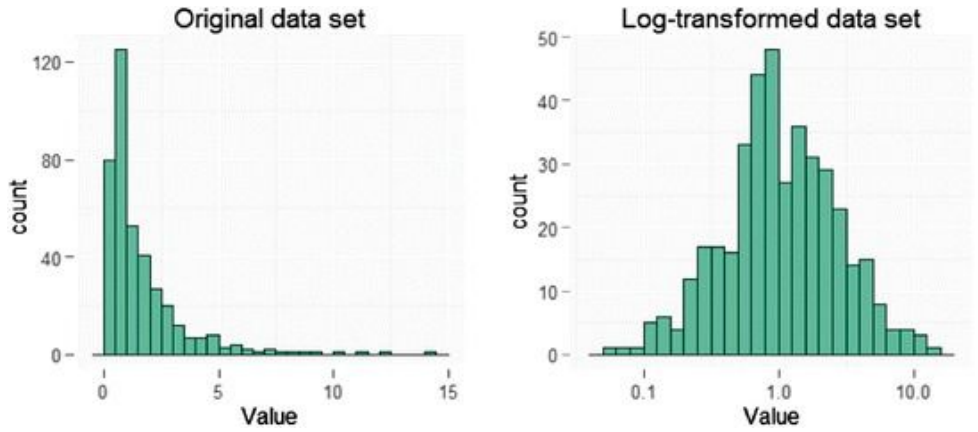


Figura 30: Comparación entre el histograma del conjunto de datos original y el transformado con logaritmo [41].

Figura 31: Aplicación de la Transformación Logarítmica a una imagen oscura [42].

Aunque la transformación logarítmica puede ser muy útil, no siempre es adecuada para todas las imágenes. Puede provocar pérdida de detalles en áreas con valores altos, ya que comprime estas regiones. Por esta razón, resulta más efectiva en imágenes donde predominan los valores bajos (zonas oscuras) sobre los altos (zonas claras), es decir, en situaciones de bajo contraste o iluminación deficiente.

Figura 32: Aplicación de la Transformación Logarítmica a una imagen clara [42].

Frecuentemente se combina con otras técnicas de ecualización, como HE o CLAHE, para optimizar el contraste global, equilibrar la iluminación de la imagen y resaltar detalles relevantes de manera uniforme, mejorando así la calidad visual y la interpretación de la información contenida en la imagen.

5
Metodología

El desarrollo de este Trabajo de Fin de Grado ha seguido una metodología mixta, basada en dos pilares fundamentales: por un lado, el método científico, como estructura de investigación orientada a la validación de hipótesis mediante la experimentación controlada; y por otro, un modelo de desarrollo iterativo e incremental, que ha permitido abordar la parte técnica de forma progresiva, flexible y escalable. Esta combinación ha facilitado una evolución continua del proyecto, promoviendo tanto la calidad del análisis como la mejora constante del sistema.

5.1. Enfoque científico

El método científico es un proceso sistemático y riguroso para generar conocimiento confiable. Su objetivo es comprender fenómenos mediante la observación, la formulación de preguntas y la construcción de explicaciones fundamentadas en evidencia verificable. A diferencia de una aproximación intuitiva o empírica, este enfoque exige que las conclusiones se basen en datos reproducibles y en experimentación controlada, lo que permite diferenciar entre resultados accidentales y relaciones causales reales [43].

En el contexto de este proyecto, el método científico se aplica para plantear hipótesis claras, ejecutar experimentos controlados y analizar los resultados de manera objetiva, asegurando que las conclusiones sean fundamentadas y replicables.

El proceso se estructura en varias fases interrelacionadas:

- **Observación y planteamiento del problema:** se identifican los factores que podrían influir en la precisión del modelo y se define el objetivo de estudio.
- **Formulación de hipótesis:** se establece una predicción sobre cómo los cambios en las imágenes (por ejemplo, disminución de brillo o aplicación de algoritmos de mejora) podrían afectar los resultados del modelo.

- **Diseño experimental:** se planifica cómo manipular las imágenes, ejecutar las detecciones y cómo medir los efectos sobre el desempeño del modelo.
- **Ejecución de experimentos:** se aplican las transformaciones a los conjuntos de imágenes y se registran los resultados de las predicciones del modelo.
- **Análisis de resultados:** se utilizan métricas cuantitativas y comparaciones visuales para evaluar la validez de las hipótesis.
- **Conclusiones y refinamiento:** los hallazgos permiten ajustar las hipótesis o plantear nuevos experimentos, siguiendo un ciclo de mejora continua.

En síntesis, este enfoque convierte al proyecto en un proceso de investigación estructurado y reproducible, donde cada iteración y cada prueba aportan evidencia científica sobre el comportamiento del modelo frente a diferentes condiciones de las imágenes. Esto no solo garantiza resultados fiables, sino que también permite explorar de manera sistemática posibles mejoras y optimizaciones del sistema.

5.2. Metodología incremental e iterativa

Desde el punto de vista del desarrollo de software, se ha optado por una metodología incremental e iterativa, alineada con principios de desarrollo ágil. En lugar de construir el sistema de manera monolítica, se ha desarrollado en bloques funcionales sucesivos (iteraciones), donde cada uno añadía nuevas capacidades, corregía errores detectados o refinaba procedimientos anteriores. Esto ha facilitado una retroalimentación constante, permitiendo evaluar el impacto de cada mejora y aprender de cada iteración para la siguiente [44].

Cada fase del ciclo metodológico mostrado en la Figura 33 se completó de manera autónoma, construyendo el proyecto de forma progresiva. Este enfoque iterativo permitió que cada etapa añadiera nuevas capacidades o refinara procedimientos previos sin afectar el trabajo realizado en fases anteriores, asegurando un desarrollo ordenado y una mejora continua a lo largo de todo el proceso.

Durante la fase de Análisis y Diseño se definieron los elementos esenciales del sistema, incluyendo los módulos de preprocesamiento de imágenes, los algoritmos de mejora y los procedimientos para evaluar resultados. En la etapa de Implementación, estos componentes

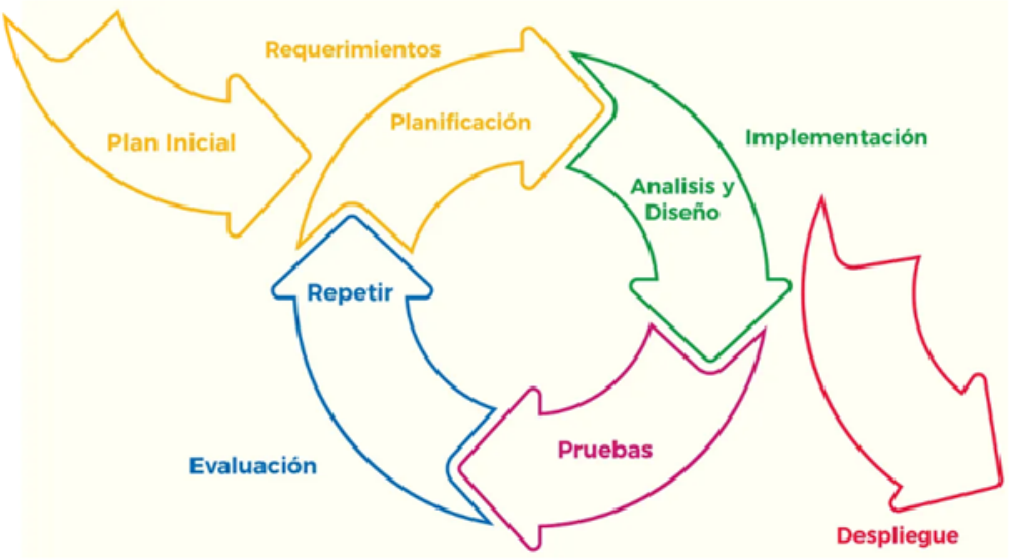


Figura 33: Diagrama metodología iterativa con sus distintas fases [45].

se construyeron de manera flexible y modular, lo que permitió probar distintas configuraciones y ajustes sin afectar las funcionalidades ya desarrolladas. La fase de experimentación se centró en ejecutar diferentes pruebas sobre cada módulo de forma independiente, asegurando la consistencia y replicabilidad de los resultados. Finalmente, en la fase de Evaluación, se analizaron los datos obtenidos en cada iteración, identificando mejoras concretas que se incorporaron en el siguiente ciclo de desarrollo, cerrando así un proceso iterativo de mejora continua y repitiendo este proceso para incorporar aquellas mejoras detectadas.

Este enfoque iterativo permite una evolución constante del proyecto, con ajustes progresivos basados en la evaluación continua de los resultados obtenidos en cada etapa.

5.3. Iteraciones del desarrollo

A continuación, se presenta una descripción resumida de las iteraciones realizadas en el proyecto:

- **Iteración 1:** Se implementaron las funcionalidades básicas, incluyendo la importación de datos y la detección inicial de objetos con YOLO. Para la validación inicial, se visualizaron directamente las imágenes con las cajas delimitadoras y etiquetas de las clases detectadas, estableciendo así una línea base de referencia.

- **Iteración 2:** Se desarrolló un sistema de evaluación cuantitativa de las predicciones, obteniendo métricas relevantes como precisión y recall proporcionadas por COCOVal, distinguidas por el tamaño de los objetos detectados. Esto permitió medir objetivamente el rendimiento del modelo e identificar debilidades para ajustes posteriores.

- **Iteración 3:** Se modificaron las imágenes originales para simular condiciones de baja iluminación mediante la reducción del brillo y se evaluaron los resultados.

- **Iteración 4:** Se incorporaron distintos algoritmos de mejora de imagen para contrarrestar los efectos de la baja iluminación. Posteriormente, se repitieron las pruebas de detección y evaluación, comparando los resultados con los obtenidos en iteraciones previas.

- **Iteración 5:** Se implementaron métodos avanzados de visualización, mostrando tanto las predicciones del modelo como las etiquetas reales de los objetos. Esto permitió un análisis cualitativo más profundo de los resultados y facilitó la identificación de patrones de error.

- **Iteración 6:** Se centró en la consolidación y presentación de resultados mediante mapas de calor y registros automáticos de todas las pruebas. Esto permitió almacenar los datos de manera organizada y comparar los resultados de distintas iteraciones, facilitando la toma de decisiones para ajustes finales.

Durante el desarrollo del proyecto, los flujos de datos se organizaron mediante dataloaders, estructuras encargadas de gestionar los distintos conjuntos de imágenes generados: originales, con brillo reducido y mejoradas mediante algoritmos de mejora. Los dataloaders automatizan tareas como la lectura de archivos, el almacenamiento temporal en memoria y la preparación de lotes de imágenes para los experimentos, garantizando trazabilidad, replicabilidad y eficiencia en el manejo de grandes volúmenes de datos. Dado que algunas imágenes del conjunto presentaban tamaños distintos, se redimensionaron a un tamaño estándar como parte del preprocesamiento, lo que permitió un manejo más sencillo y uniforme dentro de los dataloaders, que requieren dimensiones uniformes entre las imágenes para funcionar correctamente. Esto asegura un flujo de datos controlado y consistente, facilitando la gestión y el seguimiento de cada conjunto de imágenes durante todo el proyecto.

Además, se implementaron transformers modulares para aplicar de manera eficiente distintas transformaciones sobre las imágenes. Estos bloques permiten ajustar parámetros específicos, como el brillo o la aplicación de algoritmos de mejora, de forma flexible e independiente. La modularidad de los transformers evita duplicación de código y facilita la experimentación con distintas configuraciones durante las iteraciones, contribuyendo a un flujo de preprocesamiento ordenado y escalable.

Para asegurar que todo el sistema sea mantenible y adaptable, se aplicaron buenas prácticas de ingeniería de software, incluyendo la separación de responsabilidades, la documentación clara del código y la parametrización de los métodos. De este modo, cada módulo cumple una función específica y puede adaptarse fácilmente a distintos escenarios o configuraciones, lo que facilita la comprensión, el uso y la reutilización del código por otros desarrolladores.

En conjunto, esta organización permitió optimizar los tiempos de ejecución, garantizar la reproducibilidad de los experimentos y establecer una base sólida para futuras ampliaciones o desarrollos del proyecto, asegurando un manejo eficiente y confiable de los datos e imágenes.

5.4. Justificación del enfoque adoptado

La elección de una metodología iterativa y basada en el método científico responde tanto a la naturaleza exploratoria del problema como a los objetivos del proyecto. En el contexto del reconocimiento visual con condiciones de iluminación no controladas, es crucial contar con un enfoque flexible que permita adaptarse a lo observado y evaluar soluciones de forma empírica.

Asimismo, la escalabilidad del código, su capacidad de adaptación y la parametrización de su diseño hacen que el proyecto no sea una solución cerrada, sino una base de experimentación sólida, sobre la cual se pueden construir futuros desarrollos o pruebas incluso más variadas.

5.5. Flujo experimental del proyecto

Para mayor claridad, en la Figura 34 se presenta un esquema que resume de manera gráfica el procedimiento seguido en este estudio. El flujo comienza con la imagen original, sin modificaciones de brillo, la cual se procesa con la red YOLO para obtener una primera referencia sobre su capacidad de detección y el grado de precisión alcanzado.

Posteriormente, la imagen se somete a una reducción de brillo con el fin de simular condiciones de baja luminosidad y analizar cómo evoluciona el rendimiento del modelo en un entorno menos favorable. Esta versión modificada se vuelve a procesar con la red, incorporando los resultados obtenidos al flujo de análisis.

Finalmente, a la imagen oscurecida se le aplican distintos algoritmos de mejora de imagen. El objetivo de este paso es evaluar su efectividad para contrarrestar la pérdida de información provocada por la baja iluminación. Este último experimento constituye la etapa clave del proceso, ya que concentra todos los elementos analizados previamente y permite comparar de manera sistemática los efectos de cada transformación.

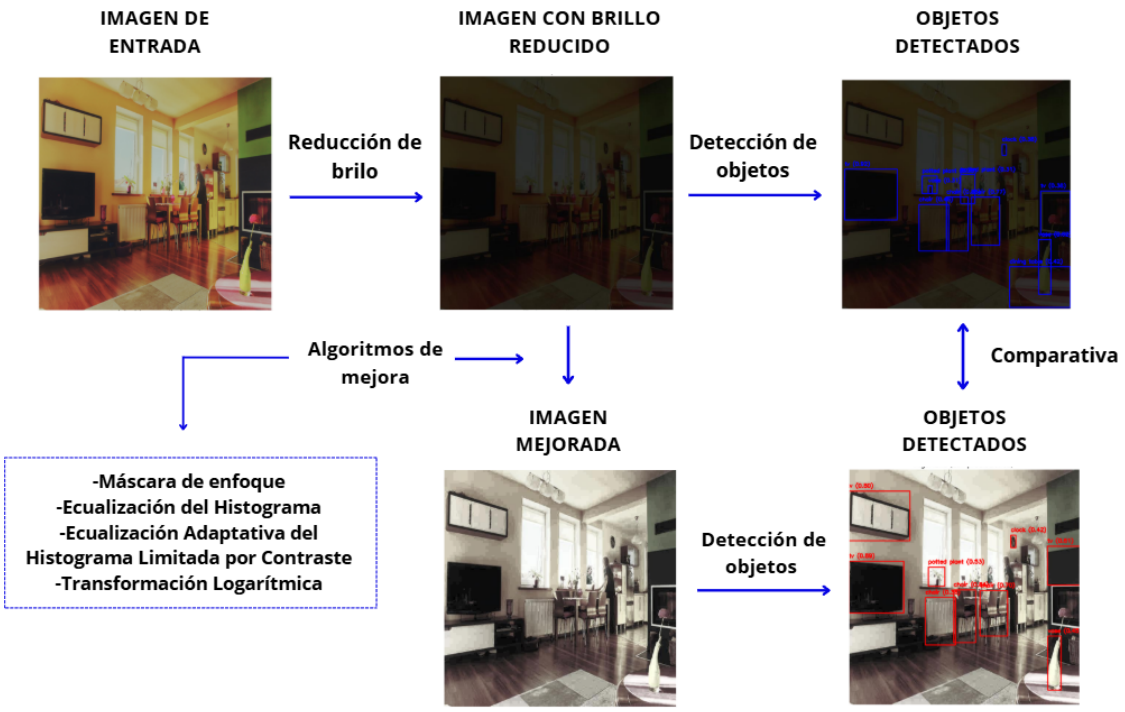


Figura 34: Esquema del proceso seguido para el tratamiento y análisis de imágenes en el proyecto.

Este proceso queda resumido en el diagrama de secuencia representado en la Figura 35. El flujo comienza con el usuario, quien define la configuración del experimento, incluyendo el modelo YOLO a utilizar, el nivel de brillo y los algoritmos de mejora de imagen a aplicar. Estas decisiones son enviadas al sistema, que actúa como controlador central, gestionando las solicitudes y procesando las respuestas de los distintos componentes.

A continuación, el sistema envía la imagen original al modelo YOLO, obteniendo una primera referencia de las predicciones sobre los objetos presentes. Posteriormente, el sistema aplica la etapa de brillo, donde la imagen se degrada según el nivel seleccionado para simular condiciones de baja luminosidad, y devuelve la imagen oscurecida al sistema. Esta versión de la imagen se procesa de nuevo por el modelo YOLO para evaluar cómo afecta la reducción de brillo al rendimiento de detección.

En la fase siguiente, el sistema aplica el algoritmo de mejora sobre la imagen oscurecida. Este componente recibe la imagen, los algoritmos seleccionados y devuelve la imagen mejorada. Finalmente, la imagen mejorada es enviada nuevamente al modelo YOLO, obteniéndose las predicciones correspondientes.

El sistema recopila todas las predicciones generadas en las tres etapas (original, oscurecida y mejorada) y las procesa para entregar al usuario las métricas finales y diversos resultados visuales, permitiendo comparar de manera sistemática los efectos de cada transformación sobre la capacidad de detección del modelo.

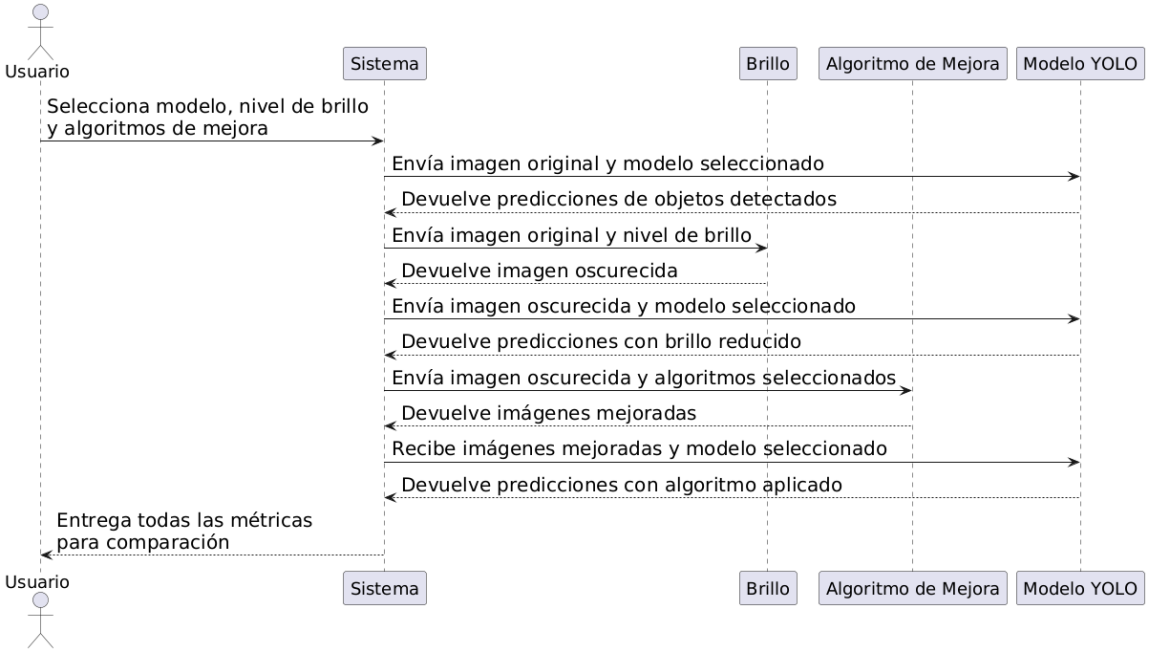


Figura 35: Diagrama de secuencia sobre el proceso seguido.

6
Resultados

6.1. Métodos utilizados

En los distintos experimentos realizados se evaluaron combinaciones de tres factores para analizar su impacto en la detección con YOLOv11: el tamaño del modelo, la alteración del brillo en las imágenes y la aplicación de algoritmos de mejora. Los tamaños de modelo considerados fueron n, s, m, l y x, que representan versiones de complejidad creciente. Los modelos más grandes suelen tener mayor capacidad para aprender características complejas, aunque también pueden ser más sensibles a variaciones en las condiciones de entrada.

Para cada tamaño de modelo se realizaron predicciones sobre el conjunto de datos original sin modificaciones y sobre imágenes con brillo alterado, ya sea mediante un valor fijo o de manera aleatoria. Siguiendo la metodología propuesta en [46], consideremos que I representa la matriz de píxeles de una imagen digital bajo condiciones de iluminación óptimas. Para simular condiciones de baja iluminación, cada píxel se ajusta mediante un factor de escala de brillo positivo b, de manera que el nuevo valor de píxel I' se calcula como:

$$I' = \frac{I}{b}$$

Esta operación genera imágenes más oscuras a medida que aumenta b, siendo $b = 1$ la imagen original sin alteración.

Las configuraciones de brillo utilizadas en los experimentos fueron:

- b = 1: imagen sin alteración de brillo (original).
- b = 3: mínima diferencia de rendimiento respecto a otros valores.
- b = 5: condiciones de iluminación baja moderada.
- b = 10: condiciones de iluminación muy bajas (imagen prácticamente negra).

- b ∈ {3,4,5} aleatorio: simula condiciones típicas de baja iluminación de forma aleatoria para cada imagen.

Por ejemplo, para una imagen original i, se generó una versión con brillo reducido i' usando $b = 3$, otra con $b = 5$, otra con $b = 10$ y una versión aleatoria con b elegido uniformemente entre 3, 4 o 5. Esto permitió evaluar la robustez del modelo frente a distintos niveles de iluminación.

Posteriormente, se aplicaron diferentes algoritmos de mejora sobre las imágenes con brillo reducido, obteniendo imágenes i'^*, que fueron procesadas nuevamente con YOLOv11 para registrar su rendimiento y comprobar si la mejora aplicada puede compensar la pérdida de calidad causada por la reducción de brillo.

De esta manera, se analizaron las predicciones en las distintas combinaciones de tamaño de modelo, nivel de brillo y algoritmo aplicado, permitiendo observar cómo cada factor influye en el desempeño general y en la robustez del sistema ante variaciones en la iluminación y en el procesamiento de las imágenes.

6.2. Conjunto de datos

El estudio utilizará el conjunto de datos COCO 2017 [31], ampliamente empleado para la evaluación de modelos de visión por computador. Este dispone de un total de unas 330.000 imágenes, de las cuales 200.000 pueden ser usadas para detección de objetos, debido a que contienen anotaciones tales como cuadros delimitadores de objetos, máscaras de segmentación y leyendas para cada imagen. COCO abarca un total de 80 categorías de objetos que incluyen tanto elementos comunes de la vida cotidiana, como vehículos y animales, como otros más específicos, tales como bolsos o paraguas. Esta diversidad de clases y escenarios proporciona un entorno ideal para evaluar el rendimiento de YOLO en la detección de objetos bajo distintas condiciones de iluminación y en contextos variados.

Aunque COCO está dividido en tres subconjuntos: entrenamiento, validación y pruebas, este estudio se centrará únicamente en el subconjunto de validación. El conjunto de entrenamiento ha sido utilizado para obtener los modelos preentrenados, mientras que el conjunto de pruebas no incluye anotaciones (máscaras de verdad), lo que impide su uso para evaluar el rendimiento del modelo.

6.3. Métricas

6.3.1. Métricas de rendimiento absoluto

Para evaluar el rendimiento de los modelos se emplearon las métricas estándar del conjunto COCO (Common Objects in Context) [31], que permiten medir tanto la precisión como la capacidad de detección de manera robusta y consistente. Estas métricas consideran tanto la localización como la cobertura de los objetos, proporcionando información detallada sobre la calidad de las predicciones.

Una de las métricas utilizadas es la precisión media o Average Precision (AP). El AP se obtiene a partir de la precisión promedio calculada en distintos niveles de recall, abarcando valores entre 0 y 1. Para comprender su cálculo es necesario introducir el concepto de Intersection over Union (IoU), que mide la superposición entre el área predicha por el modelo y la anotación verdadera de un objeto [47]. La Figura 36 ilustra este concepto.

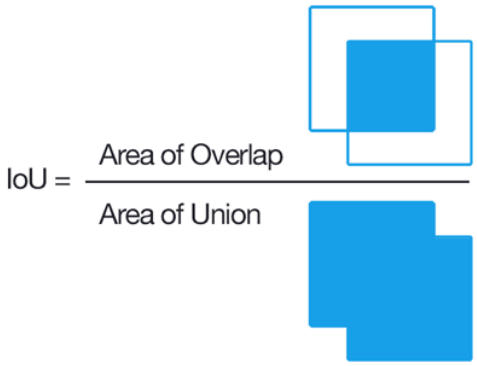


Figura 36: Esquema del cálculo del IoU, mostrando la intersección y la unión entre predicción y anotación [47].

Un valor de IoU cercano a 1 indica una coincidencia casi perfecta, mientras que valores próximos a 0 reflejan escaso o nulo solapamiento, como se muestra en la Figura 37. Este índice se emplea como criterio para decidir si una predicción se considera correcta o no, y también influye en la puntuación de confianza que recibe cada detección.

A partir de este criterio se calculan las métricas de precisión (P) y recall (R), basadas en los valores de verdaderos positivos (TP), falsos positivos (FP) y falsos negativos (FN), tal como se muestra en la Figura 38. En el caso de la detección de objetos no se consideran los verdaderos negativos (TN), ya que no se evalúa la ausencia de objetos en regiones vacías.

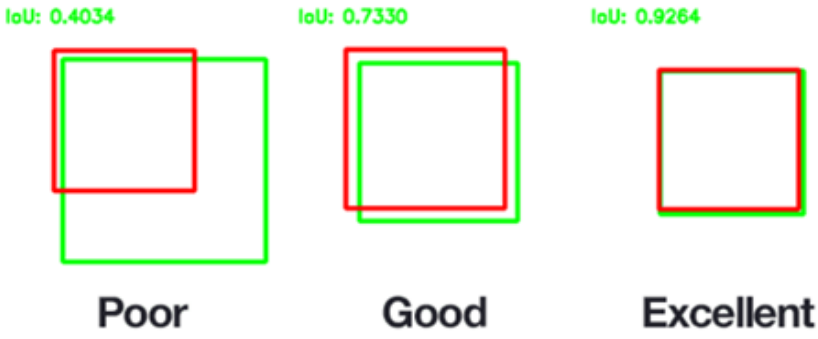


Figura 37: Ejemplos de superposición alta y baja entre predicción y anotación [47].

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

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

TP = True positive

TN = True negative

FP = False positive

FN = False negative

Figura 38: Fórmulas de precisión (P) y recall (R) basadas en TP, FP y FN [47].

Una predicción se clasifica como verdadero positivo si su IoU supera un umbral predefinido, de lo contrario, se considera falso positivo. En el protocolo de evaluación de COCO este umbral varía entre 0.50 y 0.95 en intervalos de 0.05, lo que permite medir la capacidad del modelo para localizar objetos con diferentes niveles de exigencia.

Entonces, el cálculo del AP se realiza promediando la precisión obtenida en distintos niveles de IoU para cada clase de objeto. De manera agregada, la mean Average Precision (mAP) corresponde al promedio de los valores de AP obtenidos en todas las clases del conjunto de datos.

De forma complementaria, el Average Recall (AR) indica el porcentaje de objetos presentes en la imagen que fueron correctamente detectados por el modelo. Al igual que en el caso del AP, se define una métrica agregada denominada mean Average Recall (mAR), calculada como el promedio del AR considerando objetos de distinto tamaño. En este contexto los objetos considerados pueden ser pequeños, medianos o grandes, dependiendo del área que ocupen en la superficie de la imagen.

Modelos como YOLOv11 permiten calcular automáticamente todas estas métricas, generando informes detallados que incluyen AP, AR, mAP y mAR tanto por clase como por tamaño de objeto. Este conjunto de indicadores facilita la comparación entre distintas configuraciones experimentales y proporciona una visión completa del rendimiento de cada modelo.

En resumen, el IoU, la precisión y el recall constituyen la base de la evaluación individual de cada predicción, mientras que las métricas agregadas AP, AR, mAP y mAR permiten analizar de manera global la capacidad de los modelos para detectar y localizar objetos.

6.3.2. Métricas de rendimiento relativo del uso de algoritmo de mejora con respecto a su no uso

Además de las métricas estándar del protocolo COCO, en este trabajo se proponen indicadores adicionales con el objetivo de analizar de manera más específica el impacto que tienen los algoritmos de mejora de imágenes bajo distintas configuraciones de brillo. Estas métricas permiten cuantificar no solo el rendimiento global del modelo, sino también el efecto diferencial que supone aplicar un algoritmo de preprocesamiento concreto frente a no aplicarlo.

- **Balance de impacto**: para un algoritmo y una configuración de brillo b, corresponde a la media de las diferencias de rendimiento (AP) entre aplicar y no aplicar el algoritmo. Un valor positivo implica un beneficio neto en promedio, mientras que un valor negativo sugiere un impacto globalmente perjudicial.

$$\text{Balance de impacto}_{alg,b} = \frac{1}{N_{\text{total},\,b}} \sum_{i=1}^{N_{\text{total},\,b}} \left(AP^{(i)}_{alg,b} - AP^{(i)}_{none,b} \right)$$

- **Ganancia relativa media**: expresa, para un algoritmo y una configuración de brillo b, la mejora relativa respecto al caso base sin algoritmo. Resulta especialmente útil porque contextualiza la magnitud de la ganancia, que no es equivalente en rangos bajos y altos de AP.

$$\text{Ganancia relativa media}_{alg,b} = \frac{AP_{alg,b} - AP_{none,b}}{AP_{none,b}} \times 100$$

La incorporación de estas métricas relativas proporciona una visión más detallada del comportamiento de los algoritmos de mejora bajo condiciones de brillo reducido. En particular, permiten discernir no solo si un método aumenta el rendimiento global del modelo, sino también con qué consistencia lo hace y a qué coste en términos de posibles pérdidas de precisión.

6.4. Resultados cualitativos

Antes de presentar los resultados cuantitativos y las predicciones de los distintos algoritmos, se realiza una evaluación cualitativa de las imágenes procesadas a modo de resumen de todas las transformaciones que se han hecho durante el proyecto. Para ello, en la Figura 39 se muestran comparaciones visuales de varias imágenes originales frente a sus respectivas variaciones tras la aplicación de diferentes métodos de mejora y ajuste de brillo.

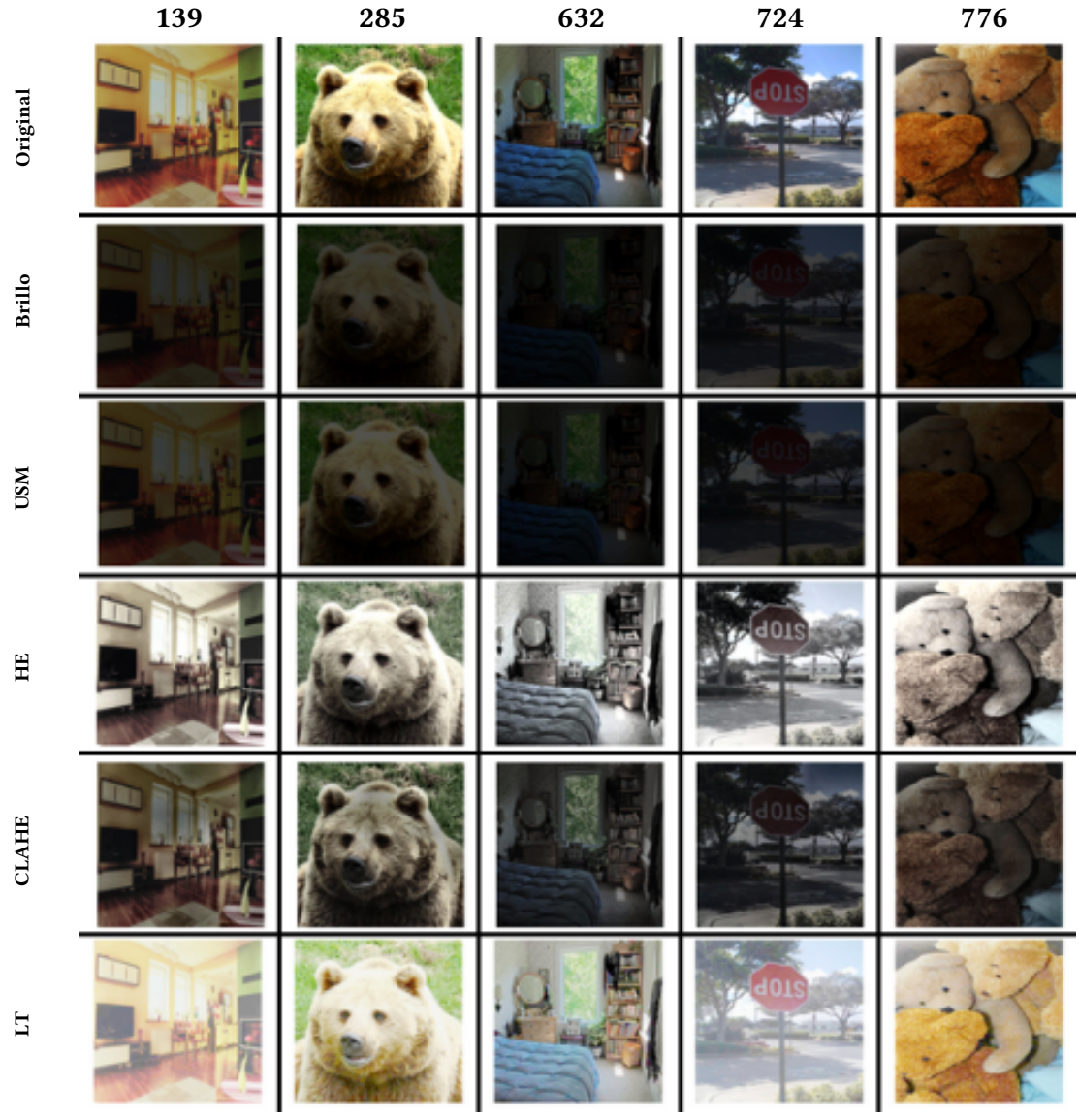


Figura 39: Comparación visual entre imagen original y distintos algoritmos de mejora y brillo, las columnas representan los identificadores de las distintas imágenes y las filas la transformación aplicada.

Esta representación permite observar de manera directa cómo cada algoritmo afecta aspectos como el contraste, la luminosidad y la visibilidad de detalles en las imágenes a simple vista. Al examinar las diferencias de forma visual, es posible identificar patrones de comportamiento de los algoritmos, así como sus fortalezas y limitaciones en contextos específicos, antes de respaldar estas observaciones con otras métricas.

Además, se presenta la Figura 40 que muestra imágenes con niveles de brillo reducidos (por ejemplo, -3, -5 y -10), que son con los que se ha trabajado en los experimentos, aparte de otros valores de brillo aleatorios dentro de estos mismos rangos de forma acotada, con el objetivo de evaluar cómo afectan estos cambios a la percepción visual y a la capacidad del modelo para detectar y localizar los objetos de la imagen.

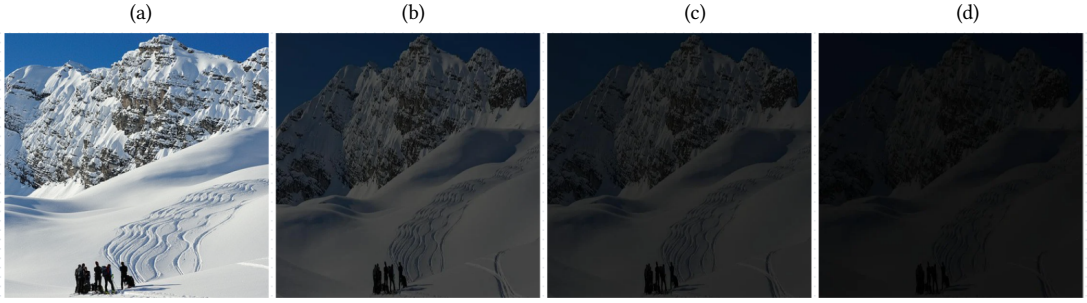


Figura 40: Comparación de distintos valores de brillo para una misma imagen: (a) original, (b) brillo reducido valor 3, (c) brillo reducido valor 5, (d) brillo reducido valor 10.

En la Figura 41 se presenta una imagen de escritorio tomada de cerca, en la que los objetos principales son bastante visibles. Las anotaciones indican la presencia de un portátil, un teclado, un ratón, varios libros en la estantería y una taza sobre la mesa. Estas anotaciones constituyen la referencia para evaluar cualitativamente el desempeño del modelo YOLO en esta imagen concreta.

En la versión original de esta imagen, el modelo detecta correctamente tres objetos: portátil, ratón y taza, mostrando valores de confianza muy altos para cada detección, superiores a 0.6 en todos los objetos y llegando hasta más de 0.9 en el caso del portátil, que es el objeto principal y más grande de la escena. A pesar de la alta confiabilidad en los objetos detectados, se introduce un falso positivo, reflejando cómo el modelo puede generar ocasionalmente predicciones adicionales incluso en condiciones favorables. Sin embargo, esto confirma que, bajo las condiciones de esta imagen, YOLO se ajusta de manera estable y fiable a la escena real.

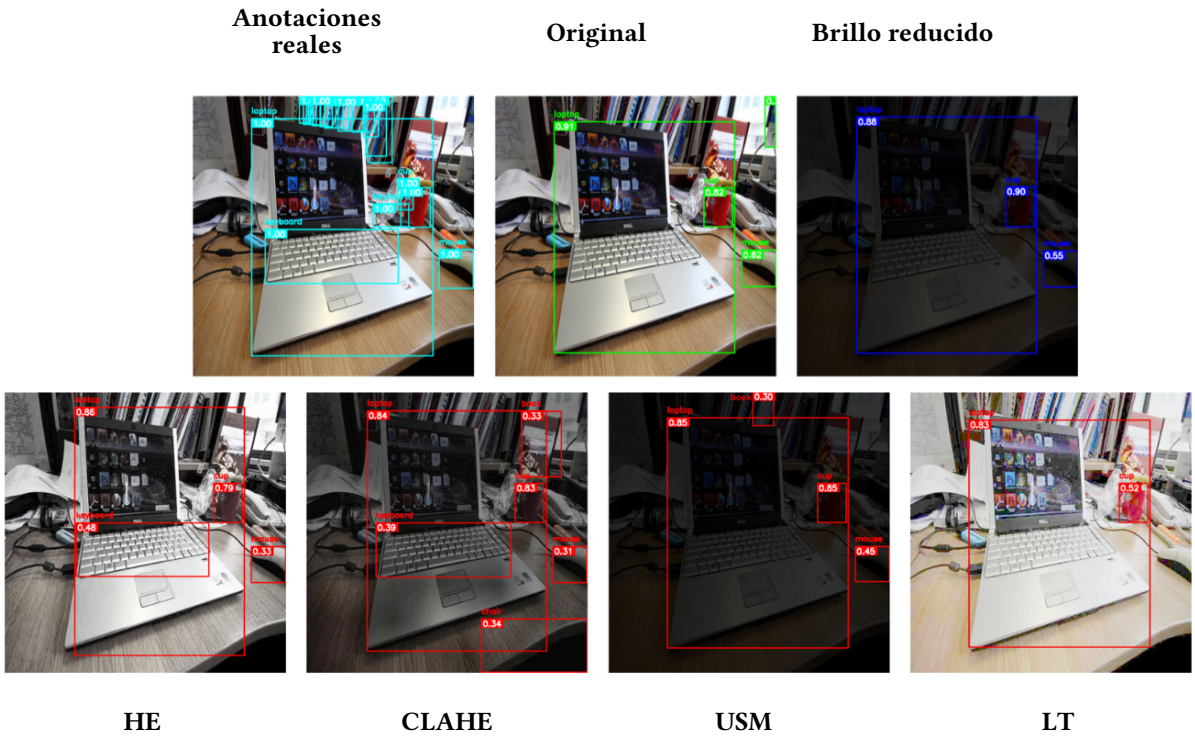


Figura 41: Comparativa de detección de objetos bajo distintos niveles de brillo y técnicas de mejora, junto con las anotaciones reales de COCO.

Cuando se aplica una reducción de brillo a esta imagen, el modelo conserva todas las detecciones, excepto el falso positivo del ejemplo anterior. En general, muestra apenas ligeras disminuciones en la confianza y, de manera destacable, aumenta la confianza de la taza.

Por su parte, la ecualización del histograma (HE) y adaptativa limitada por contraste (CLAHE) sobre esta misma imagen permiten incrementar el número de objetos detectados, incluyendo en ambos casos elementos como el teclado, que no había sido identificado hasta el momento, lo que supone una mejora en el recall. No obstante, este aumento de cobertura se acompaña de una reducción en la confianza de objetos como el ratón, ya que las demás bajadas no son significativas y de la aparición de falsos positivos como silla o libro en el caso de CLAHE. HE, en cambio, mantiene la estabilidad de los objetos principales sin introducir falsos positivos y todas las detecciones correctas conservan una localización acertada.

La máscara de enfoque (USM) aplicada a esta imagen muestra un comportamiento intermedio: mantiene alta la confianza en el portátil y detecta un libro, que no había sido detectado hasta el momento y es considerado un objeto pequeño y, por tanto, más difícil de detectar. Es-

to evidencia que, en este caso, el algoritmo ha favorecido la aparición de detalles secundarios, logrando nuevamente mejorar el recall en comparación con la imagen original y la versión con brillo reducido. Asimismo, iguala a los dos algoritmos anteriores en recall, aunque en esta ocasión omite el teclado. Aun así, la precisión en la localización de los objetos detectados correctamente se mantiene, demostrando que el algoritmo conserva su efectividad al identificar las regiones principales de interés.

Por último, la transformación logarítmica (LT) es la que consigue una peor actuación, logrando identificar únicamente el portátil y la taza.

En conjunto, los resultados cualitativos de esta imagen específica muestran que los algoritmos de preprocesamiento pueden aumentar el recall respecto a la versión original. USM y HE mejoran, por tanto, la cantidad de objetos detectados sin generar falsos positivos. CLAHE también incrementa el recall, pero a costa de introducir dos falsos positivos. LT presenta el desempeño más limitado, detectando solo dos objetos.

Estos hallazgos indican que, para esta imagen concreta, la elección de la técnica de preprocesamiento depende de si se prioriza maximizar el recall o mantener la precisión sin falsos positivos. La versión original sigue siendo estable y confiable, pero algoritmos como USM y HE pueden ser útiles para aumentar la cobertura de objetos sin comprometer la precisión ni la localización adecuada. Esto refuerza la necesidad de un análisis cuantitativo complementario que valide estas tendencias y permita establecer criterios objetivos para la mejora del sistema.

6.5. Resultados cuantitativos

6.5.1. Rendimiento absoluto

Los resultados cuantitativos obtenidos permiten evaluar de forma objetiva el rendimiento de los distintos tamaños del modelo YOLOv11 (n, s, m, l, x) frente a las variaciones en las condiciones de iluminación y la aplicación de algoritmos de mejora. Para ello, se han calculado las métricas de precisión media (AP) y recall promedio (AR), siguiendo el protocolo de evaluación COCO, sobre el conjunto de imágenes del subconjunto de validación, que son 5000 en total, procesadas en distintas versiones: original, con brillo reducido y tras la aplicación de los algoritmos estudiados.

Como se muestra en la Tabla 1, que muestra la variación del modelo YOLOv11 según el tamaño del modelo y los distintos valores de brillo estudiados, los valores de precisión media promedio (mAP) aumentan de forma consistente a medida que el modelo incrementa su tamaño, desde la versión n hasta la x. Sin embargo, todas las configuraciones sufren una degradación de rendimiento cuando el brillo disminuye de manera severa (Brillo 10), siendo más acusada en los modelos pequeños. Por ejemplo, el modelo n pasa de 0.30 en la condición original a 0.10 en Brillo 10, lo que supone una reducción de más del 65 %. En cambio, el modelo x desciende de 0.45 a 0.20 en la misma condición, lo que equivale a una pérdida cercana al 55 %, mostrando mayor resistencia a la alteración. El uso de un brillo intermedio o aleatorio mitiga parcialmente esta caída, situándose en valores intermedios entre el caso original (sin alteraciones) y el escenario más extremo.

Tabla 1: Resultados de precisión media (AP) según tamaño de modelo y alteración de brillo.

Modelo/Brillo	Original (1)	3	5	10	{3,4,5}
n	0.30	0.24	0.19	0.10	0.21
s	0.38	0.29	0.25	0.15	0.27
m	0.42	0.33	0.27	0.17	0.30
l	0.44	0.34	0.28	0.18	0.31
x	0.45	0.36	0.31	0.20	0.34

En la Tabla 2, que compara lo mismo que la anterior, se observa un comportamiento análogo en los valores de recall promedio (mAR). A mayor tamaño de modelo, mejor cobertura de detección, especialmente en condiciones adversas de brillo. Los modelos pequeños (n y s) muestran pérdidas considerables de recall con iluminación reducida, mientras que los modelos medianos y grandes preservan en mayor medida la capacidad de recuperar objetos presentes en la escena. Por ejemplo, el modelo l conserva un AR de 0.19 en Brillo 10 frente al 0.46 en la condición original, mientras que el modelo s desciende de 0.40 a 0.16, reflejando una caída mucho más pronunciada.

La comparación conjunta entre AP y AR revela un patrón interesante: mientras que el AP refleja con mayor claridad la pérdida de precisión en condiciones de brillo extremo, el AR indica que los modelos grandes aún logran detectar una fracción significativa de objetos, aunque con

Tabla 2: Resultados de recall promedio (AR) según tamaño de modelo y alteración de brillo.

Modelo/Brillo	Original (1)	3	5	10	{3,4,5}
n	0.33	0.26	0.21	0.12	0.23
s	0.40	0.32	0.27	0.16	0.29
m	0.44	0.35	0.29	0.19	0.32
l	0.46	0.36	0.30	0.19	0.33
x	0.47	0.38	0.33	0.21	0.36

menor exactitud en la localización. Esto sugiere que los modelos de mayor tamaño no solo mantienen un rendimiento superior en términos de precisión, sino que también garantizan una cobertura más estable frente a cambios de iluminación.

Por otra parte, los resultados de los heatmaps se presentan separados según el tamaño del modelo YOLOv11, como se puede observar en las siguientes figuras: Figura 42, Figura 43, Figura 44, Figura 45 y Figura 46, las cuales, a diferencia de las tablas, permiten comparar directamente AP y AR, mostrando el desempeño de cada versión ante distintas condiciones de imagen. Para cada experimento de cada modelo se ha generado un heatmap que refleja los valores obtenidos de AP para todas las combinaciones de brillo, algoritmos y tamaño de objeto evaluados. Cada heatmap incluye una leyenda específica, asociando los intervalos de color a los valores de AP correspondientes, lo que facilita la interpretación visual de las variaciones de rendimiento en función de las condiciones de la imagen.

En condiciones óptimas, sobre las imágenes originales sin alteraciones, se observa que el rendimiento mejora progresivamente con el tamaño del modelo. Los modelos más grandes (l y x) alcanzan valores de AP superiores al 0.70 % y AR cercanos al 0.65 %, mientras que los modelos más pequeños (n y s) se sitúan por debajo del 0.50 % en ambas métricas. Esta tendencia confirma que los modelos de mayor complejidad son capaces de extraer características más precisas y realizar detecciones más fiables, aunque requieren mayor capacidad computacional.

Cuando se introduce una reducción de brillo, el rendimiento se ve afectado de forma significativa en todos los modelos. La caída de AP puede superar el 30 % en los modelos pequeños, mientras que en los modelos grandes se mantiene en torno al 15–20 %. Esta pérdida de precisión se traduce también en una disminución del recall, indicando que el modelo no solo localiza

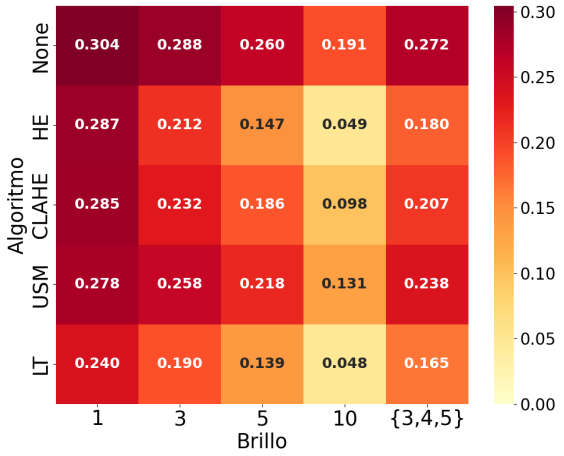
peor los objetos, sino que también deja de detectar algunos de ellos. La sensibilidad a la iluminación se manifiesta de manera más acusada en los modelos menos complejos, que muestran mayor variabilidad en sus predicciones y una mayor propensión a generar falsos positivos o a omitir objetos presentes en la imagen.

Por otro lado, la aplicación de algoritmos de mejora permite recuperar parcialmente el rendimiento perdido. CLAHE y la ecualización del histograma destacan como los métodos más efectivos, logrando incrementos de AP y AR que en algunos casos se acercan a los valores originales. En particular, CLAHE muestra una mejora consistente en todos los tamaños de modelo, especialmente en los intermedios (s y m), donde se observa una recuperación significativa tanto en precisión como en cobertura.

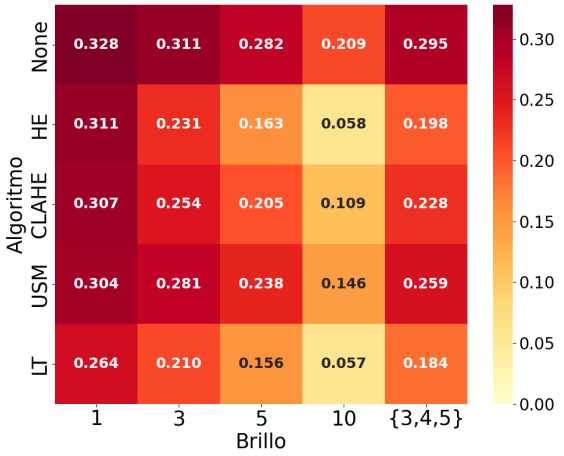
Por ejemplo, en el modelo YOLOv11-m, la aplicación de CLAHE permite alcanzar un AP de 0.46 frente al 0.52 de la imagen original, mientras que el AR se mantiene estable en torno a 0.38–0.40. La máscara de enfoque y la transformación logarítmica también ofrecen mejoras respecto a la imagen oscurecida, aunque sus resultados son más variables y dependen del tamaño del modelo. En general, los modelos grandes se benefician más de estas técnicas, mostrando mayor capacidad de recuperación, mientras que los modelos pequeños presentan limitaciones más evidentes.

Además, se ha observado que el impacto de las transformaciones no es uniforme entre las distintas clases de objetos ni entre los tamaños de imagen. Algunas categorías presentan mayor estabilidad frente a la reducción de brillo, mientras que otras se ven más afectadas, lo que sugiere que la sensibilidad del modelo varía según la naturaleza visual del objeto y su representación en el conjunto de datos. Esta variabilidad por clase refuerza la necesidad de considerar no solo métricas globales, sino también el comportamiento específico por categoría, especialmente en aplicaciones donde ciertos objetos tienen mayor relevancia.

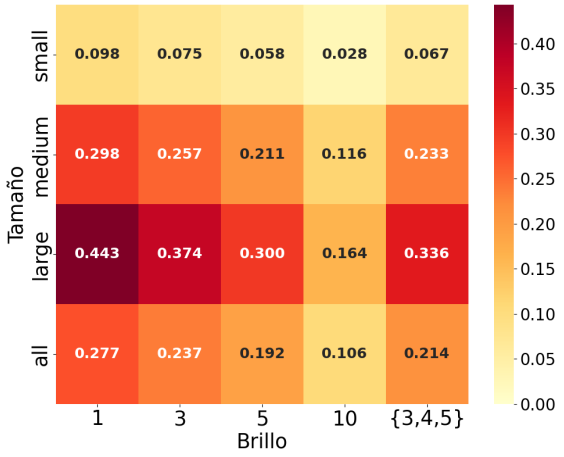
En conjunto, los resultados cuantitativos permiten establecer conclusiones claras: la reducción de brillo afecta negativamente al rendimiento del modelo en todos los tamaños, aunque con mayor severidad en los modelos pequeños; los algoritmos de mejora ofrecen una solución parcial pero efectiva, siendo CLAHE el más robusto en términos generales y los modelos más grandes muestran una mayor capacidad de adaptación ante condiciones adversas.



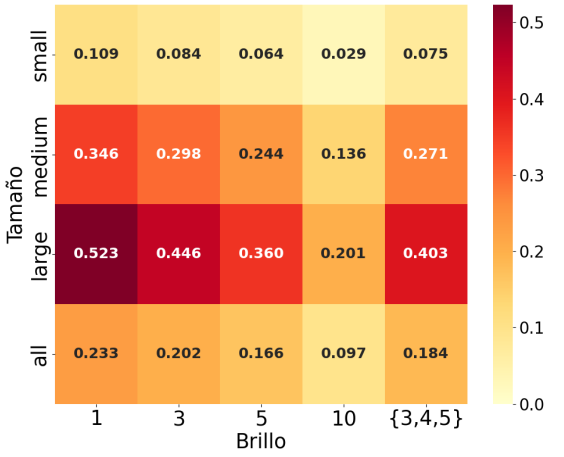
(a) AP: brillo vs algoritmo



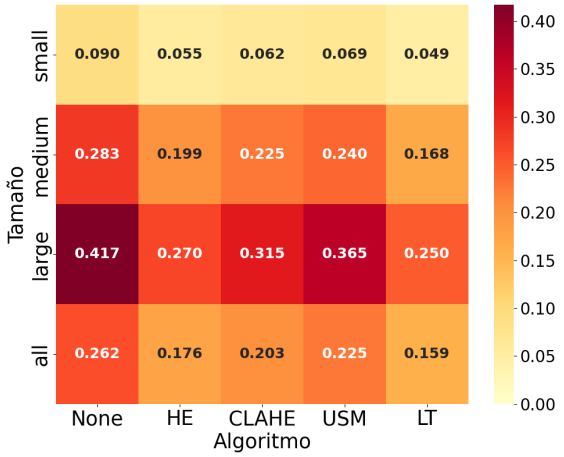
(b) AR: brillo vs algoritmo



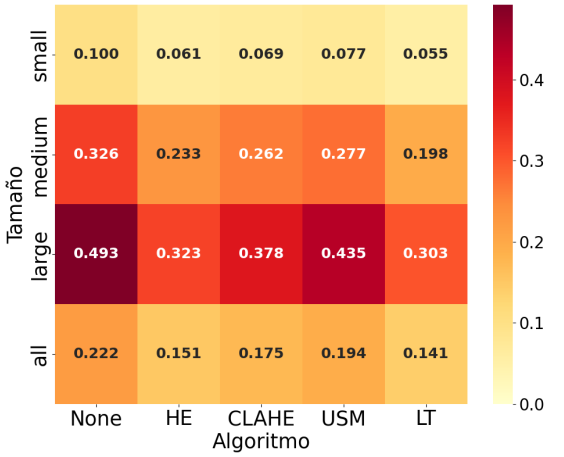
(c) AP: brillo vs tamaño



(d) AR: brillo vs tamaño

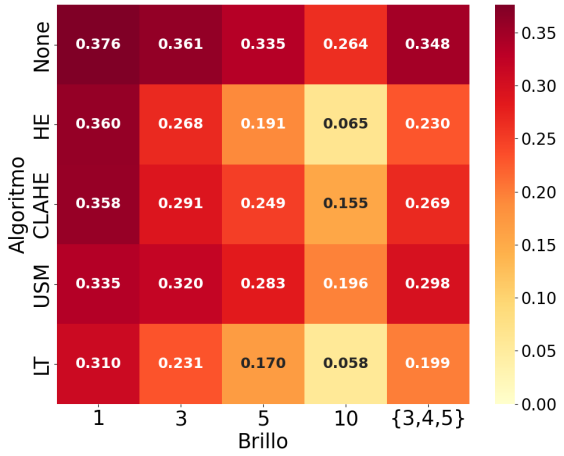


(e) AP: algoritmo vs tamaño

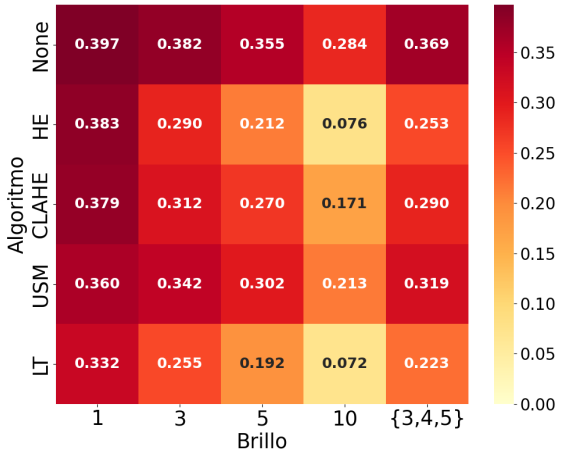


(f) AR: algoritmo vs tamaño

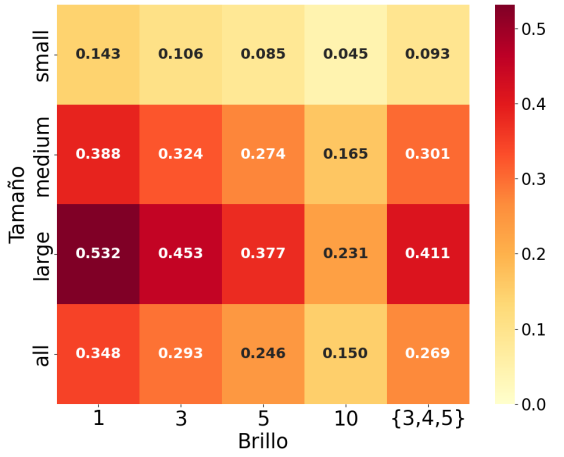
Figura 42: Heatmaps de AP y AR para el modelo YOLO-n.



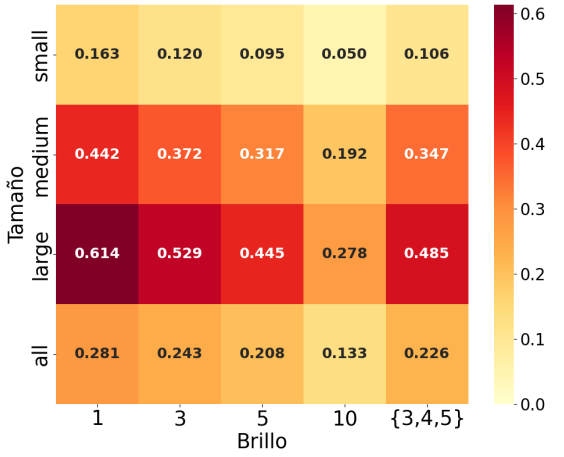
(a) AP: algoritmo vs brillo



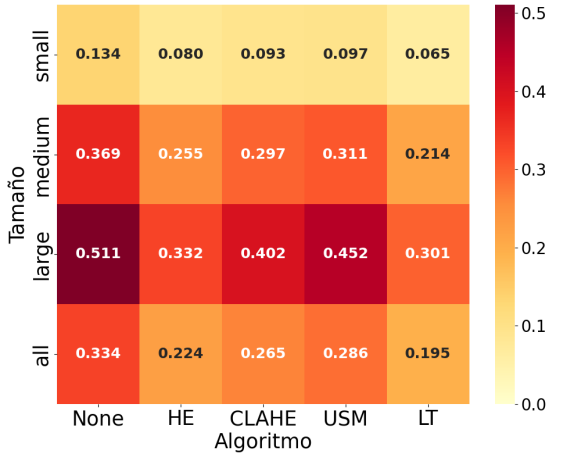
(b) AR: algoritmo vs brillo



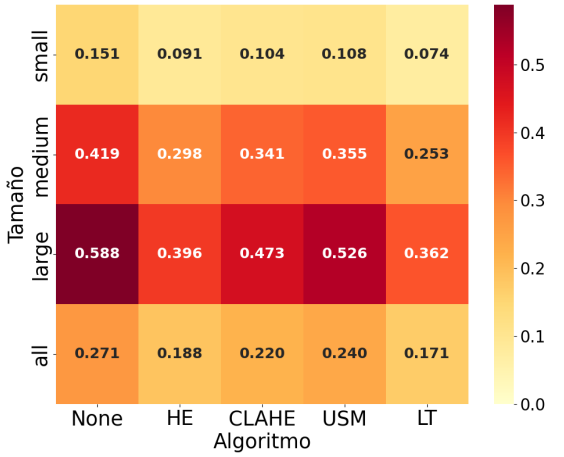
(c) AP: brillo vs tamaño



(d) AR: brillo vs tamaño

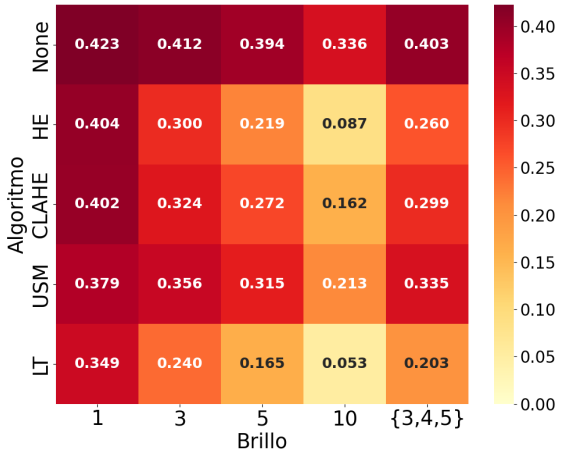


(e) AP: algoritmo vs tamaño

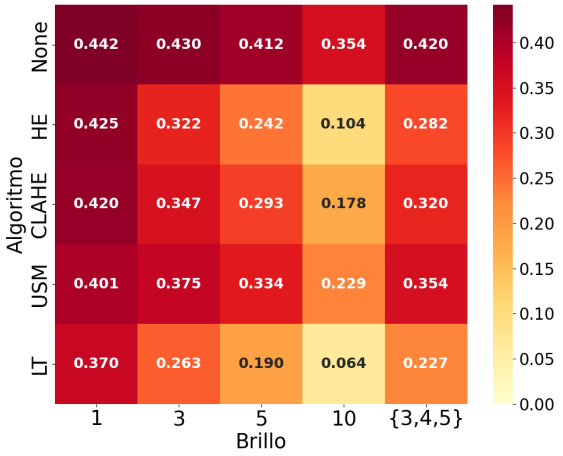


(f) AR: algoritmo vs tamaño

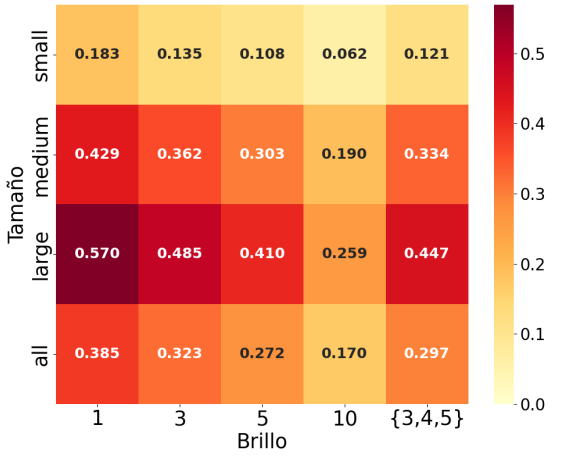
Figura 43: Heatmaps de AP y AR para el modelo YOLO-s.



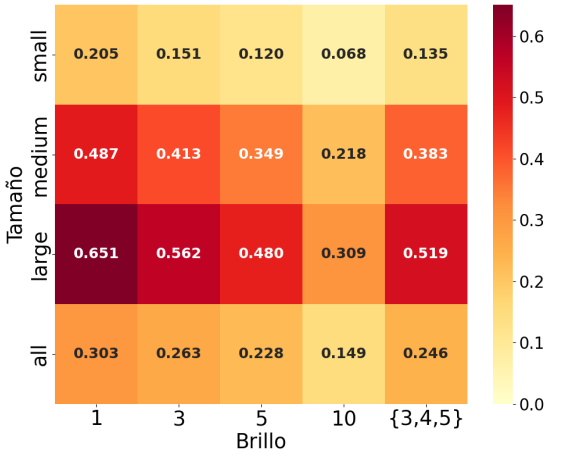
(a) AP: algoritmo vs brillo



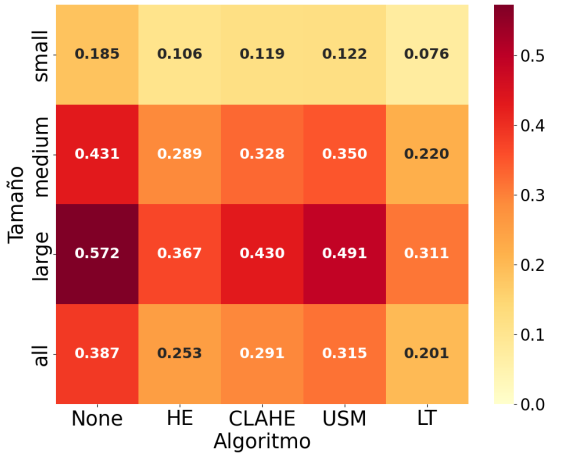
(b) AR: algoritmo vs brillo



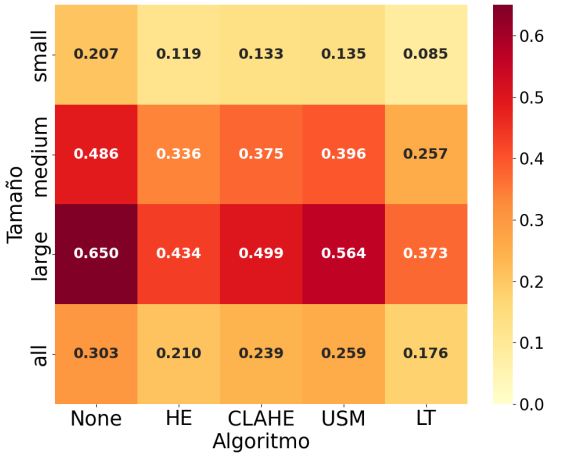
(c) AP: brillo vs tamaño



(d) AR: brillo vs tamaño

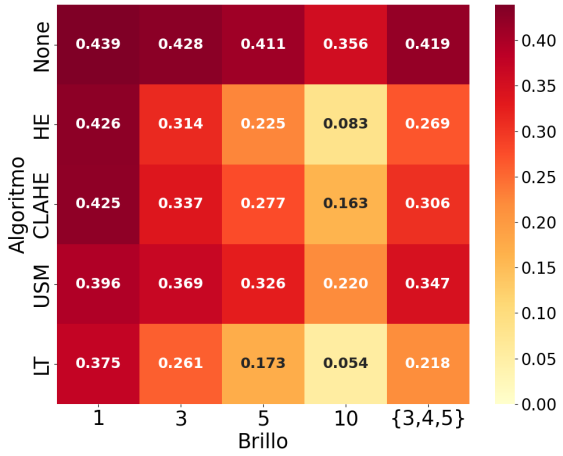


(e) AP: algoritmo vs tamaño

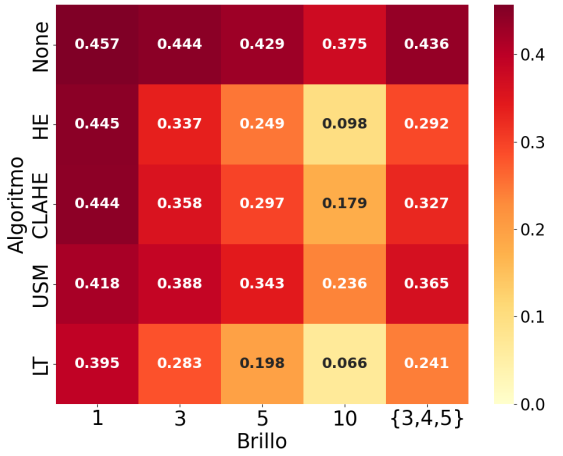


(f) AR: algoritmo vs tamaño

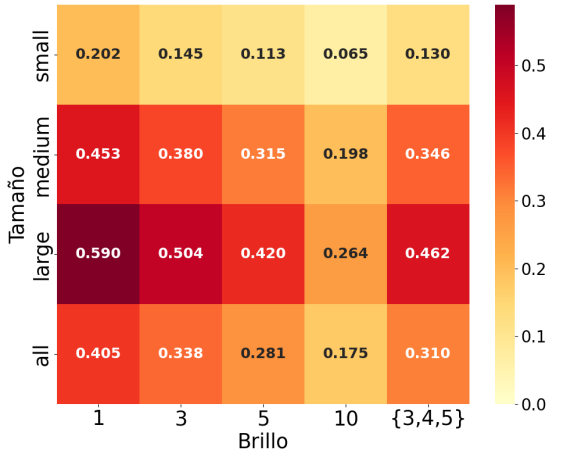
Figura 44: Heatmaps de AP y AR para el modelo YOLO-m.



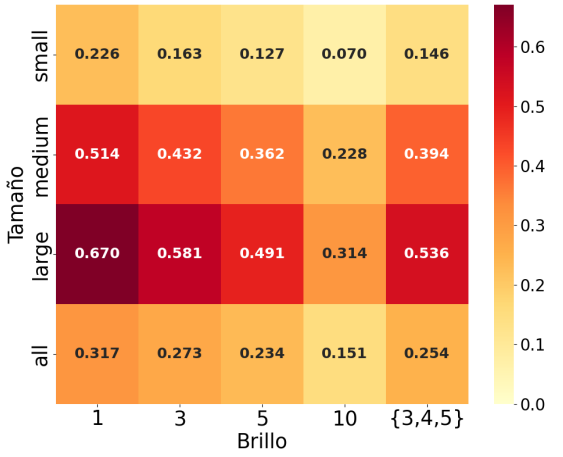
(a) AP: algoritmo vs brillo



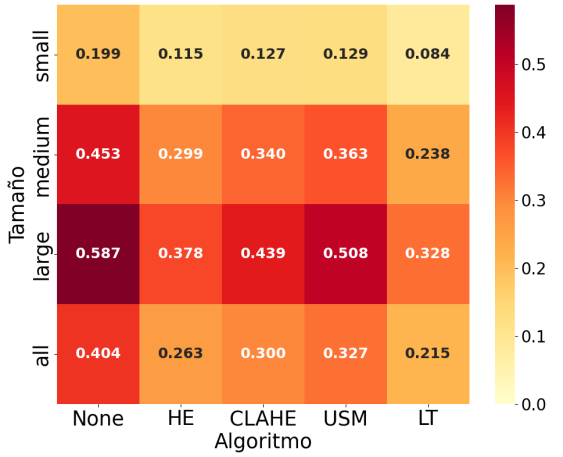
(b) AR: algoritmo vs brillo



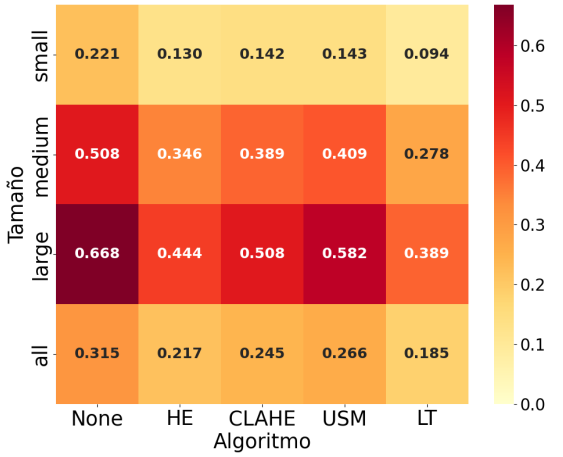
(c) AP: brillo vs tamaño



(d) AR: brillo vs tamaño

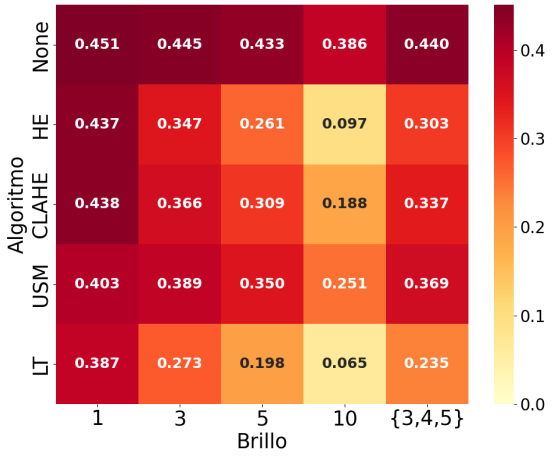


(e) AP: algoritmo vs tamaño

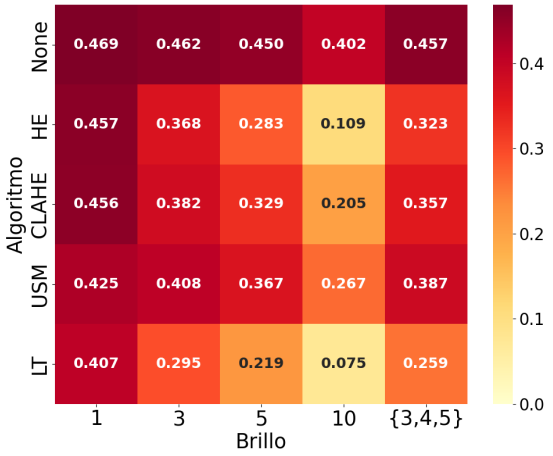


(f) AR: algoritmo vs tamaño

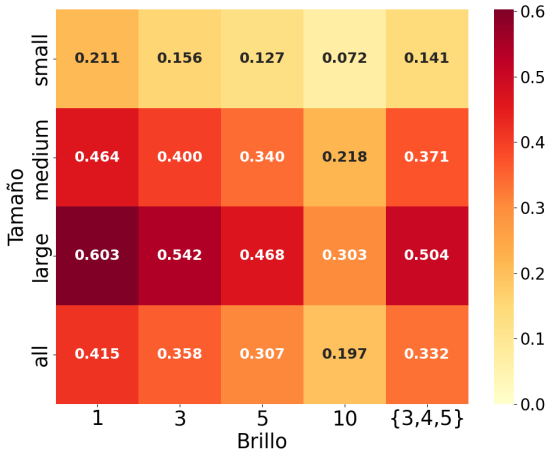
Figura 45: Heatmaps de AP y AR para el modelo YOLO-l.



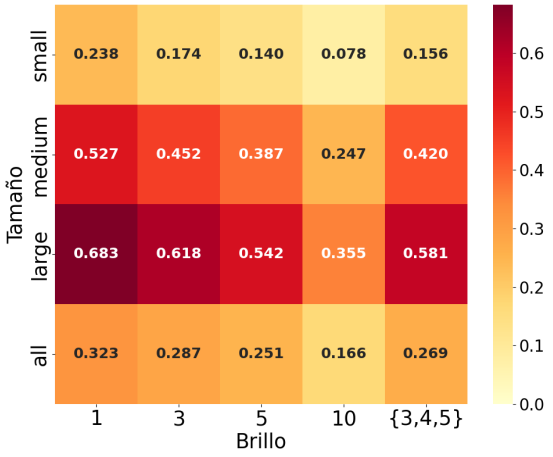
(a) AP: algoritmo vs brillo



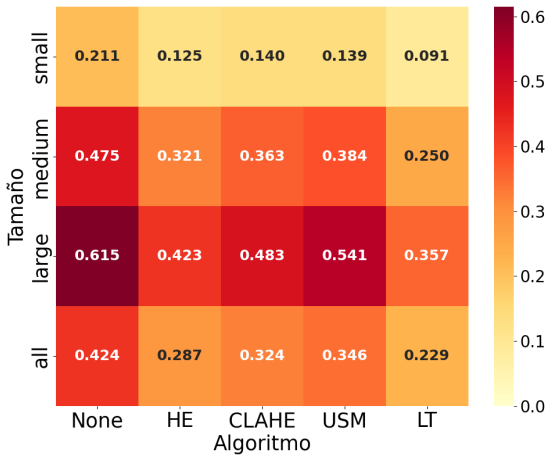
(b) AR: algoritmo vs brillo



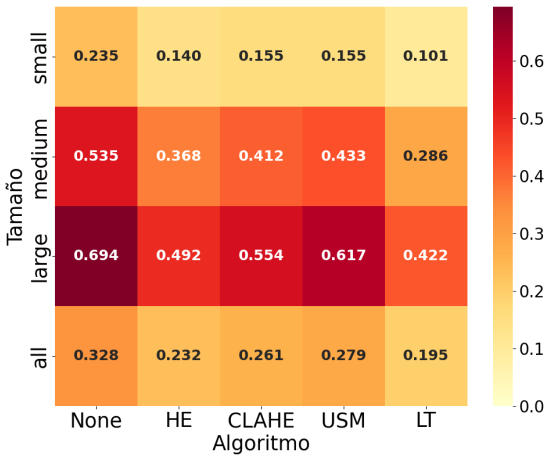
(c) AP: brillo vs tamaño



(d) AR: brillo vs tamaño



(e) AP: algoritmo vs tamaño



(f) AR: algoritmo vs tamaño

Figura 46: Heatmaps de AP y AR para el modelo YOLO-x.

6.5.2. Rendimiento relativo del uso de algoritmo de mejora con respecto a su no uso

El análisis de la primera parte de estas métricas presentadas en la Tabla 3 muestra que, de manera general y considerando los valores medios de AP agregados por combinación de nivel de brillo y algoritmo aplicado, la aplicación de los métodos evaluados no produce mejoras respecto al caso base sin algoritmo. Tanto el balance de impacto como la ganancia relativa media reflejan pérdidas en todos los escenarios, lo que indica que, en promedio, la aplicación de los algoritmos tiende a reducir el rendimiento.

Tabla 3: Balance de impacto y ganancia relativa media por nivel de brillo y algoritmo

Brillo	Algoritmo	Balance impacto	Ganancia relativa media (%)
10	HE	-0.230	-75.1
10	CLAHE	-0.153	-49.5
10	USM	-0.104	-33.3
10	LT	-0.251	-81.0
3	HE	-0.099	-25.6
3	CLAHE	-0.077	-19.8
3	USM	-0.048	-12.3
3	LT	-0.148	-37.9
5	HE	-0.158	-43.2
5	CLAHE	-0.108	-29.3
5	USM	-0.068	-18.3
5	LT	-0.198	-53.2
{3,4,5}	HE	-0.128	-34.0
{3,4,5}	CLAHE	-0.093	-24.5
{3,4,5}	USM	-0.059	-15.5
{3,4,5}	LT	-0.172	-45.2

Los valores de balance de impacto son siempre negativos, con pérdidas que van desde moderadas como -0.048 (brillo 3 con USM) hasta más pronunciadas como -0.251 (brillo 10 con LT). De igual modo, la ganancia relativa media varía entre -12.3 % y -81.0, confirmando que la magnitud de la degradación depende del nivel de brillo y del algoritmo empleado.

El impacto negativo no es uniforme y depende de la intensidad del brillo. Con valores bajos o moderados (brillo 3 o 5) la degradación es menor, especialmente con USM, mientras que con brillo alto (valor 10) los resultados se deterioran significativamente, siendo LT el más desfavorable con -81.0 % de pérdida relativa. La condición de brillo aleatorio se sitúa en un punto intermedio, aunque más cercana a los escenarios de alto brillo.

Cabe destacar que estos resultados reflejan un análisis global, donde cada fila del experimento representa un promedio sobre un amplio conjunto de imágenes. Bajo este enfoque, los algoritmos tienden a reducir el rendimiento. Sin embargo, existen ejemplos concretos donde algún algoritmo mejoró la predicción, especialmente en imágenes con condiciones extremas de iluminación, aunque estas mejoras individuales se diluyen al promediar sobre todo el conjunto.

Se observa que la magnitud de la pérdida depende tanto del algoritmo como del nivel de brillo. USM se comporta como el menos perjudicial, especialmente con brillo bajo (valor 3), con pérdida mínima (-0.048) y ganancia relativa media de -12.3 %. LT es el más perjudicial, especialmente con brillo alto (valor 10), con balance de impacto de -0.251 y ganancia relativa media de -81.0 %. HE y CLAHE presentan resultados intermedios, manteniendo la misma tendencia: mayor brillo implica mayor degradación.

Por último, esto se refleja en las figuras. La Figura 47 muestra las transformaciones aplicadas al conjunto de datos únicamente por brillo y algoritmos, sin involucrar un modelo de detección. La Figura 48 presenta los resultados usando YOLOv11n, mientras que la Figura 49 corresponde a YOLOv11x. La comparación evidencia que YOLOv11x, al ser más grande, compensa mejor las alteraciones de brillo, manteniendo un desempeño más robusto.

En conclusión, los resultados permiten identificar tendencias claras: USM es el algoritmo menos perjudicial y recomendable para minimizar pérdidas, mientras que LT representa la opción más arriesgada. Además, a mayor brillo, mayor es el impacto negativo de todos los algoritmos. Esto sugiere que aplicar técnicas de mejora de manera indiscriminada puede ser contraproducente y que la selección del método debe considerar tanto el tipo de algoritmo como la intensidad del brillo en las imágenes.

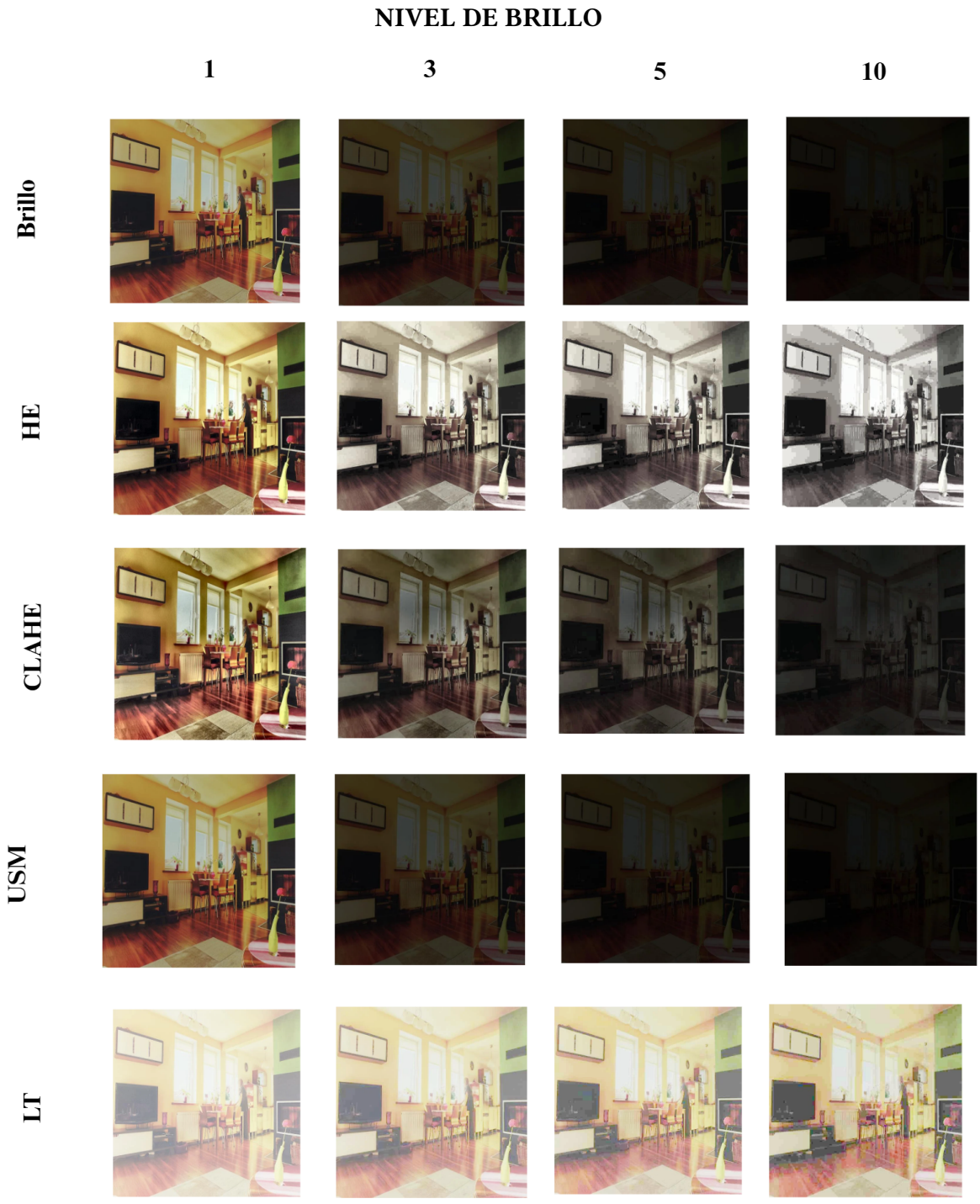


Figura 47: Resultados de la aplicación de todas las combinaciones de brillo y algoritmos realizadas para una misma imagen con identificador 129.

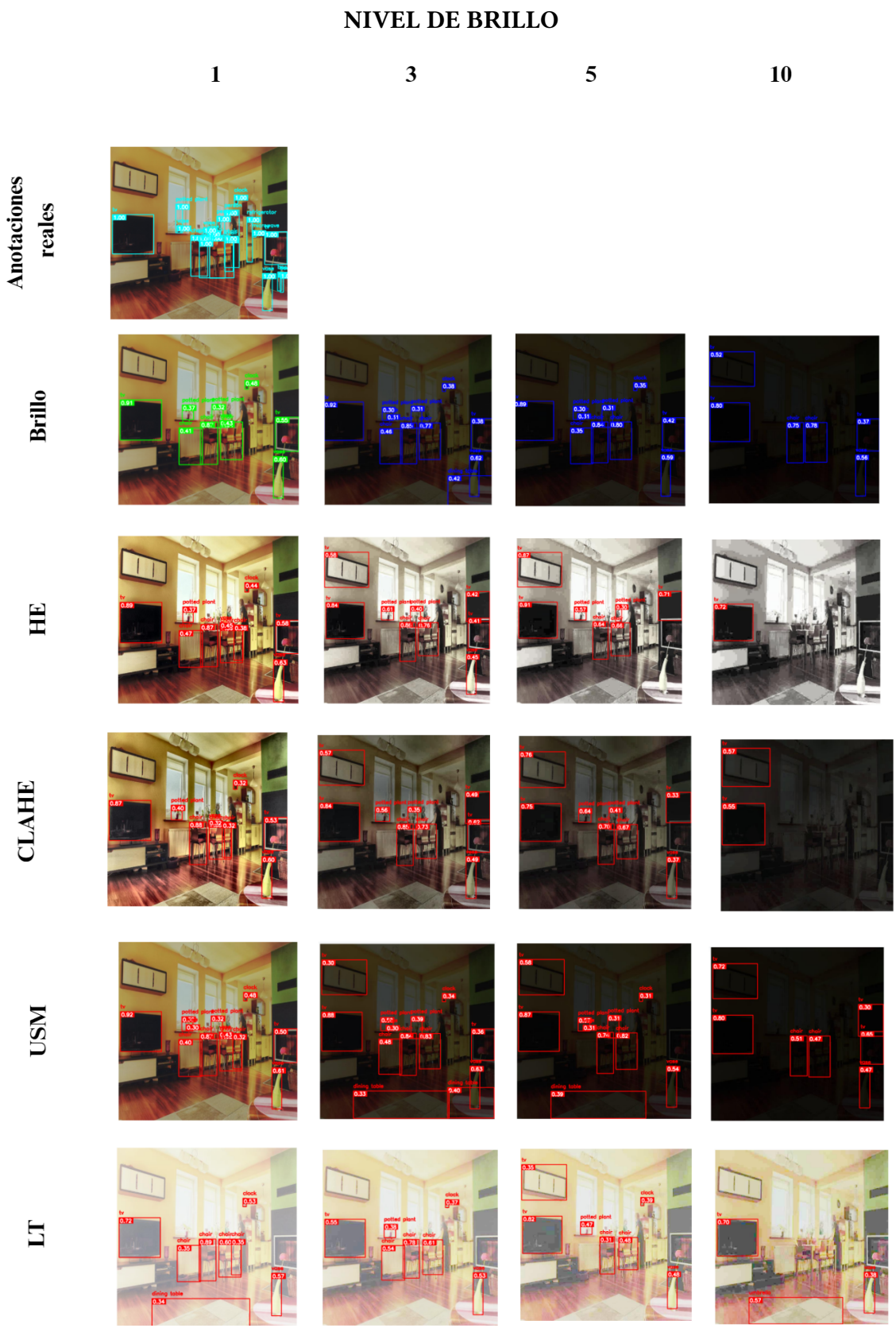


Figura 48: Resultados de las predicciones para todas las combinaciones de brillo y algoritmos realizadas con YOLOv11n para una misma imagen con identificador 129.

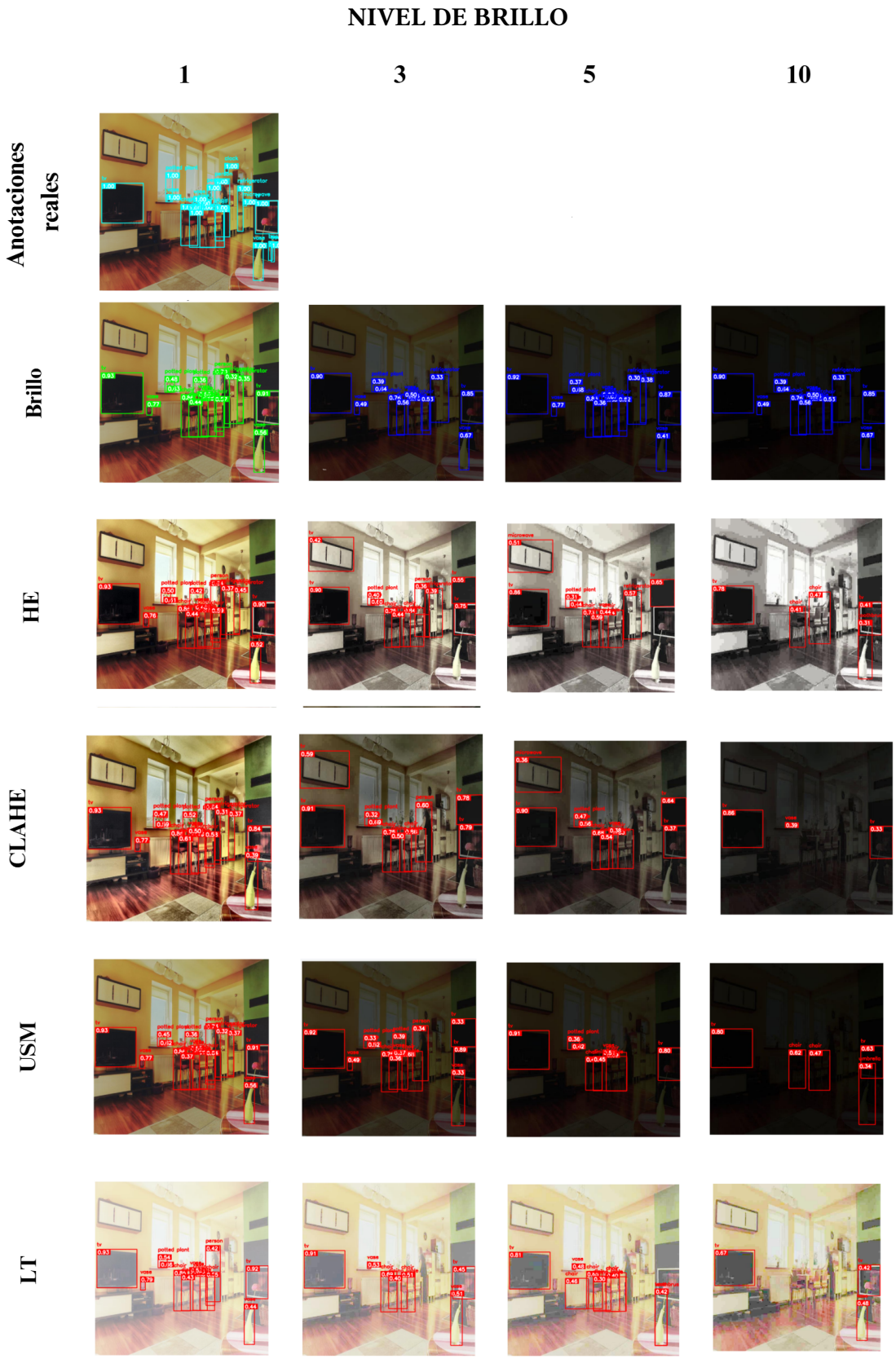


Figura 49: Resultados de las predicciones para todas las combinaciones de brillo y algoritmos realizadas con YOLOv11x para una misma imagen con identificador 129.

7

Discusión de los resultados

Los resultados obtenidos en este estudio permiten realizar una reflexión crítica sobre el comportamiento del modelo YOLOv11 ante variaciones en las condiciones de iluminación y la aplicación de algoritmos de mejora. La combinación de análisis cualitativo y cuantitativo ha revelado patrones consistentes que aportan información valiosa sobre la robustez del sistema, sus limitaciones y las posibles estrategias para optimizar su rendimiento en entornos reales.

Uno de los hallazgos consiste en que la disminución de brillo genera un efecto similar en las métricas de evaluación, afectando tanto la precisión (AP) como la cobertura (AR). Los objetos pequeños son los más perjudicados, con una caída significativa en AP y AR, debido a que su información visual se pierde rápidamente cuando la luminosidad disminuye. Los objetos medianos muestran una reducción moderada, mientras que los grandes presentan un impacto menor, aunque no despreciable. Esto indica que la sensibilidad del modelo a la iluminación es particularmente crítica para objetos de tamaño reducido. Además, esta pérdida se manifiesta en una disminución del número de objetos detectados, en la aparición de falsos positivos y en una menor definición de los cuadros delimitadores, como se observa en la Figura 50, que destaca por la imprecisión de los cuadros delimitadores en la versión de brillo reducido con respecto a la original.

Además, el modelo tiende a confundir clases visualmente similares o a omitir objetos con bajo contraste. Esta sensibilidad no se manifiesta de manera uniforme entre todas las clases de objetos: algunas categorías, especialmente aquellas con formas bien definidas o colores llamativos, logran mantener niveles de confianza aceptables incluso bajo condiciones adversas. Un ejemplo de esto se aprecia en la Figura 51, donde, al tratarse de objetos grandes y visualmente representativos, el rendimiento en la detección se mantiene prácticamente sin cambios. Por el

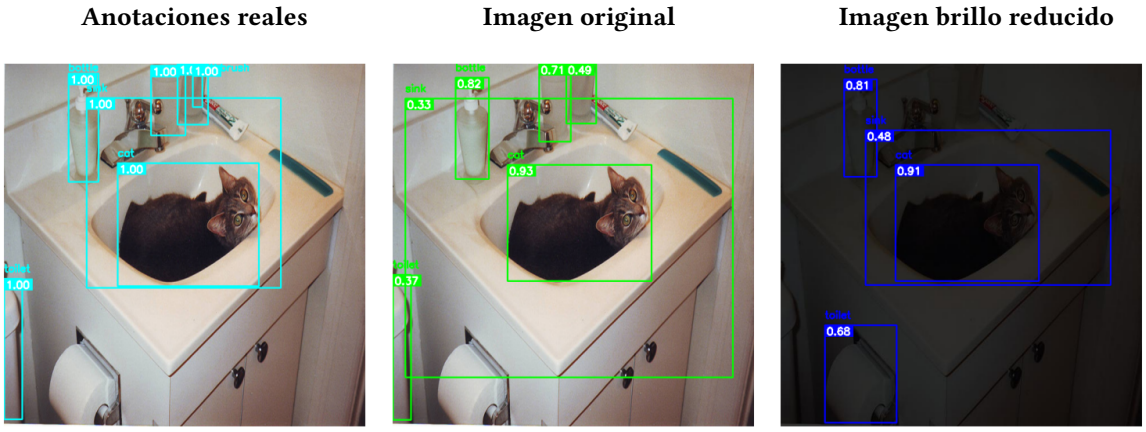


Figura 50: Comparación de distintas detecciones para una misma imagen. Se observa que la reducción de brillo disminuye la precisión en los cuadros delimitadores y aparecen falsos positivos,

contrario, otras clases, más dependientes del contexto, con menor contraste o baja saliencia visual, se ven notablemente perjudicadas; como se observa en la Figura 52, algunos objetos dejan de detectarse y aparecen falsos positivos, reflejando la dificultad del modelo para interpretar correctamente estas instancias en presencia de iluminación reducida.

Incluso, se ha comprobado que el impacto de las transformaciones no es uniforme entre los distintos modelos. Los modelos más grandes, como YOLOv11-l y YOLOv11-x, muestran una mayor capacidad de adaptación, manteniendo niveles de rendimiento más estables frente a la reducción de brillo y respondiendo mejor a las técnicas de mejora. Esto se debe a que su mayor profundidad y capacidad de representación les permite extraer características más robustas, incluso en condiciones de iluminación degradada. Por el contrario, los modelos más pequeños (n, s y m) presentan un rendimiento más limitado. Tienen dificultades para generalizar bajo condiciones de iluminación adversas, mostrando detecciones parciales o una mayor cantidad de falsos positivos, y requieren condiciones de iluminación más favorables para alcanzar un rendimiento comparable.

Esta diferencia entre modelos se refleja claramente en la Figura 53. Los modelos grandes, como YOLOv11-l y YOLOv11-x, detectan múltiples objetos con altos niveles de confianza, gracias a su arquitectura más compleja y al mayor número de parámetros, lo que les permite capturar mejor las características de los objetos y mantener la precisión incluso en escenarios con elementos parcialmente ocultos o superpuestos. Por su parte, los modelos más pequeños,

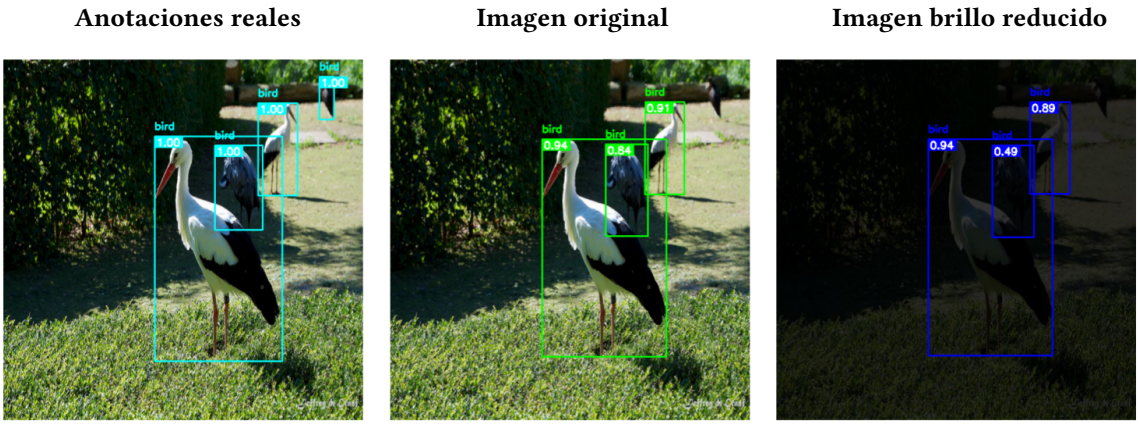


Figura 51: Comparación de distintas detecciones para una misma imagen. A pesar de la reducción de brillo, los objetos con colores distintivos se detectan correctamente.

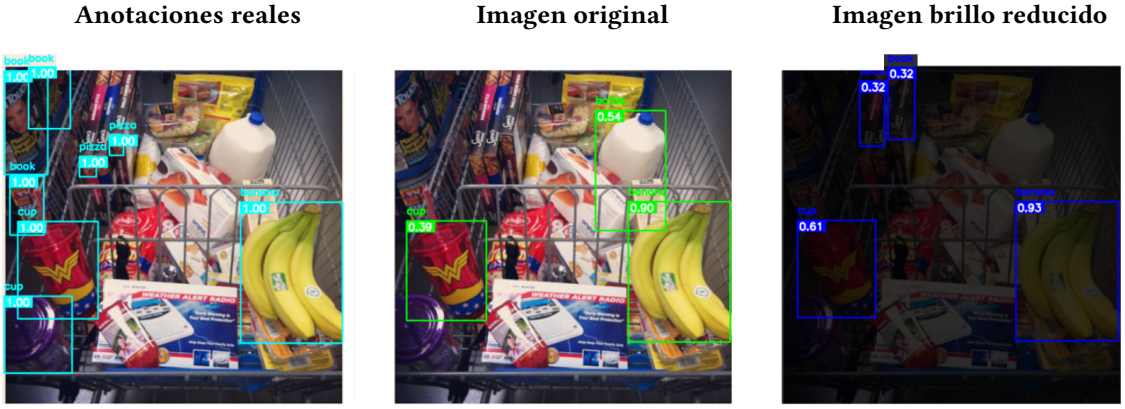


Figura 52: Comparación de distintas detecciones para una misma imagen. De nuevo, la imagen oscurecida genera falsos positivos y omite objetos como la botella.

como YOLOv11-n y YOLOv11-s, apenas identifican un solo objeto, mostrando limitaciones debido a su menor capacidad de procesamiento y al reducido número de filtros en sus capas convolucionales. En el caso intermedio, YOLOv11-m detecta más objetos que n y s, aunque con puntuaciones de confianza menores, evidenciando que la mayor cobertura conlleva menor fiabilidad. Esto refleja que la complejidad del modelo influye tanto en la cantidad de detecciones como en la precisión, un aspecto especialmente relevante para aplicaciones prácticas donde la confianza en la predicción es crucial.

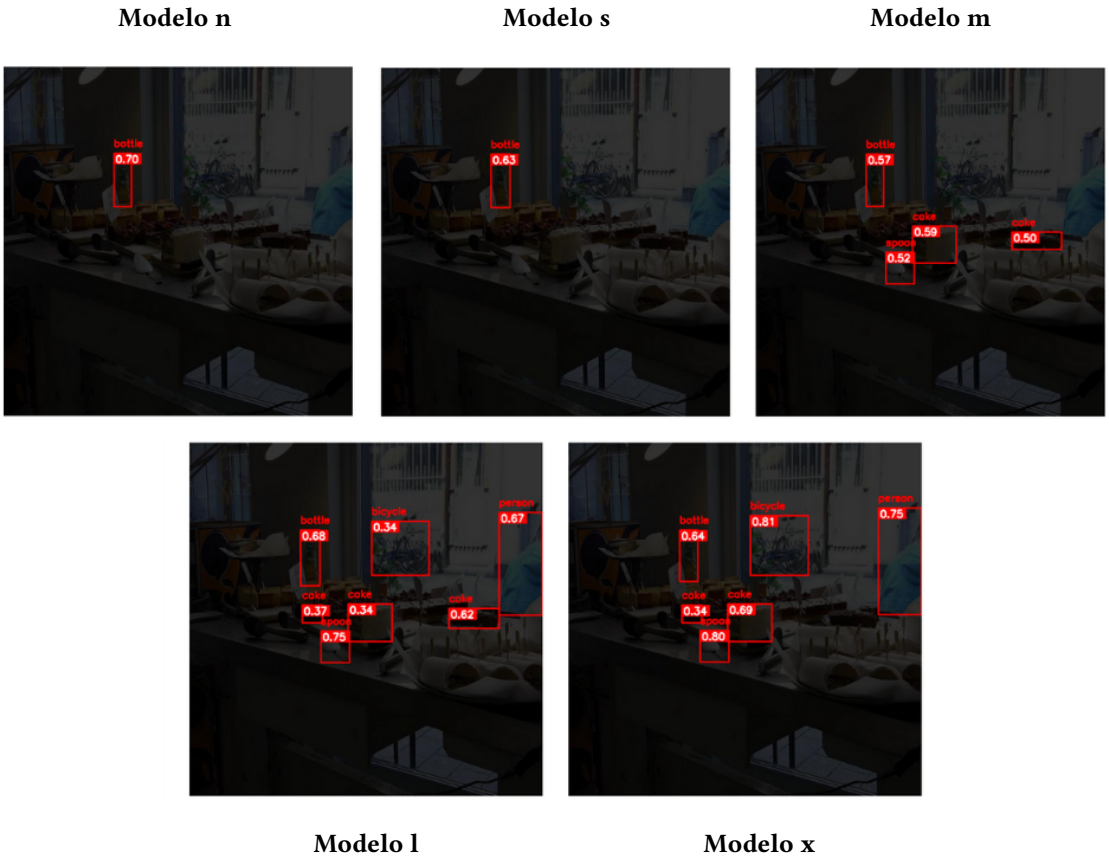


Figura 53: Comparación de distintas detecciones para una misma imagen con todos los tamaños de modelo YOLOV11.

Los resultados muestran de forma destacable que, en la mayoría de las imágenes, la mejora aplicada produce un rendimiento incluso inferior al de la imagen simplemente oscurecida. Esto se refleja en valores de mAP y mAR más bajos respecto a la imagen con brillo reducido, indicando que el detector se confunde ante alteraciones en la intensidad que no corresponden a patrones aprendidos durante el entrenamiento.

Este fenómeno puede parecer contraintuitivo, pero tiene una base técnica clara: los modelos de visión por computador suelen entrenarse con imágenes que presentan distribuciones estándar de luminosidad, intensidad y textura, por lo que al aplicar mejoras visuales, dichas distribuciones se modifican y se introducen inconsistencias que el modelo interpreta de manera incorrecta. En particular, ciertos algoritmos generan microvariaciones que resultan agradables al ojo humano, pero que para el detector representan ruido semántico, alterando las texturas que emplea para reconocer y localizar objetos. Como consecuencia, la mejora perceptual de

la imagen no garantiza un aumento en la capacidad de detección; existe una disociación entre calidad visual y rendimiento del modelo, ya que recuperar contraste no implica recuperar información relevante para la tarea, y la manipulación puede incluso degradar la precisión. Sin embargo, en escenarios concretos, especialmente cuando la imagen original presenta niveles de oscuridad o bajo contraste que ocultan estructuras relevantes, estas técnicas de mejora pueden incrementar el recall al hacer más visibles objetos que de otro modo pasarían desapercibidos para el detector.

Por tanto, el análisis realizado evidencia que el rendimiento de las predicciones tras la aplicación de algoritmos de mejora depende tanto del tipo de algoritmo como de la intensidad del brillo aplicado a las imágenes.

En este sentido, USM (Unsharp Mask) se destaca como la opción más acertada. Sus resultados bajo condiciones de brillo bajo muestran pérdidas reducidas, lo que lo convierte en el algoritmo con menor impacto negativo. Este comportamiento puede explicarse porque el realce de bordes característico de USM genera modificaciones más localizadas en la imagen, afectando menos al contenido global que otros métodos de ajuste más agresivos. En este sentido, consigue mantener una degradación limitada, lo que lo posiciona como una alternativa relativamente segura en escenarios de iluminación moderada.

Por el contrario, la transformación logarítmica (LT) es el algoritmo con peor desempeño. Sus resultados son especialmente desfavorables en niveles de brillo altos, donde las pérdidas alcanzan los valores más extremos observados. Esta tendencia puede deberse a que la transformación logarítmica comprime intensamente los niveles de intensidad, lo que en condiciones de iluminación alterada puede distorsionar aún más la información útil para el modelo de detección, reduciendo drásticamente su capacidad de reconocimiento.

Los dos restantes, HE (Histogram Equalization) y CLAHE (Contrast Limited Adaptive Histogram Equalization) presentan un comportamiento intermedio. Aunque su efecto es menos perjudicial que el de LT, ambos siguen la misma tendencia de incremento del deterioro conforme aumenta el brillo. Estas técnicas, al redistribuir los niveles de intensidad de la imagen, pueden introducir contrastes artificiales o ruidos locales que, en lugar de facilitar la detección, dificultan la extracción de características relevantes por parte del modelo. El hecho de que CLAHE no muestre mejoras significativas sobre HE refuerza la conclusión de que estas transformaciones no aportan beneficios globales.

Como se ha mencionado anteriormente, cabe resaltar que estos resultados corresponden a promedios globales. Es decir, aunque los algoritmos no ofrecen mejoras sostenidas, existen casos puntuales dentro del conjunto de datos donde proporcionan una corrección parcial y han mejorado la detección. No obstante, dichas mejoras se diluyen al considerar el conjunto completo de datos, lo que limita su aplicabilidad práctica de forma generalizada.

Así, los hallazgos sugieren que la elección de un algoritmo de mejora debe ser cuidadosa y dependiente del contexto. USM podría emplearse como una opción relativamente estable cuando se busque minimizar pérdidas, mientras que LT debería evitarse en escenarios con alta variabilidad de brillo. HE y CLAHE, aunque menos extremos, tampoco demuestran ventajas claras.

Otro aspecto importante es la interacción entre los distintos factores evaluados. El rendimiento del sistema no depende únicamente del modelo o del algoritmo aplicado, sino de la combinación entre tamaño del modelo, tipo de imagen y técnica de mejora. Esta interacción compleja sugiere que no existe una solución universal, sino que es necesario adaptar la configuración del sistema a la circustancia específica de aplicación. Por ejemplo, en entornos con iluminación controlada, puede bastar con modelos más ligeros, mientras que en escenarios dinámicos o impredecibles, será necesario recurrir a modelos más robustos y aplicar técnicas de preprocesamiento que compensen las deficiencias de la imagen.

En conjunto, los resultados permiten concluir que la iluminación constituye un factor determinante en la detección de objetos y que su modificación repercute de manera generalizada en el desempeño del modelo. Los métodos de mejora de imágenes proporcionan una recuperación parcial del rendimiento, pero su eficacia es limitada y depende de diversos factores. Esto se explica por dos razones: los algoritmos de mejora actúan sobre la intensidad y el contraste de la imagen, pudiendo introducir ruido o alterar patrones espaciales relevantes, y los sistemas de detección no interpretan la información visual de la misma manera que los humanos, por lo que lo que se percibe como "mejorado" visualmente no necesariamente incrementa la precisión del modelo. La sensibilidad del rendimiento varía según la clase de objeto, la resolución de la imagen y la arquitectura del modelo. Asimismo, mejorar la apariencia visual de la imagen no garantiza la recuperación completa del desempeño original, destacando la importancia de estrategias complementarias, como entrenar con condiciones de iluminación adversas o emplear mecanismos de atención que compensen la pérdida de información relevante.

8

Conclusiones y Líneas Futuras

8.1. Conclusiones

Este Trabajo de Fin de Grado ha tenido como objetivo principal investigar cómo afectan las variaciones de brillo en imágenes a la capacidad de detección de objetos por parte del modelo YOLOv11 en concreto. Para ello, se ha diseñado un enfoque experimental que permite evaluar de manera ordenada cómo responde el modelo a diferentes condiciones de brillo, utilizando tanto las imágenes originales como versiones ajustadas con distintos niveles de iluminación o algoritmos de mejora. Se han generado datasets modificados con distintos grados de brillo o con distintos algoritmos aplicados y se ha evaluado el rendimiento del modelo en cada caso mediante métricas como precisión, recall y las medias de cada una de ellas.

Los análisis realizados permiten sintetizar el comportamiento del modelo frente a distintas condiciones de brillo y algoritmos de mejora, así como comparar su desempeño con el rendimiento alcanzado sobre las imágenes originales sin alteraciones. Los resultados muestran que, en condiciones óptimas, el modelo mantiene niveles significativamente más altos de precisión media (AP) y recall promedio (AR), lo que evidencia que la arquitectura YOLOv11 es capaz de detectar la mayoría de los objetos presentes con gran exactitud y cobertura. Al introducir variaciones de brillo, se observa una disminución progresiva del rendimiento: los métodos de mejora pueden aumentar parcialmente la cobertura y la precisión, pero ninguno logra igualar completamente los valores originales. Los modelos más grandes conservan un rendimiento superior y una mayor estabilidad frente a estas alteraciones de iluminación, mientras que los modelos pequeños presentan caídas más pronunciadas, reflejando una mayor sensibilidad tanto en precisión como en recall ante condiciones adversas, especialmente cuando se trata

de objetos pequeños. Esta evidencia cuantitativa y cualitativa confirma que la iluminación es un factor determinante en la detección de objetos, que la recuperación mediante algoritmos de mejora es parcial y depende del tamaño y la arquitectura del modelo, y que el rendimiento óptimo solo se alcanza cuando las imágenes mantienen sus condiciones originales de luminosidad.

Por otro lado, es relevante indicar que los objetivos establecidos en el anteproyecto se han alcanzado por completo, asegurando que el trabajo desarrollado cumple con las metas previstas inicialmente, de forma que, aparte de los dos algoritmos mencionados que se incluirían, como son la ecualización del histograma (HE) y la ecualización del histograma adaptativa limitada por contraste (CLAHE), se ha estudiado el efecto de los algoritmos de máscara de enfoque y transformación logarítmica. Además, se fijó que se estudiaría el efecto de una reducción de brillo entre los valores aleatorios de 3, 4 y 5 aunque, finalmente, se han añadido los valores fijos 3, 5 y 10, aparte del ya mencionado.

La importancia de este trabajo radica en que aporta evidencia empírica sobre una limitación concreta de los modelos de detección actuales: su vulnerabilidad ante variaciones de brillo. En aplicaciones reales, como la videovigilancia, la robótica o la asistencia médica, las condiciones de iluminación no siempre son óptimas, y comprender cómo se comportan los modelos en estos escenarios es esencial para mejorar su fiabilidad. Este TFG contribuye a ese entendimiento desde una perspectiva experimental, sin modificar el modelo, sino analizando su respuesta ante distintos estímulos controlados en las imágenes.

Durante el desarrollo del proyecto se han encontrado diversas dificultades. Una de las principales ha sido la gestión de los datos, especialmente al generar versiones modificadas de los conjuntos de imágenes sin perder la coherencia en las anotaciones. También ha sido complejo interpretar los resultados cuando las métricas mostraban variaciones sutiles, lo que ha exigido un análisis estadístico cuidadoso. Por último, el uso de la plataforma Google Colab implicó algunas limitaciones técnicas, especialmente relacionadas con la disponibilidad limitada de GPU. Estas restricciones se han gestionado mediante una planificación cuidadosa y la optimización del código, aprovechando al máximo los recursos disponibles en cada sesión.

A nivel académico, este TFG ha permitido aplicar conocimientos de visión por computador, programación y metodología científica. Profesionalmente, ha brindado la oportunidad de familiarizarse con el diseño de experimentos, la gestión de datos y la evaluación crítica de

modelos de inteligencia artificial. A nivel personal, ha supuesto un reto estimulante que ha reforzado mi capacidad de análisis, mi autonomía investigadora y mi interés por la investigación aplicada en inteligencia artificial.

En resumen, el trabajo ha cumplido con los objetivos previstos y ha aportado una contribución al estudio del comportamiento de modelos de detección ante variaciones de brillo. Las conclusiones obtenidas permiten entender mejor las limitaciones del modelo y abren la puerta a futuras investigaciones orientadas a mejorar su robustez. Este TFG ha constituido una experiencia formativa completa, representando un avance significativo en mi crecimiento académico y profesional.

8.2. Líneas futuras

Los resultados obtenidos en este trabajo permiten abrir múltiples líneas de investigación complementarias que podrían enriquecer los hallazgos presentes, tanto a nivel experimental como teórico.

En primer lugar, una posible extensión consistiría en incorporar el análisis de otras formas de degradación de calidad visual además del brillo. Si bien en este estudio se ha centrado la atención en la luminosidad debido a su relevancia práctica y su impacto directo en la percepción de los objetos en imágenes, existen numerosos factores que pueden afectar negativamente la calidad de una imagen y, en consecuencia, el rendimiento de los modelos de detección. Entre estos factores se encuentran el ruido digital, especialmente común en entornos con iluminación escasa o el desenfoque, tanto por movimiento como por errores de enfoque. La evaluación del comportamiento de los modelos bajo estas condiciones permitiría una caracterización más completa de su robustez.

Asimismo, otra línea prometedora consiste en ampliar el análisis a otros modelos de detección de objetos diferentes a YOLO. Aunque esta arquitectura ha sido elegida por su equilibrio entre precisión y velocidad, existen alternativas ampliamente reconocidas en la comunidad de visión por computador que podrían servir como puntos de comparación, tales como Faster R-CNN, SSD o modelos más recientes como DETR. Evaluar la sensibilidad de cada uno de estos modelos frente a las degradaciones lumínicas o su capacidad para aprovechar las técnicas de mejora aplicadas permitiría establecer patrones comunes o diferenciales entre arquitecturas, lo cual resulta valioso para futuras aplicaciones donde se deba elegir el modelo más adecuado

en función del entorno.

Otro aspecto relevante para futuras investigaciones es la aplicación de este enfoque a escenarios reales, más allá del entorno controlado proporcionado por el conjunto de datos COCO. Si bien COCO ofrece una base sólida y estandarizada para la comparación de modelos, su naturaleza estática limita el análisis de problemas reales donde las condiciones de captura de imagen son cambiantes e impredecibles. Llevar a cabo estudios en entornos reales, como sistemas de videovigilancia, cámaras embarcadas en vehículos o dispositivos médicos en hospitales, permitiría comprobar si los modelos mantienen su rendimiento cuando se enfrentan a variaciones no previstas de iluminación, movimiento o ruido. Además, este enfoque facilitaría la validación de los algoritmos de mejora propuestos, analizando su aplicabilidad práctica y su impacto en tareas críticas.

Finalmente, una línea adicional de investigación podría centrarse en el desarrollo de un predictor de sensibilidad. Diseñar un regresor capaz de estimar el impacto de distintas condiciones de iluminación y degradaciones visuales sobre el rendimiento del modelo permitiría anticipar la eficacia de algoritmos de mejora y ajustar dinámicamente parámetros de detección para cada imagen de manera individual. Este enfoque permitiría identificar de forma automática qué técnica de mejora proporcionará el mejor rendimiento para cada fotografía, optimizando los resultados sin necesidad de pruebas exhaustivas.

En conjunto, todas estas líneas de investigación representan oportunidades claras para profundizar en el estudio de la robustez de los modelos de detección ante degradaciones visuales y seguir avanzando hacia sistemas de visión artificial más precisos, adaptativos y confiables en condiciones del mundo real.

Referencias

[1] Wikipedia, *Inteligencia artificial*, Wikipedia, La enciclopedia libre, [Internet; acceso: 10-agosto-2025], 2025. dirección: https://es.wikipedia.org/w/index.php?title=Inteligencia_artificial&oldid=169405701.

[2] PRTR, *Qué es la Inteligencia Artificial (IA)*, planderecuperacion.gob.es, [Internet; acceso: 9-agosto-2025], 2025. dirección: https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr.

[3] Tecnobits, *Machine Learning y Deep Learning*, tecnobits.com, [Internet; acceso: 9-agosto-2025], 2025. dirección: https://tecnobits.com/machine-learning-y-deep-learning/.

[4] B. UOC, *Diferencias entre Machine Learning y Deep Learning*, blogs.uoc.edu, [Internet; acceso: 12-agosto-2025], 2025. dirección: https://blogs.uoc.edu/informatica/es/machine-learning-vs-deep-learning-diferencias/.

[5] Intersog, *Diferencias entre Machine Learning y Deep Learning*, intersog.co.il, [Internet; acceso: 12-agosto-2025], 2025. dirección: https://intersog.co.il/blog/technologies/ais-powerful-duo-difference-between-machine-learning-and-deep-learning/.

[6] AprendeIA, *Aprendizaje Supervisado vs No Supervisado*, aprendeia.com, [Internet; acceso: 12-agosto-2025], 2025. dirección: https://aprendeia.com/2018/06/15/diferencia-entre-aprendizaje-supervisado-y-no-supervisado/.

[7] IBM, *Aprendizaje Semi-Supervisado*, ibm.com, [Internet; acceso: 12-agosto-2025], 2025. dirección: https://www.ibm.com/es-es/think/topics/semi-supervised-learning.

[8] O. UAEH, *Aprendizaje por Refuerzo*, otech.uaeh.edu.mx, [Internet; acceso: 12-agosto-2025], 2025. dirección: https://otech.uaeh.edu.mx/noti/index.php/ia/conceptos-de-inteligencia-artificial-que-es-el-aprendizaje-por-refuerzo/.

[9] I. S. Fe-CONICET, *Hiperparámetros*, ingar.santafe-conicet.gov.ar, [Internet; acceso: 12-agosto-2025], 2025. dirección: http://www.ingar.santafe-conicet.gov.ar/convocatoria/hiper-parametros/.

[10] Geekflare, *Redes Neuronales*, geekflare.com, [Internet; acceso: 13-agosto-2025], 2025. dirección: https://geekflare.com/es/neural-networks/.

[11] JuanDRH, *Redes Neuronales*, juandrh.github.io, [Internet; acceso: 13-agosto-2025], 2021. dirección: https://juandrh.github.io/blog/deep.learning/2021/07/30/Redes-Neuronales.html.

[12] ResearchGate, *Funciones de Activación (Sigmoid, ReLU, Softmax)*, researchgate.net, [Internet; acceso: 15-agosto-2025], 2025. dirección: https://www.researchgate.net/figure/Plot-of-the-sigmoid-ReLU-and-softmax-activation-functions_fig3_371831710.

[13] M. Ali, *Introducción a las funciones de activación en las redes neuronales*, datacamp.com, [Internet; acceso: 13-agosto-2025], 2024. dirección: https://www.datacamp.com/es/tutorial/introduction-to-activation-functions-in-neural-networks.

[14] Spotfire, *Qué es una Red Neuronal*, spotfire.com, [Internet; acceso: 15-agosto-2025], 2025. dirección: https://www.spotfire.com/glossary/what-is-a-neural-network.

[15] A. AWS, *Qué es una Red Neuronal*, aws.amazon.com, [Internet; acceso: 15-agosto-2025], 2025. dirección: https://aws.amazon.com/es/what-is/neural-network/.

[16] C. Bits, *Ejemplo de Imagen Digital*, codificandobits.com, [Internet; acceso: 17-agosto-2025], 2025. dirección: https://codificandobits.com/blog/redes-convolucionales-introduccion/.

[17] M. Jiménez, *Curso CNN*, migueljimenezg.github.io, [Internet; acceso: 15-agosto-2025], 2025. dirección: https://migueljimenezg.github.io/cursos/Machine%20Learning/Deep%20Learning/CNN/CNN.html.

[18] IBM, *Redes Neuronales Convolucionales*, ibm.com, [Internet; acceso: 16-agosto-2025], 2025. dirección: https://www.ibm.com/es-es/think/topics/convolutional-neural-networks.

[19] GIMP, *Visión Artificial y CNN*, docs.gimp.org, [Internet; acceso: 16-agosto-2025], 2019. dirección: https://docs.gimp.org/2.8/es/plug-in-convmatrix.html.

[20] A. Vázquez, *Clasificación de imágenes con redes profundas*, medium.com, [Internet; acceso: 16-agosto-2025], 2025. dirección: https://medium.com/@alejandravz/clasificaci%C3%B3n-de-im%C3%A1genes-con-redes-profundas-202db1646d64.

[21] YouTube, *Video sobre CNN*, YouTube, [Vídeo en línea; acceso: 16-agosto-2025], 2025. dirección: https://www.youtube.com/watch?v=3h1_-AaVjWY&t=2s.

[22] Velog, *Imagen de arquitectura CNN*, velog.velcdn.com, [Internet; acceso: 17-agosto-2025], 2025. dirección: https://velog.velcdn.com/images/kimsoohyun/post/b394bb3e-2da7-4ea9-976b-3175c810e263/image.jpg.

[23] Ultralytics, *Glosario IoU*, ultralytics.com, [Internet; acceso: 17-agosto-2025], 2025. dirección: https://www.ultralytics.com/es/glossary/intersection-over-union-iou.

[24] Python.org, *Documentación Oficial de Python*, docs.python.org, [Internet; acceso: 17-agosto-2025], 2025. dirección: https://docs.python.org/es/3/tutorial/.

[25] L. World, *Python Logo*, logos-world.net, [Internet; acceso: 22-agosto-2025], 2025. dirección: https://logos-world.net/python-logo/.

[26] PyTorch, *PyTorch*, github.com, [Internet; acceso: 25-agosto-2025], 2025. dirección: https://github.com/pytorch/pytorch.

[27] PyTorch, *PyTorch*, proyectoidis.org, [Internet; acceso: 25-agosto-2025], 2025. dirección: https://proyectoidis.org/pytorch/.

[28] IBM, *PyTorch*, ibm.com, [Internet; acceso: 25-agosto-2025], 2025. dirección: https://www.ibm.com/es-es/topics/pytorch.

[29] T. Blog, *Google Colab - Easy Way to Learn and Use TensorFlow*, blog.tensorflow.org, [Internet; acceso: 25-agosto-2025], 2018. dirección: https://blog.tensorflow.org/2018/05/colab-easy-way-to-learn-and-use-tensorflow.html.

[30] HostGator, *Google Colab*, hostgator.mx, [Internet; acceso: 27-agosto-2025], 2025. dirección: https://www.hostgator.mx/blog/google-colab/.

[31] C. Dataset, *COCO Dataset*, cocodataset.org, [Internet; acceso: 27-agosto-2025], 2025. dirección: https://cocodataset.org/#detection-eval.

[32] Wikipedia, *Logo de LaTeX*, Wikipedia, [Internet; acceso: 28-agosto-2025], 2025. dirección: https://es.wikipedia.org/wiki/LaTeX.

[33] L. Project, *Sitio Oficial de LaTeX*, latex-project.org, [Internet; acceso: 1-septiembre-2025], 2025. dirección: https://latex-project.org/.

[34] C. G. Valenzuela, *¿Qué es LaTeX y cómo funciona esta útil herramienta para crear documentos?* computerhoy.20minutos.es, [Internet; acceso: 1-septiembre-2025], 2022. dirección: https://computerhoy.20minutos.es/tecnologia/latex-como-funciona-util-herramienta-crear-documentos-1165366.

[35] Wikipedia, *Unsharp Masking*, Wikipedia, [Internet; acceso: 1-septiembre-2025], 2024. dirección: https://en.wikipedia.org/wiki/Unsharp_masking.

[36] A. Support, *Unsharp Mask filter in Motion*, support.apple.com, [Internet; acceso: 2-septiembre-2025], 2023. dirección: https://support.apple.com/es-us/guide/motion/motn169f7953/mac.

[37] R. Depaoli, L. A. Fernández y D. Diaz, "Optimización de la ecualización del histograma en el procesamiento de imágenes digitales," Departamento de Ingeniería e Investigaciones Tecnológicas, Universidad Nacional de La Matanza, Florencio Varela 1903, San Justo, Provincia de Buenos Aires, Argentina, Informe técnico, 2025, [Internet; acceso: 1-septiembre-2025]. dirección: https://sedici.unlp.edu.ar/bitstream/handle/10915/21082/51.pdf?sequence=1.

[38] S. Easily, *¿Qué es la Ecualización del Histograma? Mejora el Contraste de la Imagen*, es.statisticseasily.com, [Internet; acceso: 1-septiembre-2025], 2025. dirección: https://es.statisticseasily.com/glosario/%C2%BFQu%C3%A9-es-la-ecualizaci%C3%B3n-del-histograma%3F-Mejora-el-contraste-de-la-imagen./.

[39] T. A. Learner, *CLAHE*, theailearner.com, [Internet; acceso: 1-septiembre-2025], 2019. dirección: https://theailearner.com/tag/clahe/.

[40] S. Topics, *Log Transformation*, sciencedirect.com, [Internet; acceso: 3-septiembre-2025], 2025. dirección: https://www.sciencedirect.com/topics/computer-science/log-transformation.

[41] N. Castellón, *La transformación logarítmica para el análisis*, linkedin.com, [Internet; acceso: 2-septiembre-2025], 2025. dirección: https://www.linkedin.com/posts/naren-castellon-1541b8101_la-transformaci%C3%B3n-logar%C3%ADtmica-para-el-an%C3%A1lisis-activity-7192269236541562882-9uX2/.

[42] J. A. Rodríguez-Rodríguez, M. A. Molina-Cabello, R. Benítez-Rochel y E. López-Rubio, "The effect of image enhancement algorithms," en *Proceedings of the International Conference*, [Internet; acceso: 3-septiembre-2025], ene. de 2021, págs. 3084-3089. DOI: 10.1109/ICPR48806.2021.9412110.

[43] Wikipedia, *Método Científico*, Wikipedia, [Internet; acceso: 1-septiembre-2025], 2025. dirección: https://es.wikipedia.org/wiki/M%C3%A9todo_cient%C3%ADfico.

[44] Wikipedia, *Desarrollo Iterativo y Creciente*, Wikipedia, [Internet; acceso: 1-septiembre-2025], 2025. dirección: https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente.

[45] M. S. Gráficos, *Diseño Iterativo — La Metodología que Perfeccionará tus Proyectos*, medium.com, [Internet; acceso: 1-septiembre-2025], 2023. dirección: https://medium.com/sue%C3%B1os-graficos/dise%C3%B1o-iterativo-la-metodolog%C3%ADa-que-perfeccionar%C3%A1-tus-proyectos-21034b0d277e.

[46] J. A. Rodríguez-Rodríguez, M. A. Molina-Cabello, R. Benítez-Rochel y E. Lopez-Rubio, "The effect of image enhancement algorithms on convolutional neural networks," en *2020 25th International Conference on Pattern Recognition (ICPR)*, [Internet; acceso: 2-septiembre-2025], IEEE, Milan, Italy, ene. de 2021, págs. 3084-3088. DOI: 10.1109/ICPR48806.2021.9412110. dirección: https://doi.org/10.1109/ICPR48806.2021.9412110.

[47] A. Rosebrock, *Intersection over Union (IoU) for object detection*, pyimagesearch.com, [Internet; acceso: 3-septiembre-2025], 2016. dirección: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

Apéndice A
Manual de instalación

A.1. Introducción

Este proyecto está diseñado para ejecutarse en **Google Colab**, lo que simplifica la instalación y evita problemas de compatibilidad de hardware o software en el equipo local. Google Colab proporciona un entorno preconfigurado con acceso a GPUs y librerías comunes, permitiendo ejecutar el proyecto con mucha más comodidad.

A.2. Requisitos previos

- Contar con una **cuenta de Google** con acceso a Colab.
- Conexión a internet estable.
- Activar el uso de **GPU en Colab** para acelerar las inferencias (Entorno de ejecución → Cambiar tipo de hardware → GPU).
- Descargar el archivo ZIP del proyecto proporcionado y tenerlo disponible para subirlo a Colab.

A.3. Cargar el proyecto en Colab

1. Abrir Google Colab desde un navegador.
2. Importar o abrir el notebook proporcionado en el ZIP (extensión .ipynb).
3. Importar todos los demás archivos del proyecto a Colab o subir el ZIP al completo y descomprimirlo ejecutando el siguiente código en una celda, específicamente en la carpeta "/content/" para mayor comodidad:

```
import zipfile
import os

# Ruta del archivo ZIP y carpeta de destino
zip_path = "/content/proyectoTFG.zip"
destino = "/content/"

# Crear la carpeta de destino si no existe
os.makedirs(destino, exist_ok=True)

# Abrir y extraer todo
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(destino)

print(f"Archivo descomprimido en {destino}")
```

Este proceso creará una carpeta con todos los archivos necesarios para ejecutar el proyecto.

A.4. Instalación de dependencias

El proyecto incluye un archivo **requirements.txt** con las versiones exactas de las librerías utilizadas. Para instalar todas las dependencias, ejecutar la primera celda del proyecto que hace lo siguiente:

```
!pip install -r requirements.txt
```

Es importante instalar estas dependencias antes de ejecutar cualquier celda del notebook para evitar errores de compatibilidad. A continuación, ya se podría ejecutar cada una de las celdas siguiendo el flujo definido.

A.5. Estructura del proyecto

Por otro lado, la estructura del proyecto es la siguiente y son todos los archivos que deben subirse a Colab para que el código funcione correctamente:

- **Desarrollo TFG María Jesús García.ipynb** → Notebook principal con todo el flujo de trabajo: carga de datos, predicciones, evaluación y visualización.

- **setup_utils.py** → Descarga y preparación del dataset COCO 2017.
- **dataloader.py** → Funciones para crear DataLoaders para las imágenes originales.
- **enhance.py** → Funciones para aplicar las transformaciones a las imágenes, tanto la disminución de brillo como los distintos algoritmos.
- **pred.py** → Función para realizar las predicciones y almacenarlas en formato JSON compatible con COCO.
- **procesar_jsons.py** → Función para calcular métricas AP/AR y generar archivo CSV con los resultados.
- **generar_heatmaps.py** → Función para crear heatmaps comparativos del rendimiento entre modelos y transformaciones.
- **mostrar.py** → Funciones para visualizar predicciones sobre imágenes originales y transformadas.
- **valid.py** → Función para evaluar predicciones utilizando COCOeval.
- **requirements.txt** → Lista de dependencias necesarias para ejecutar el proyecto.
- **analisis_resultados.py** → Funciones **analisis_metricas** y **analisis_media_ap** para comparar algoritmos frente al caso base (AlgNone), calcular tasas de mejora/empeoramiento, balances de impacto y ratios beneficio-riesgo, generando resúmenes en CSV.

A.6. Notas adicionales

- Se recomienda ejecutar todas las celdas del notebook en orden para asegurar la reproducibilidad de los resultados.
- El uso de GPU es opcional pero altamente recomendable para reducir el tiempo de ejecución de los modelos.

Apéndice B
Manual de usuario

El notebook principal está organizado en **celdas con tareas específicas**. Cada celda llama a funciones de los scripts correspondientes. Esto permite separar la preparación de datos, predicciones, evaluación y visualización, haciendo el flujo reproducible, modular y fácil de mantener. A continuación, se describirán cada una de ellas y se incluirán extractos de código de las mismas.

B.1. Celda 1: Instalación de dependencias

```
pip install -r requirements.txt
```

Su función es instalar todas las librerías necesarias para ejecutar el proyecto.

B.2. Celda 2: Preparación del dataset

```
from dataloader import get_coco_dataloader
from setup_utils import download_and_extract_coco

print("\nDescargando datos...")
download_and_extract_coco()

dataset, dataloader_base, coco_data_base, path2json = get_coco_dataloader()
coco_gt = coco_data_base.coco
```

- **Función:** Descarga el conjunto de datos, prepara el dataloader para procesarlo posteriormente y devuelve el objeto COCO original para evaluación.
- **Parámetros:** Batch size y shuffle dentro de **get_coco_dataloader()**.
- **Resultado:** Dataset listo para predicciones.

B.3. Celda 3: Configuración de experimentos

```
usar_brillo_aleatorio = False
brillo_valor_fijo = 10
tamaños_modelo = ['n', 's', 'm']

algoritmos = [
    {"id": 1, "param": None},
    {"id": 2, "param": [3.5, 8]},
    {"id": 4, "param": [0.3, 3]},
    {"id": 6, "param": None}
]
```

- **Función:** Define parámetros generales de los experimentos.
- **Parámetros:**
  - **usar_brillo_aleatorio**: Si se activa, el brillo se aplica de forma aleatoria a cada imagen con un valor entre 3, 4 y 5; si no, se aplica un brillo fijo definido por **brillo_valor_fijo**.
  - **brillo_valor_fijo**: Número entero que indica el valor de brillo cuando no se usa brillo aleatorio.
  - **tamaños_modelo**: Tamaños YOLO a evaluar. Puede ser sólo uno o una lista de varios.
  - **algoritmos**: Lista de algoritmos de mejora y parámetros.

```
brillo_nombre = "bAlea" if usar_brillo_aleatorio else f"b{brillo_valor_fijo
    }"
path_datos_brillo = "/content/dism_brillo_alea/" if usar_brillo_aleatorio
    else f"/content/dism_brillo_b{brillo_valor_fijo}/"

print(f"Aplicando {'BRILLO ALEATORIO' if usar_brillo_aleatorio else f'
    BRILLO FIJO con valor: {brillo_valor_fijo}'}")

dataloader_brillo, coco_brillo = aplicar_y_guardar_brillo(
```

```
    path2json=path2json,
    brillo_alea=usar_brillo_aleatorio,
    brillo_valor=brillo_valor_fijo,
    save=False,
    save_dir=path_datos_brillo
)
```

Los dataloaders de cada algoritmo se crean de la misma forma. A continuación, una vez definidos los parámetros, se crea el dataloader que contiene las imágenes con la disminución de brillo aplicada y se pueden guardar las imágenes estableciendo **save=True**.

```
nombre_pred = f"pred_{tamaño}_bNone_AlgNone"
ruta_json = f"/content/Jsons/{nombre_pred}.json"
carpeta_pred = f"predictions/{nombre_pred}"

print(f"[1] Predicción original: {nombre_pred}")
realizar_predicciones(
    model=model,
    dataloader=dataloader_base,
    coco_data=coco_data_base,
    output_path=ruta_json,
    model_size=tamaño,
    brillo="bNone",
    algoritmo="AlgNone"
)

val = validar(ruta_json, coco_gt)
```

Por último, en esta celda se realizan las predicciones sobre los tres conjuntos de imágenes: primero sobre las imágenes originales, sin aplicar brillo ni algoritmos; luego sobre las imágenes con brillo reducido, que puede ser aleatorio o fijo según la configuración y, finalmente, sobre las imágenes con los algoritmos de mejora ya aplicados.

Todos estos bloques tienen la misma estructura y se controlan mediante un bucle externo que itera sobre los tamaños de modelo YOLO definidos y un bucle interno que recorre los algoritmos configurados.

Además, la llamada a la función de validación calcula las métricas automáticamente a partir del JSON generado y las anotaciones COCO para luego comparar los efectos del brillo y los algoritmos de mejora sobre la precisión y el recall del modelo.

- **Parámetros:**
  - **model**: Modelo YOLO a usar, que se carga automáticamente según el bucle destinado a los distintos tamaños de YOLO.
  - **dataloader**: DataLoader con las imágenes a predecir, que según el caso es el específico generado para ese conjunto de imágenes (originales, con brillo o transformadas).
  - **coco_data**: Objeto COCO asociado al DataLoader.
  - **output_path**: Ruta de salida del JSON; se genera a partir de **nombre_pred**, pero se puede modificar manualmente.
  - **model_size**: Tamaño del modelo YOLO a evaluar; afecta precisión y velocidad de predicción.
  - **brillo**: Indica si se aplica brillo a las imágenes.
  - **algoritmo**: Indica si se aplica algún algoritmo de mejora.

B.4. Celda 4: Procesamiento de JSONs a CSV

```
carpeta_origen="/content/Jsons/"
carpeta_destino="/content/resultados.csv"

procesar_jsons(coco_gt,json_folder=carpeta_origen,output_csv=
    carpeta_destino)
```

- **Función:** Calcula métricas a partir de todos los JSON y genera un CSV resumen.
- **Parámetros:** Carpeta de JSONs y archivo CSV de salida.

B.5. Celda 5: Generación de heatmaps

```
from generar_heatmaps import generar_heatmaps
generar_heatmaps(csv_path="/content/resultados.csv",
                 output_dir="/content/heatmaps",
                 guardar=True)
```

- **Función:** Genera mapas de calor para comparar rendimiento entre modelos y transformaciones.
- **Parámetros:** CSV de métricas, carpeta de salida y booleano para indicar si se quieren almacenar o no.

B.6. Celda 6: Visualización de predicciones

```
from mostrar import iniciar
mostrar_predicciones_multiple = iniciar(
    coco_gt,
    ruta_original="/content/YOLO Data/val2017/",
    ruta_brillo="/content/YOLO Data/val2017/",
    ruta_algoritmo1="/content/YOLO Data/val2017/",
    json_pred1="/content/pred_n_bNone_Alg1.json",
    # demás JSONs...
)

# Se pueden visualizar en orden, por nombre o aleatoriamente
# mostrar_predicciones_multiple(cantidad=10)
mostrar_predicciones_multiple(nombre_imagen="000000000139.jpg")
# mostrar_predicciones_multiple(cantidad=10, aleatorias=True)
```

- **Función:** Visualiza bounding boxes, categoría del objeto y confianza sobre las imágenes indicadas.
- **Parámetros:** Los parámetros necesarios son el objeto COCO, las rutas a las imágenes a mostrar, que deben haber sido guardadas previamente al generar los dataloaders y las rutas a los archivos JSON cuyas detecciones queremos representar. Además, se puede mostrar una imagen específica o varias, visualización aleatoria o por identificador.

- **Resultado:** Comparación visual de predicciones.

B.7. Celda 7: Análisis de métricas relativas

```
from analisis_resultados import analisis_metricas_relativas,
    analisis_media_ap_relativa

# Ejecutar análisis general de métricas relativas
analisis_metricas_relativas("resultados_heatmap.csv")

# Ejecutar análisis detallado solo con media_AP relativa
res_df_rel = analisis_media_ap_relativa("resultados_heatmap.csv")
display(res_df_rel)  # para ver en tabla bonita en Colab
```

- **Función:** Calcula la diferencia relativa de todas las métricas respecto al caso base sin la aplicación de ningún algoritmo y genera un análisis resumido por algoritmo y brillo.
- **Parámetros:**
  - csv_path: Ruta al archivo CSV que contiene los resultados de detección y métricas.
  - Opcionalmente, se pueden modificar funciones internas para elegir métricas específicas.
- **Resultado:**
  - metricas_relativas.csv: Contiene la diferencia relativa (%) de cada métrica frente al caso base, agrupada por algoritmo y brillo.
    - Balance impacto (%): Promedio de la diferencia relativa.
    - Ganancia relativa media (%): Media de mejora relativa.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA