MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Experiencias en la aplicación de librerías gráficas para el tratamiento automatizado de flujos de video y su aplicación en materia de seguridad en instalaciones

Autor: Inés Blanco de la Puerta

Director: Emilio Manuel Domínguez Adan

Madrid julio 2024

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

‘Experiencias en la aplicación de librerías gráficas para el tratamiento automatizado de flujos de video y su aplicación en materia de seguridad en instalaciones’

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2023/24 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

BLANCO DE LA PUERTA INES - 45338271G

Firmado digitalmente por BLANCO DE LA PUERTA INES - 45338271G
Fecha: 2024.07.21 23:08:25 +02'00'

Fdo.: Inés Blanco de la Puerta Fecha: 15/07/2024

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

DOMINGUEZ ADAN EMILIO MANUEL - 02620715A

Firmado digitalmente por DOMINGUEZ ADAN EMILIO MANUEL - 02620715A
Fecha: 2024.07.22 07:32:51 +02'00'

Fdo.: Emilio Manuel Domínguez Adan Fecha: 15/07/2024

MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Experiencias en la aplicación de librerías gráficas para el tratamiento automatizado de flujos de video y su aplicación en materia de seguridad en instalaciones

Autor: Inés Blanco de la Puerta

Director: Emilio Manuel Domínguez Adan

Madrid julio 2024

EXPERIENCIAS EN LA APLICACIÓN DE LIBRERÍAS GRÁFICAS PARA EL TRATAMIENTO AUTOMATIZADO DE FLUJOS DE VIDEO Y SU APLICACIÓN EN MATERIA DE SEGURIDAD EN INSTALACIONES

Autor: Blanco de la Puerta, Inés
Director: Domínguez Adan, Emilio Manuel
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este proyecto, se ha desarrollado un sistema de detección y seguimiento de objetos en tiempo real utilizando técnicas avanzadas de visión por computadora, librerías gráficas y aprendizaje profundo. El objetivo principal ha sido mejorar la seguridad en instalaciones mediante el análisis automatizado de flujos de video. Para ello, se ha reentrenado un modelo YOLOv8 con un conjunto de datos específico, logrando una detección precisa en diversos escenarios.

Palabras clave: YOLO, visión por computadora, detección de objetos, aprendizaje profundo, seguridad

1. Introducción

El proyecto surge con la intención de mejorar la seguridad en plantas industriales ubicadas en regiones remotas mediante el desarrollo de un sistema automatizado de detección y respuesta ante incidentes. Con la digitalización y la automatización como herramientas clave, se busca optimizar procesos, reducir errores humanos y aumentar la precisión en la supervisión de estas infraestructuras. Además, la creciente sofisticación de amenazas externas subraya la necesidad de soluciones de seguridad más avanzadas y efectivas.

La implementación de librerías gráficas en el tratamiento automatizado de flujos de video se propone como una estrategia para crear un sistema proactivo y eficiente que pueda anticiparse a las amenazas y prevenir incidentes antes de que ocurran. En este contexto, se pretende aplicar algoritmos avanzados de detección de objetos, como YOLO, para mejorar la capacidad de monitoreo y respuesta en tiempo real, asegurando así la integridad de las instalaciones y la continuidad de las operaciones.

2. Definición del proyecto

El objetivo de este proyecto es desarrollar un sistema capaz de analizar flujos de video en tiempo real para detectar y rastrear objetos relevantes, como personas y vehículos, mejorando así la seguridad en infraestructuras críticas. El desarrollo del proyecto se estructura en varias fases. Inicialmente, se realiza una revisión exhaustiva de la literatura para identificar las técnicas más efectivas en la detección de objetos. En esta fase, se comparan los métodos tradicionales de procesamiento de imágenes con técnicas avanzadas como las Redes Neuronales Convolucionales, optando finalmente por YOLOv8 debido a sus ventajas en términos de rapidez y precisión. Posteriormente, se

recolectan y preprocesan imágenes para crear un conjunto de datos diversificado que abarca diferentes condiciones de iluminación, ángulos de cámara y distancias. El modelo se reentrena con este conjunto de datos, optimizando sus parámetros para mejorar el rendimiento en escenarios de vigilancia de seguridad. Finalmente, se prueba el modelo sobre flujos de video en tiempo real para verificar su funcionamiento en un entorno real, asegurando así un sistema eficiente y robusto para aplicaciones de seguridad crítica.

3. Descripción del modelo

El modelo de detección utilizado es una versión reentrenada de YOLOv8, optimizada para reconocer personas y vehículos en diferentes condiciones de iluminación y ángulos de cámara. El reentrenamiento del modelo se realizó utilizando técnicas de transferencia de aprendizaje, aprovechando una red previamente entrenada. Esta técnica permite reducir el tiempo de entrenamiento y mejorar la precisión del modelo en la tarea específica.

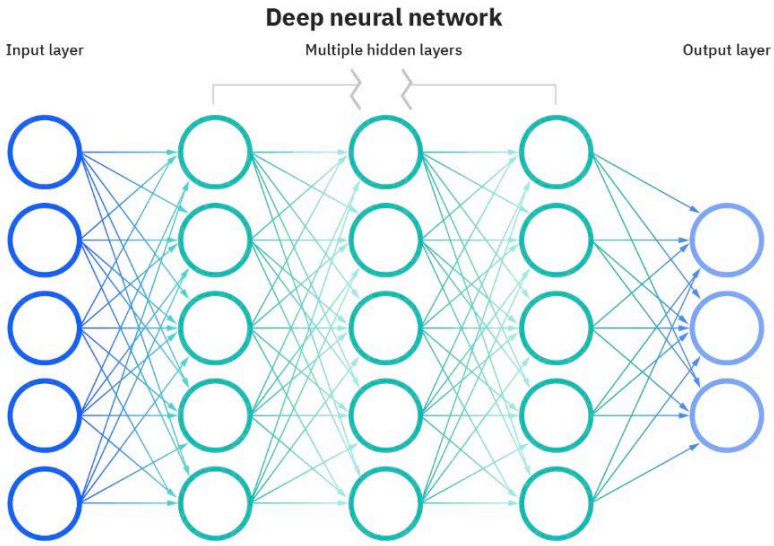


Ilustración 1. Estructura de una red neuronal profunda – CNN (IBM, 2024)

Para entrenar y optimizar el modelo, se ajustan hiperparámetros clave como el tamaño del lote, el número de épocas y la tasa de aprendizaje. Realizamos pruebas experimentales con diferentes configuraciones de estos hiperparámetros para determinar la combinación óptima para maximizar la precisión y eficiencia del modelo. El rendimiento del modelo se evalúa no solo con las métricas resultantes del entrenamiento y la validación, sino también mediante su desempeño en videos de cámaras de seguridad.

El modelo muestra un desempeño sólido en tiempo real, aunque se encuentra con desafíos bajo condiciones de iluminación extrema y cuando los objetos están parcialmente ocultos.

4. Resultados

Los resultados de las pruebas experimentales indicaron que el modelo YOLOv8 reentrenado puede detectar y rastrear objetos con una buena precisión en tiempo real. Durante las pruebas con diferentes configuraciones de hiperparámetros, se observó que un tamaño de lote pequeño y una tasa de aprendizaje alta proporcionaron mejores resultados en términos de precisión media (mAP) del modelo. Sin embargo, al evaluar el modelo con videos de cámaras de seguridad, se obtuvieron mejores resultados con configuraciones de mayor tamaño de lote y menor tasa de aprendizaje. Decidimos utilizar el ensayo con mejor resultado en las pruebas con videos de cámaras de seguridad, ya que reflejan mejor las condiciones reales de uso del modelo.

La evaluación en videos de cámaras de seguridad mostró que el modelo puede mantener una detección consistente y precisa incluso en escenarios dinámicos. No obstante, se identificaron algunas limitaciones bajo condiciones de iluminación adversas, cuando los objetos estaban parcialmente ocultos o demasiado lejos. Además, se realizó una prueba en tiempo real utilizando la cámara del ordenador. Durante esta prueba, el modelo demostró un desempeño sólido, detectando y rastreando objetos con precisión en la mayoría de los casos. Sin embargo, también se encontraron desafíos similares relacionados con la iluminación extrema y la detección de objetos parcialmente ocultos.

Ilustración 2. Capturas de pantalla realizadas durante la prueba de video en tiempo real en distintos ambientes

5. Conclusiones

En conclusión, el desarrollo y reentrenamiento del modelo YOLOv8 han demostrado ser efectivos para mejorar la seguridad en infraestructuras críticas mediante la detección y

el seguimiento de objetos en tiempo real. El proyecto ha logrado desarrollar un sistema robusto que analiza flujos de video, detectando y rastreando objetos con alta precisión y rapidez. Sin embargo, se han identificado áreas de mejora, particularmente en condiciones de iluminación desfavorables, con objetos parcialmente ocultos o demasiado lejos.

Para abordar estas limitaciones, se recomienda ampliar y diversificar el conjunto de datos de entrenamiento para incluir una mayor variedad de condiciones ambientales y perspectivas. La aplicación de técnicas avanzadas de aumento de datos (data augmentation) y ajustes de hiperparámetros puede mejorar la robustez y adaptabilidad del modelo en diferentes escenarios del mundo real. Además, integrar métodos de aprendizaje continuo permitiría que el modelo se adapte y mejore continuamente a medida que procesa más datos del entorno real, aumentando así su precisión y fiabilidad en aplicaciones de vigilancia y seguridad industrial.

En resumen, el modelo YOLOv8 ha demostrado ser eficiente y robusto para aplicaciones de seguridad en tiempo real, proporcionando detecciones consistentes incluso con movimientos rápidos y cambios en el entorno. Las observaciones realizadas subrayan la importancia de continuar optimizando y diversificando el entrenamiento del modelo para asegurar un rendimiento óptimo en una variedad de condiciones reales, asegurando así su eficacia y utilidad en la detección automatizada de incidentes en infraestructuras críticas.

EXPERIENCES IN THE APPLICATION OF GRAPHIC LIBRARIES FOR THE AUTOMATED PROCESSING OF VIDEO STREAMS AND THEIR APPLICATION TO SECURITY IN INSTALLATIONS.

Author: Blanco de la Puerta, Inés.
Supervisor: Domínguez Adan, Emilio Manuel.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In this project, a real-time object detection and tracking system has been developed using advanced computer vision techniques, graphics libraries and deep learning. The main objective has been to improve security in facilities through the automated analysis of video streams. For this purpose, a YOLOv8 model has been retrained with a specific dataset, achieving accurate detection in various scenarios.

Keywords: YOLO, computer vision, object detection, deep learning, security.

1. Introduction

The project arises with the intention of improving safety in industrial plants located in remote regions through the development of an automated incident detection and response system. With digitalization and automation as key tools, the aim is to optimize processes, reduce human errors and increase the accuracy in the monitoring of these infrastructures. In addition, the growing sophistication of external threats underscores the need for more advanced and effective security solutions.

The implementation of graphical libraries in the automated processing of video streams is proposed as a strategy to create a proactive and efficient system that can anticipate threats and prevent incidents before they occur. In this context, it is intended to apply advanced object detection algorithms, such as YOLO, to improve real-time monitoring and response capabilities, thus ensuring the integrity of the facilities and the continuity of operations.

2. Project definition

The objective of this project is to develop a system capable of analyzing video streams in real time to detect and track relevant objects, such as people and vehicles, thus improving security in critical infrastructures. The development of the project is structured in several phases. Initially, an exhaustive literature review is carried out to identify the most effective techniques for object detection. In this phase, traditional image processing methods are compared with advanced techniques such as Convolutional Neural Networks, finally opting for YOLOv8 due to its advantages in terms of speed and accuracy. Subsequently, images are collected and preprocessed to create a diversified dataset covering different illumination conditions, camera angles and distances. The model is retrained with this dataset, optimizing its parameters to improve performance in security surveillance scenarios. Finally, the model is tested on real-time video streams to verify its performance in a real environment, thus ensuring an efficient and robust system for safety-critical applications.

3. Description of the model

The detection model used is a retrained version of YOLOv8, optimized to recognize people and vehicles in different lighting conditions and camera angles. The model retraining was performed using transfer learning techniques, taking advantage of a previously trained network. This technique allows reducing the training time and improving the accuracy of the model in the specific task.

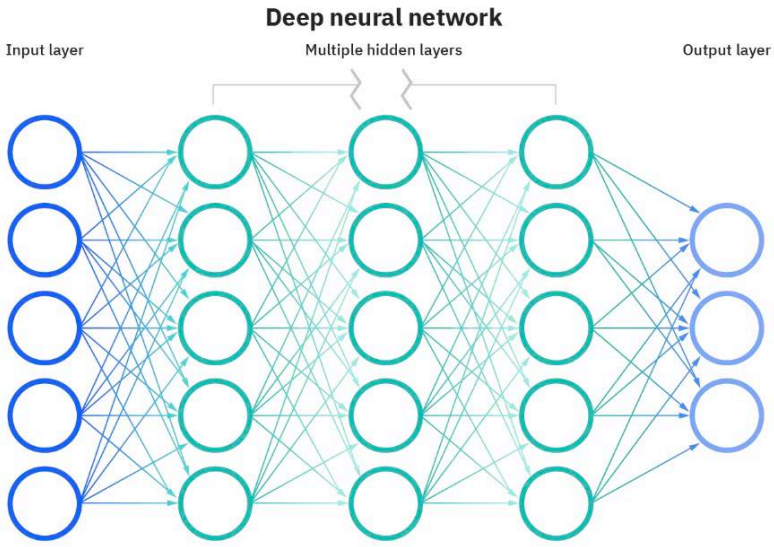


Figure 1. Structure of a deep neural network – CNN (IBM, 2024)

To train and optimize the model, key hyperparameters such as batch size, number of epochs and learning rate are adjusted. We perform experimental tests with different settings of these hyperparameters to determine the optimal combination to maximize model accuracy and efficiency. The model's performance is evaluated not only with the metrics resulting from training and validation, but also by its performance on security camera videos.

The model shows robust performance in real time, although it encounters challenges under extreme lighting conditions and when objects are partially hidden.

4. Results

Experimental test results indicated that the retrained YOLOv8 model can detect and track objects with good real-time accuracy. During testing with different hyperparameter settings, it was observed that a small batch size and high learning rate provided better results in terms of mean accuracy (mAP) of the model. However, when evaluating the model with security camera videos, better results were obtained with larger batch size and lower learning rate configurations. We decided to use the trial with the best result in the tests with security camera videos, since they better reflect the real conditions of use of the model.

The evaluation on security camera videos showed that the model can maintain consistent and accurate detection even in dynamic scenarios. However, some limitations were identified under adverse lighting conditions, when objects were partially hidden or too far away. In addition, a real-time test was performed using the computer camera. During this test, the model demonstrated solid performance, detecting and tracking objects accurately in most cases. However, similar challenges related to extreme lighting and detection of partially hidden objects were also encountered.

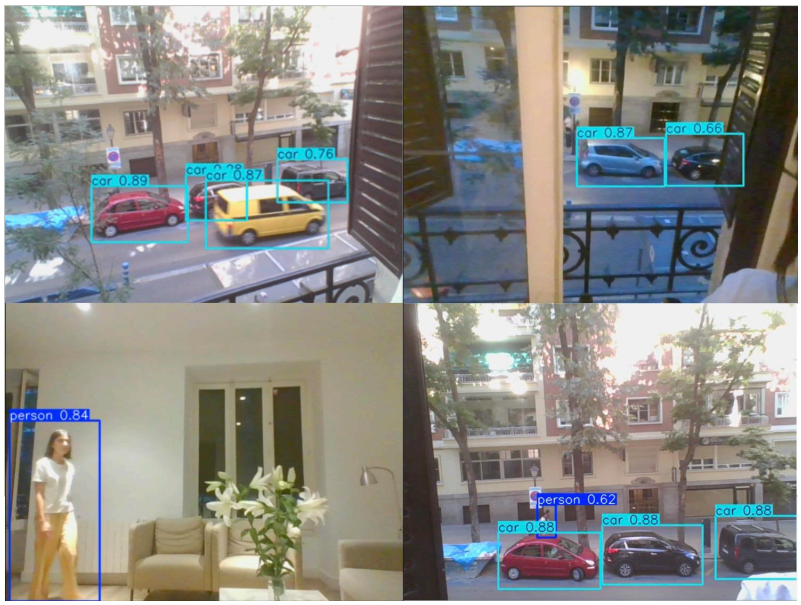


Figure 2. Screenshots taken during real-time video testing in different environments

5. Conclusions

In conclusion, the development and retraining of the YOLOv8 model has proven to be effective in improving critical infrastructure security by detecting and tracking objects in real time. The project has succeeded in developing a robust system that analyzes video streams, detecting and tracking objects with high accuracy and speed. However, areas for improvement have been identified, particularly in unfavorable lighting conditions, with objects partially hidden or too far away.

To address these limitations, it is recommended that the training dataset be expanded and diversified to include a wider variety of environmental conditions and perspectives. The application of advanced data augmentation techniques and hyperparameter adjustments can improve the robustness and adaptability of the model in different real-world scenarios. In addition, integrating continuous learning methods would allow the model to continuously adapt and improve as it processes more data from the real environment, thus increasing its accuracy and reliability in surveillance and industrial safety applications.

In a nutshell, the YOLOv8 model has proven to be efficient and robust for real-time security applications, providing consistent detections even with rapid movements and

changes in the environment. The observations made underscore the importance of continuing to optimize and diversify the training of the model to ensure optimal performance in a variety of real-world conditions, thus ensuring its effectiveness and utility in automated incident detection in critical infrastructures.

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

En la actualidad, la digitalización y automatización de procesos han surgido como poderosas herramientas que transforman tanto la industria como la vida cotidiana de las personas. Estas nuevas tecnologías no solo optimizan la seguridad, eficiencia y la productividad en los entornos industriales, sino que también ofrecen soluciones innovadoras que simplifican y mejoran nuestra calidad de vida diaria.

En la industria, desde la implementación de sistemas robóticos para tareas peligrosas hasta la gestión del tráfico en entornos logísticos, la automatización ha permitido optimizar operaciones, reducir errores humanos y aumentar la precisión en la ejecución de tareas. Sin embargo, en un entorno industrial en constante evolución, la necesidad de avanzar en la automatización es más imperativa que nunca. Con el surgimiento de amenazas externas cada vez más sofisticadas, como ciberataques y riesgos ambientales, es fundamental implementar soluciones de seguridad más avanzadas y eficaces para garantizar la integridad de los entornos físicos y la información crítica. La industria es uno de los sectores que puede llegar a verse más afectado ante la falta de un sistema de seguridad actualizado a las nuevas necesidades.

La motivación del proyecto parte de la idea de desarrollar un sistema capaz de detectar y responder a posibles incidencias en las numerosas plantas eléctricas distribuidas a lo largo de nuestra región. La automatización de esta tarea supondrá un respiro para todas las empresas del sector de la producción eléctrica, o de cualquier otro, que tenga plantas industriales en lugares remotos y poco accesibles.

Se propone aplicar librerías gráficas al tratamiento automatizado de flujos de video, para dar con aquellas que permitan una mejor aplicación en materia de seguridad e instalaciones. Se aspira a crear un sistema inteligente, ágil y proactivo, capaz de anticiparse a las amenazas y prevenir incidentes antes de que ocurran.

1.2 DESCRIPCIÓN DEL PROBLEMA

La necesidad de seguridad ha sido una preocupación fundamental desde las primeras comunidades nómadas que surgen al principio de la humanidad. En un entorno donde los peligros naturales y las amenazas externas estaban siempre presentes, la protección de la vida y de los recursos se convierte en una prioridad. De aquí surge la aparición de guardianes, que vigilaban los asentamientos al llegar la noche, el uso de antorchas, que no solo proporcionaba visibilidad nocturna, sino que también actuaba como elemento disuasorio, y el empleo de animales guardianes. A lo largo de la historia, conforme las sociedades han ido evolucionando, también lo han hecho estos métodos. Desde los fosos que protegían los castillos medievales, hasta el uso de contraseñas y símbolos de identificación o sistemas de alarma mecánicos, que usaban campanas y timbres para generar un sonido que alertara a las personas cercanas.

Hoy en día las cosas han cambiado mucho, estás técnicas se han sustituido por circuitos cerrados de televisión, también conocidos como CCTV, que permiten visualizar en tiempo real y tener registro en video de escenas y situaciones que ocurren en un determinado lugar, desde un centro de control con el objetivo de minimizar los riesgos ante situaciones de seguridad. Estos sistemas también son utilizados para otras aplicaciones, como puede ser el control y supervisión de procesos industriales, el control del tráfico o las clases online en entornos educativos durante la pandemia.

Las principales funciones de un CCTV son la supervisión y la verificación (Rojas Campo, 2022).

- La supervisión se realiza a través de un centro de monitoreo operado por una o varias personas. Estas se encargan de vigilar las imágenes que les llegan de las cámaras en tiempo real. Pudiendo así alertar y apoyar al personal presente en las diferentes localizaciones ante situaciones que se salgan de lo común.
- El CCTV te permite guardar grabaciones pasadas con el fin de poder verificar situaciones extraordinarias después de que el evento en cuestión haya tenido lugar.

Esta validación es esencial para poder comprobar que es lo que ocurrió y tomar medidas para mitigar el riesgo de que vuelva a ocurrir.

Ilustración 1. Centro de monitoreo (Rojas Campo, 2022)

Durante muchos años, un CCTV ha estado compuesto por los siguientes equipos: cámaras de seguridad, equipos o software de administración y almacenamiento de video (DVR o NVR), equipos de visualización, de alimentación o suministro de energía, infraestructura (cableado y conectores) para la interconexión de equipos, y la intervención de personas. Sin embargo, nos encontramos en un momento de la historia en el que la sociedad está en continuo cambio. Los constantes avances tecnológicos, la interconexión global y la digitalización de los procesos industriales nos llevan a una realidad en la que tanto las organizaciones como los individuos estamos expuestos a amenazas mayores. Como consecuencia, aparece la necesidad de implementar soluciones de seguridad más avanzadas y eficaces para garantizar la integridad de los entornos físicos y la información crítica.

La aplicación de tecnologías emergentes como la Inteligencia Artificial (IA) y el Aprendizaje Automático (ML) se propone como una solución integral a la hora de abordar los desafíos contemporáneos en el campo de la seguridad.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Ante el problema de procesamiento de imágenes y flujos de video, actualmente, se están utilizando numerosas bibliotecas gráficas que parecen ofrecer resultados satisfactorios en cuanto a rendimiento y funcionalidades. El objetivo del presente trabajo es profundizar en el uso de estas librerías aplicadas al procesado y post-procesado de videos en una primera fase en modo offline y en segundo lugar sobre flujos de video en protocolo RTSP. Teniendo como fin último, desarrollar un sistema capaz de detectar, clasificar y rastrear objetos/personas para anticiparse a potenciales amenazas.

1.3.2 OBJETIVOS ESPECÍFICOS

El objetivo principal de este trabajo es el estudio y aplicación de técnicas tradicionales de procesamiento de imágenes, así como el de las Redes Neuronales Convolucionales (CNN) para resolver un problema de visión artificial.

Nuestro algoritmo debe ser capaz de llevar a cabo correctamente los siguientes objetivos:

- Dada una imagen de entrada, determinar si hay un objeto presente, localizar su posición y clasificarlo adecuadamente.
- Dado un video, cuyos fotogramas no aparecen en la base de datos utilizada para el entrenamiento, el algoritmo debe ser capaz de detectar, clasificar y seguir el objeto estudiado a lo largo del video.

Cumplir con estos objetivos garantiza que el sistema desarrollado no solo pueda realizar detecciones en imágenes estáticas, sino que también sea capaz de manejar flujos de video en tiempo real. Esto permitirá un seguimiento consistente y preciso de los objetos de interés, demostrando así la eficacia y robustez del sistema en diferentes escenarios y condiciones.

1.4 METODOLOGÍA

El objetivo principal de este trabajo es el estudio y aplicación práctica de diferentes soluciones a un problema de visión por computadora.

- Comenzaremos analizando las herramientas de procesado de imágenes y videos que ofrecen diferentes librerías de Python, realizando un estudio detallado junto con una implementación práctica de estos métodos en videos offline. Este estudio abarcará sus rendimientos, interoperabilidad, aplicaciones y capacidades.
- A continuación, se realizará un análisis exhaustivo de las Redes Neuronales Convolucionales (CNN), seguido por la implementación práctica del tipo de red más adecuada según las especificaciones del proyecto.
- Posteriormente, se llevará a cabo una comparativa entre los resultados obtenidos mediante los métodos tradicionales que ofrecen las librerías de Python y la Red Neuronal Convolucional.
- Por último, se tratará de llevar la primera fase aún más lejos, adaptando el software creado previamente a flujos de video en tiempo real. La finalidad de esta fase consiste en la automatización de tareas de procesado de las imágenes para el reconocimiento de posibles incidencias antes de que estas lleguen a ocurrir.

Capítulo 2. ESTADO DEL ARTE

En este capítulo se pretende revisar los trabajos y soluciones existentes relacionados con el proyecto. Antes de desarrollar cualquier proyecto, es fundamental cuestionarse si existen investigaciones previas con resultados comparables a los objetivos propuestos. A lo largo del capítulo, se explorarán tecnologías y enfoques utilizados en el procesamiento de imágenes y videos, con el objetivo de apoyar y contextualizar el desarrollo del proyecto. Esta revisión permitirá establecer un marco de referencia claro y fundamentado, que guiará la metodología y las decisiones técnicas en las siguientes fases del proyecto.

2.1 INTRODUCCIÓN A LAS LIBRERÍAS GRÁFICAS

Actualmente, se están empleando un gran número de bibliotecas gráficas con resultados aparentemente bastante buenos en materia de rendimiento y prestaciones del procesamiento de imágenes. El auge de estas bibliotecas puede situarse en las últimas dos décadas, con un crecimiento significativo impulsado por avances tecnológicos.

A principios de los años 2000, aparecen las primeras bibliotecas de procesamiento de imágenes de código abierto. Es decir, bibliotecas donde los usuarios tienen la libertad de utilizar la herramienta de forma gratuita, adaptarlas a sus necesidades específicas, colaborar con otros desarrolladores para mejorarlas y contribuir al avance de la visión por computadora.

Aquí surge OpenCV (Open Source Computer Vision Library), herramienta que se convirtió en fundamental para la investigación y el desarrollo en visión por computadora. Esta ofrece una amplia gama de algoritmos y funciones para el procesamiento de imágenes y videos. Desde filtrados y transformaciones, a manipulación de contornos y formas, y operaciones con pixeles para ajustar el brillo y contraste (Stallman, 2022).

En los años siguientes, con el crecimiento de la comunidad que utilizaba estas herramientas y, por ende, el aumento en la demanda de aplicaciones de visión por computadora en diversas

industrias, surgen nuevas bibliotecas capaces de aportar funciones adicionales para el análisis y procesamiento de imágenes en el ecosistema de Python. Entre las que destacan, Scikit-Image o SciPy. La primera, al igual que OpenCV, ofrece una amplia gama de algoritmos y herramientas para el procesado de imágenes. Además, su facilidad de uso y buena documentación, la convierte en una opción muy popular para aplicaciones más simples.

En paralelo surge NumPy, biblioteca que proporciona una infraestructura robusta para la computación numérica en Python. Esta utiliza arreglos multidimensionales (ndarrays) para representar datos, facilitando la manipulación eficiente de grandes conjuntos de datos numéricos. Estos arreglos son compatibles con los tipos de datos utilizados en las bibliotecas de procesamiento de imágenes mencionadas anteriormente, lo que favorece la interoperabilidad entre NumPy y estas librerías.

Si tenemos en cuenta que una imagen digital a color está formada por una matriz de píxeles (A x B x C), donde cada píxel es un punto único de color en la imagen. A y B representan la anchura y altura, mientras que, C es la profundidad de color o bit. Llegamos a la conclusión de que, gracias a esta interoperabilidad, los usuarios pueden trabajar con imágenes de una manera más eficiente y flexible.

Con los años, OpenCV ha lanzado nuevas versiones que han traído consigo mejoras significativas en la calidad del código, la estabilidad, el rendimiento y la funcionalidad de la herramienta. Esto ha establecido la posición de OpenCV como una de las principales bibliotecas de visión por computadora, siendo utilizada en una amplia gama de aplicaciones y proyectos de desarrollo.

Podemos encontrar OpenCV en campos tan dispares como aplicaciones de teléfonos móviles, diagnósticos médicos, coches de conducción autónoma, la industria y la robótica. Normalmente, para mejorar el rendimiento en estas aplicaciones, también se hace uso de bibliotecas de aprendizaje profundo.

Ilustración 2. Aplicación de Open CV en seguridad vial (Stallman, 2022)

El Aprendizaje Profundo o 'Deep Learning' es una rama del Aprendizaje Automático, 'Machine Learning', que se centra en el entrenamiento de modelos de inteligencia artificial para aprender y extraer representaciones complejas de datos. Entre las arquitecturas más conocidas se encuentran las redes neuronales convolucionales (CNN) capaces de realizar tareas como clasificación de imágenes, detección y seguimiento de objetos, segmentación semántica o generación de imágenes.

'TensorFlow' y 'Pytorch' son dos de las bibliotecas de 'Deep Learning' más potentes que existen en la actualidad. Ambas ofrecen herramientas avanzadas para construir y entrenar modelos de redes neuronales convolucionales, lo que permite abordar una amplia gama de problemas en el campo del procesamiento de imágenes y videos. El primero, es conocido por su flexibilidad y escalabilidad en la implementación de estos modelos. Mientras que, el segundo, destaca por su facilidad de uso y flexibilidad a la hora de definir y construir tus propios modelos.

2.2 MACHINE LEARNING Y DEEP LEARNING

La **Inteligencia Artificial** (IA) es un campo de la informática que se enfoca en crear sistemas y programas capaces de realizar tareas que imitan funciones cognitivas, es decir, aquellas que, cuando son realizadas por humanos, requieren inteligencia. Esto incluye actividades como el reconocimiento de voz, la toma de decisiones y el procesamiento de lenguaje natural, entre otras. Además, se considera IA cualquier conjunto de reglas programadas que instruyen a la máquina sobre cómo comportarse en situaciones específicas, como, por ejemplo, un par de sentencias if-else. (Oppermann, n.d.) Por lo tanto, cuando hablamos de inteligencia artificial, debemos considerar dos subcampos más específicos: el Aprendizaje Automático, o 'Machine Learning', y el Aprendizaje Profundo, también conocido como 'Deep Learning'.

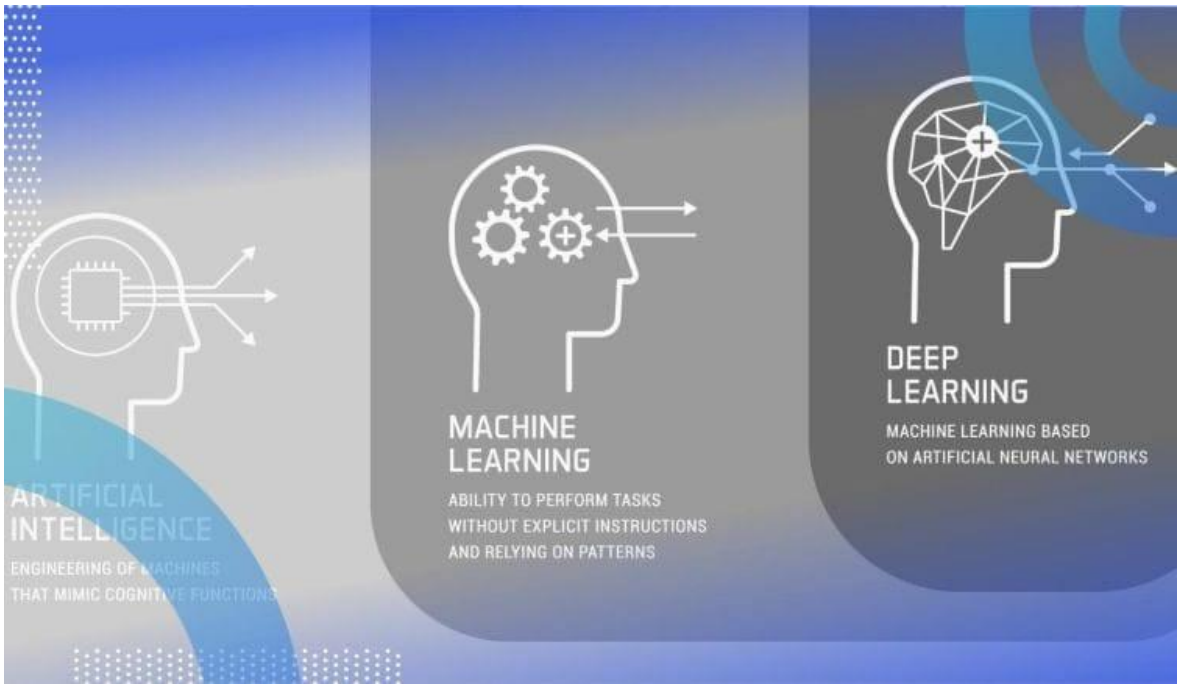


Ilustración 3. Esquema de la Inteligencia Artificial, Machine Learning y Deep Learning (Oppermann, n.d.)

Podemos pensar en el '**Machine Learning**' como una serie de algoritmos y modelos estadísticos que analizan datos, aprenden de ellos y toman decisiones basadas en esos conocimientos aprendidos. La principal diferencia entre la Inteligencia Artificial y el Aprendizaje Automático está en que no es lo mismo programar algo que aprender automáticamente a hacerlo.

No existe un algoritmo capaz de resolver con precisión cualquier tipo de problema. Por tanto, el tipo de algoritmo que elijamos dependerá del problema específico que queramos abordar, así como el número de variables y otras consideraciones. Entre estos algoritmos destacan:

- Modelos de clasificación, como el Clasificador de Naïve Bayes.
- Modelos de regresión, como las Máquinas de Soporte de Vectores (SVM).
- Técnicas de 'clusterización', como K-means.
- Árboles de decisión.

Hoy en día se les conoce como algoritmos 'planos' ya que, a diferencia de los algoritmos de aprendizaje profundo, estos no pueden aplicarse directamente a los datos en bruto, sino que requieren de un paso previo denominado 'extracción de características'.

La extracción de características suele ser bastante complicada y requiere un conocimiento detallado del problema a tratar para encontrar el algoritmo que mejor lo resuelva. Además, para alcanzar un resultado óptimo tendremos que adaptar, probar y refinar dicho algoritmo a lo largo de varias iteraciones (Oppermann, n.d.).

El '**Deep Learning**', o Aprendizaje Profundo, es una rama del aprendizaje automático que se basa en algoritmos inspirados en la estructura y funcionamiento del cerebro humano, conocidos como redes neuronales artificiales. Estas redes neuronales están compuestas por múltiples capas de nodos interconectados, lo que permite que el sistema aprenda representaciones de datos cada vez más abstractas a medida que avanza en las capas.

El término "profundo" se refiere a la profundidad de estas redes, es decir, a la presencia de múltiples capas entre la entrada y la salida de datos. Esta profundidad permite que el sistema aprenda automáticamente características relevantes de los datos de entrada, lo que lo hace

especialmente poderoso para tareas de reconocimiento de patrones en datos complejos, como imágenes, sonido o texto. En otras palabras, la extracción de características está integrada en el proceso que tiene lugar dentro de una red neuronal artificial sin intervención humana.

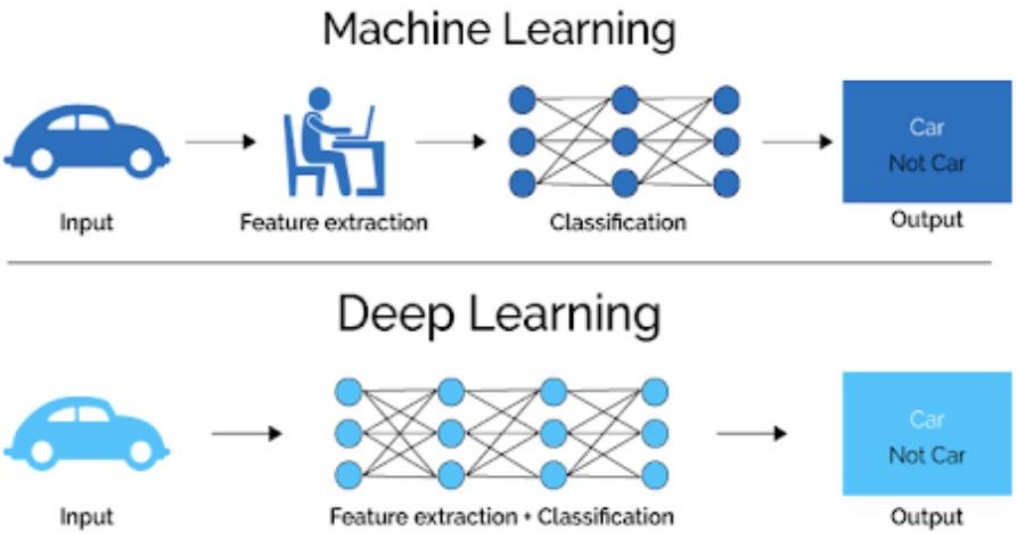


Ilustración 4. Intervención del hombre en las tareas de extracción de características (Oppermann, n.d.)

El 'Deep Learning' ha demostrado ser altamente efectivo en una variedad de aplicaciones, incluyendo el reconocimiento de imágenes y voz, la traducción automática, la conducción autónoma, el procesamiento del lenguaje natural, entre otros. Su capacidad para aprender representaciones de datos complejos de manera automática y jerárquica ha impulsado avances significativos en inteligencia artificial en los últimos años.

2.3 MÉTODOS TRADICIONALES DE EXTRACCIÓN DE CARACTERÍSTICAS

En 'Machine Learning', el paso previo al análisis de una imagen o video es su tratamiento, con el fin de modificarla para mejorar el rendimiento y la eficiencia de los modelos de aprendizaje automático. Existen varios procedimientos según el problema ante el que nos encontremos: eliminación de ruido, corrección de ojos rojos, mejora de contraste, resalte de contornos, ajuste de color, etc.

2.3.1 FILTROS LINEALES

Entre los métodos de filtrado lineal más comunes se encuentran el filtro promedio y el filtro gaussiano para el suavizado, los filtros Sobel y Prewitt para el realce de bordes y el filtro Laplaciano Gaussiano para la detección de características.

El suavizado, es un procedimiento utilizado para homogeneizar una imagen con el fin de eliminar ruido. Puede resultar muy útil tanto en el preprocesamiento de imágenes, para mejorar su calidad antes del análisis, como para reducir el ruido y la redundancia antes de comprimir una imagen. Algunas de las técnicas más utilizadas son:

El **filtro promedio**, 'median filter', es una técnica de filtrado digital lineal que consiste en tomar el promedio de los píxeles en una vecindad definida alrededor de cada píxel en la imagen.

1. Se selecciona una ventana, también llamada máscara o 'kernel', de tamaño definido V(i, j), que se desplazará sobre cada píxel de la imagen.
2. Para cada posición de la ventana, se calcula el promedio de los valores de los píxeles dentro de esa ventana. A continuación, el valor del píxel central de la ventana se reemplaza con el promedio calculado.
3. Este proceso se repite para cada píxel de la imagen, desplazando la ventana de izquierda a derecha y de arriba hacia abajo.

Se puede expresar mediante la siguiente ecuación:

$$g_{i,j} = \frac{1}{Nv} \sum_{(l,m)\in V(i,j)} f_{l,m}$$

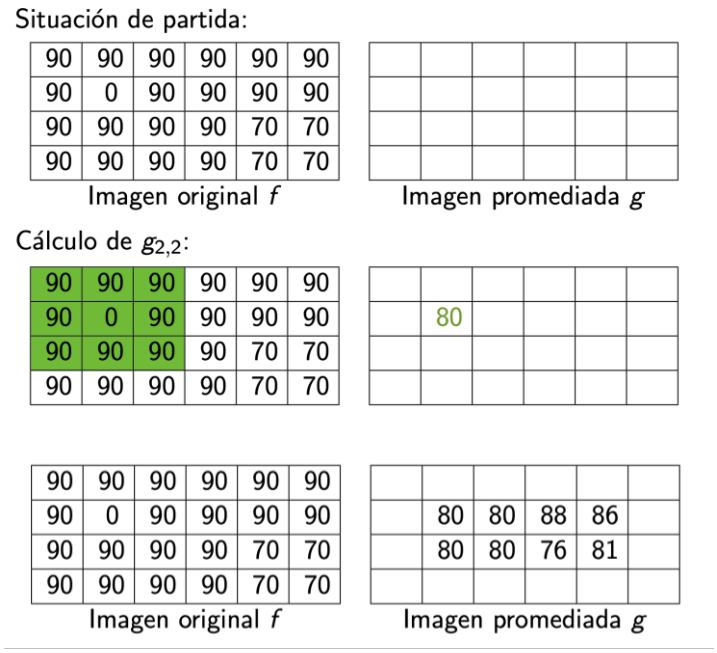


Ilustración 5. Proceso de suavizado mediante el filtro promedio (Gallego, 2018)

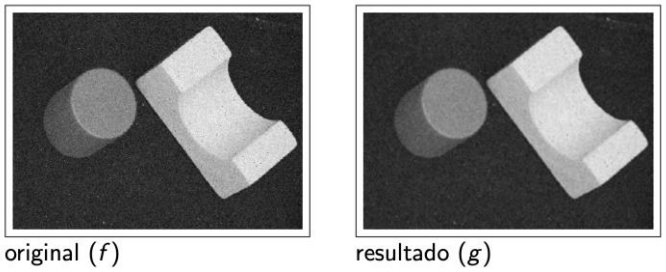


Ilustración 6. A la izqda, imagen original. A la dcha, imagen suavizada (Gallego, 2018)

El **filtro gaussiano** es un filtro lineal que se utiliza para suavizar imágenes y reducir el ruido, manteniendo la estructura general de la imagen. Funciona de la siguiente manera:

1. Se define un 'kernel' gaussiano, que es una matriz de pesos basada en la función gaussiana. Este 'kernel' tiene más peso en el centro y menos en los bordes.
2. La imagen se convoluciona con este 'kernel', es decir, cada píxel de la imagen se reemplaza por un promedio ponderado de sus vecinos, donde los pesos son dados por el 'kernel' gaussiano.
3. El proceso se realiza para cada píxel de la imagen.

Se puede expresar mediante la siguiente fórmula:

$$g_{i,j} = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \sum_{(l,m)\in V(i,j)} f_{l,m}$$

donde σ es la desviación estándar que controla el grado de suavizado.

La vecindad puede tener varios tamaños o formas según su uso. A mayor tamaño de la vecindad, más emborronamiento de la imagen, pero menos ruido sobrevivirá.

Ilustración 7. Ejemplos de vecindad (Gallego, 2018)

Estos filtros son útiles para eliminar ruido aleatorio en una imagen, sin embargo, no es adecuado para imágenes que requieren preservación de bordes y detalles finos. En estos casos, utilizaremos filtros no lineales como el **filtro de mediana**, capaz de preservar mejor los bordes mientras elimina el ruido.

2.3.2 MÉTODOS DE SEGMENTACIÓN

Los métodos de segmentación tienen como objetivo la división de una imagen por regiones con atributos similares (niveles de gris, textura, etc.) para diferenciar objetos. La separación es un paso previo para obtener información sobre la localización de objetos en una imagen.

Gracias a esto, se pueden realizar tareas como el tratamiento de zonas localizadas (ej. caras), cálculo de áreas o distancias, seguimiento de objetos o interpretación de una escena.

Los algoritmos de segmentación se basan en propiedades básicas de los valores del nivel de gris (Vega, 2017):

- Discontinuidad: Si los bordes de las regiones son lo suficiente diferentes del fondo, el algoritmo se centrará en buscar cambios bruscos de nivel de intensidad. Este método es conocido como 'Segmentación mediante contornos' y utiliza la derivada espacial o la transformada de Hough.
- Similitud: Este método divide la imagen basándose en la búsqueda de zonas que tengan valores similares. Un método muy utilizado es la 'Segmentación por umbralización', que aprovecha la diferencia de intensidad entre un objeto y el fondo

Aunque estos métodos nos llevan a aproximaciones bastante acertadas, no son perfectos. Por ello, existen operaciones adicionales que, combinadas entre sí, pueden ayudar a eliminar motas del fondo que no correspondan a objetos o rellenar fisuras dentro de los mismos. Estas operaciones forman parte de los métodos morfológicos

2.3.2.1 Segmentación mediante contornos

La segmentación mediante contornos es una técnica que identifica la frontera o líneas que delimitan regiones diferentes. En el contorno pueden observarse cambios de textura o nivel de gris, lo cual es útil para su identificación.

Existen diferentes métodos para calcular los contornos en una imagen, cada uno con sus características y aplicaciones específicas. A continuación, se describen los métodos más comunes:

1. La **derivada espacial** mide el cambio en la intensidad de los píxeles percibido al recorrer la imagen en una dirección, esta puede ser horizontal o vertical. Por ejemplo, si en una fila de píxeles se observa un cambio de intensidad de 20 a 90, este salto sugiere la presencia de un contorno.

2. El **gradiente** de una imagen en un punto es un vector que indica la dirección, sentido y magnitud del cambio de intensidad más fuerte en ese punto. Este se calcula tomando las derivadas parciales en las direcciones x e y. Por definición, $\overrightarrow{\nabla f} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$, pero estas derivadas parciales carecen de sentido en un conjunto discreto de puntos $M\ x\ N$. Por tanto, el gradiente se aproximará mediante $G = \begin{pmatrix} \frac{\Delta f}{\Delta x} \\ \frac{\Delta f}{\Delta y} \end{pmatrix}$.

   Para una imagen *f* de dimensiones *M x N*, el gradiente se calculará según:

$$G_{i,j} = \begin{pmatrix} f_{i,j} - f_{i,j-1} \\ f_{i,j} - f_{i-1,j} \end{pmatrix}$$

3. También se pueden usar plantillas (o **kernels**) para calcular las componentes del gradiente mediante convoluciones.

   Para una imagen *f* y plantillas P_x, P_y, el gradiente será igual a:

$$G_x = f \cdot P_x$$
$$G_y = f \cdot P_y$$

Dir.	Roberts	Prewitt	Sobel
Y	$\begin{matrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$	$\begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$	$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$
X	$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{matrix}$	$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$	$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$

Ilustración 8. Plantillas comunes para el cálculo del gradiente (Gallego, 2018)

Además, para contornos que son líneas rectas destaca la transformada de Hough, y para detectar contornos donde el gradiente es máximo se utiliza la derivada segunda o Laplaciana.

A continuación, se adjunta una imagen en la que se ha llevado a cabo la segmentación mediante contornos.

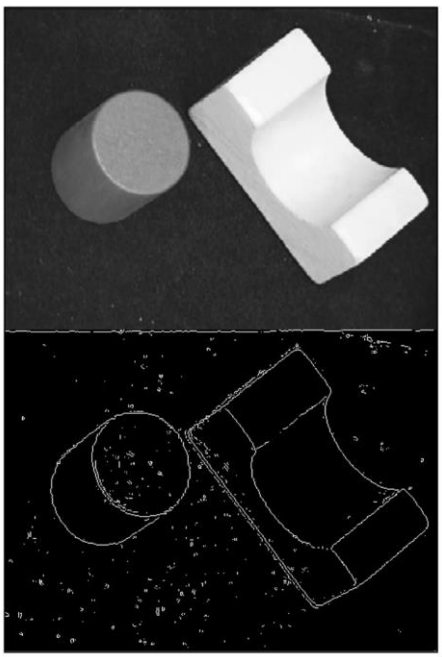


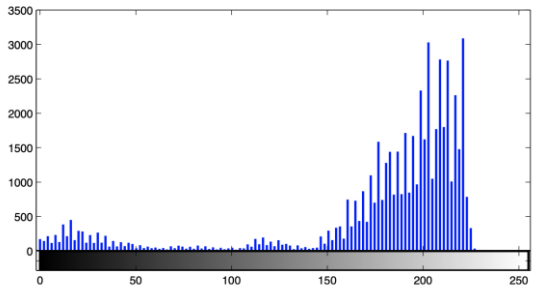
Ilustración 9. Segmentación de una imagen mediante la técnica del gradiente (Gallego, 2018)

2.3.2.2 Umbralización (histogramas de color)

La separación por histograma aprovecha la diferencia de intensidad de un objeto frente al fondo para realizar la segmentación. El histograma es una representación de la distribución de intensidades en una imagen. Se trata de convertir una imagen a escala de grises y generar una gráfica donde se muestran el número de píxeles por cada nivel de gris que aparece en la imagen.

Cuando nos encontramos ante una imagen con un objeto claro sobre fondo oscuro, o viceversa, bastará con asignar un umbral, U, dentro de los niveles de gris (0 - 255) de tal forma que aquellos píxeles de intensidad menor al umbral se convertirán en negros (píxel a

1) y los mayores en blanco (píxel a 0). Consiguiendo así una imagen binaria, como la que se muestra en la siguiente figura.



$$g(x,y) = \begin{cases} 0 \; si \; f(x,y) > U \\ 1 \; si \; f(x,y) \leq U \end{cases}$$

Ilustración 10. Histograma de intensidad de una imagen (Gallego, 2018)

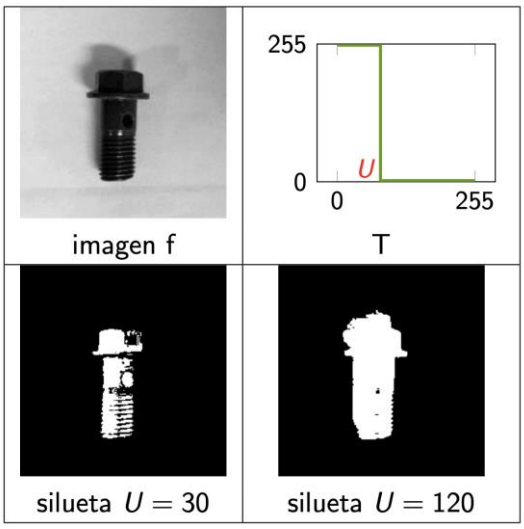


Ilustración 11. Imagen antes y después de aplicarle el método de umbralización para distintos umbrales (Gallego, 2018)

Sin embargo, es más común encontrarnos con imágenes bimodales, es decir, imágenes cuyo histograma muestras dos picos principales en los niveles de gris. Estos picos representan diferentes regiones dentro de la imagen, específicamente el fondo y múltiples objetos. En este caso, se suele utilizar el método Otsu que busca encontrar un umbral de intensidad que divida los píxeles en dos clases (fondo y objeto) de manera que la dispersión dentro de cada

clase sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre clases diferentes (Vega, 2017) .

Este método también puede usarse en imágenes RGB. Para ello, simplemente tendremos que separar la imagen en tres imágenes monocromas, una por cada canal, y aplicar el método.

2.3.2.3 *Operaciones morfológicas*

Los métodos morfológicos utilizan operaciones no lineales con el objetivo de analizar y procesar las estructuras de los objetos dentro de una imagen. En nuestro caso, nos interesan para mejorar la separación por regiones. En ocasiones, tras realizar la segmentación, ya sea mediante contornos o umbral, nos encontramos con regiones incompletas o motas en el fondo que no corresponder a un objeto. Las operaciones morfológicas básicas incluyen la erosión, dilatación, apertura y cierre.

Para llevar a cabo estas operaciones, partimos de la imagen $f_{i,j}$ y una plantilla p, matriz $p_{r,s} \in \mathbb{B}$, utilizada para examinar cada píxel de la imagen mediante una convolución. La imagen se define mediante la siguiente ecuación:

$$g_{i,j} = \mathrm{O}\left(f_{i+r-q,j+s-q}\right) \forall (r,s)/p_{r,s} > 0$$

Donde O es una operación de grupo que se aplica solo a los $f_{u,v}$ que corresponden a elementos de p no nulos.

Existen dos operaciones esenciales:

La **erosión** es una operación que reduce el tamaño de los objetos en una imagen eliminando los píxeles de los bordes. Para ello, se utiliza el operador morfológico mínimo, $\mathrm{O} = \mathrm{mín}$

$$g = f \ominus p$$

La **dilatación,** o acreción, aumenta el tamaño de los objetos en una imagen añadiendo píxeles a los bordes. Para ello, se utiliza el operador morfológico máximo, $\mathrm{O} = \mathrm{máx}$

$$g = f \oplus p$$

Si combinamos ambas operaciones, puede dar lugar a:

La **apertura** es una combinación de erosión seguida de dilatación. Se utiliza para eliminar pequeños objetos o ruido mientras se preservan las estructuras más grandes.

$$g = f \circ p = (f \ominus p) \oplus p$$

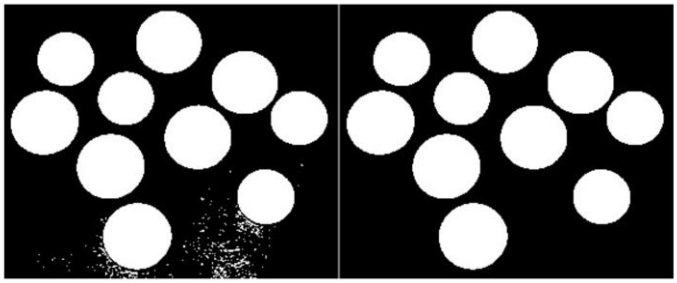


Ilustración 12. Imagen binarizada, antes (izqda) y después (dcha) de la operación de apertura (Gallego, 2018)

El **cierre** es una combinación de dilatación seguida de erosión. Se utiliza para cerrar pequeños agujeros dentro de los objetos y conectar regiones adyacentes.

$$g = f \bullet p = (f \oplus p) \ominus p$$

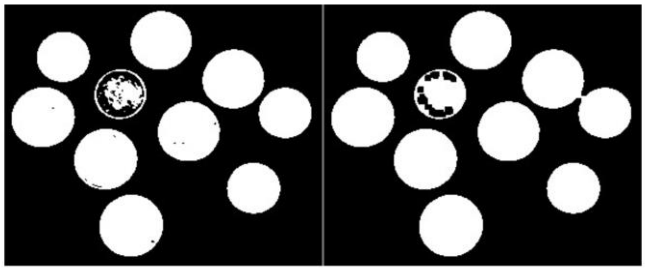


Ilustración 13. Imagen binarizada. Antes (izqda) y después (dcha) de la operación de cierre (Gallego, 2018)

2.3.3 COMPARACIÓN CON MÉTODOS ACTUALES

Estos métodos tradicionales han sentado las bases para el procesamiento de imágenes, pero con el avance de la tecnología, han surgido enfoques más avanzados, como el aprendizaje profundo, que han demostrado ser más eficaces en muchas tareas de procesamiento de imágenes.

Ventajas de los métodos tradicionales frente al aprendizaje profundo:

- Simplicidad y comprensibilidad.
- Menor necesidad de recursos computacionales.
- Rapidez en el entrenamiento.

Desventajas de los métodos tradicionales frente al aprendizaje profundo:

- Dificultad para capturar relaciones complejas y no lineales en los datos.
- Incapacidad de automatización a la hora de extraer las características (requieren un preprocesamiento manual).

2.4 REDES NEURONALES ARTIFICIALES - ANN

Una Red Neuronal Artificial, en inglés: 'Artificial Neuronal Network', es un modelo computacional inspirado en la estructura y funcionamiento del cerebro humano. Estas utilizan los datos proporcionados a la entrada para aprender a reconocer patrones que les ayuden a resolver problemas de clasificación, regresión y agrupamiento.

En 1943, el neurofisiólogo Warren McCulloch y el matemático Walter Pitts presentaron por primera vez un modelo computacional simplificado que mostraba como se podía imitar el funcionamiento conjunto de las neuronas biológicas en cerebros de animales para llevar a cabo cálculos complejos utilizando una lógica proposicional (Gerón, 2017).

La **neurona** es la unidad básica de procesamiento que podemos encontrar dentro de una red neuronal. De manera análoga a una neurona biológica, tendrá conexiones de entrada, a través de las cuales recibirá estímulos externos, y de salida.

2.4.1 PERCEPTRÓN

En 1957, Frank Rosenblatt comienza a hablar del perceptrón o **TLU** (threshold logic unit), para referirse a una neurona artificial algo diferente. En este caso, tanto los valores de entrada como los de salida son números, en vez de valores binarios (0,1). Además, cada entrada tendrá asociada un peso que determinará la intensidad con la que cada variable afecta a la neurona (Gerón, 2017).

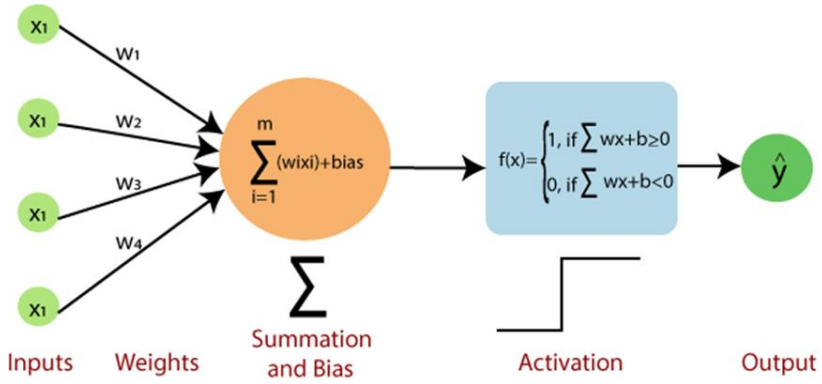


Ilustración 14. Estructura de un perceptrón (Kılıç, 2023)

Primero, la neurona resolverá un problema de regresión lineal utilizando las variables de entrada, los pesos y un término independiente, al que se denomina sesgo o 'bias'.

$$\boldsymbol{z} = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + b = \boldsymbol{w^T} \cdot \boldsymbol{x} + b$$

Las variables de entrada, $\boldsymbol{x}$, definen una recta o hiperplano cuya inclinación y posición se modificarán mediante los parámetros. Estos parámetros incluyen los pesos, $\boldsymbol{w}$, que determinan la inclinación, y el término independiente o 'bias', $\boldsymbol{b}$, que proporciona información sobre la posición vertical de la recta en el espacio.

A continuación, se añade una función de activación al resultado de la regresión lineal.

$$h_{w,b}(\boldsymbol{x}) = step(\boldsymbol{z}) = step(\boldsymbol{w^T} \cdot \boldsymbol{x} + b)$$

La función de activación determina si una neurona debe ser "activada" o no en función de su valor tras realizar la operación de regresión lineal. Generalmente, puede ser de tipo escalón, lineal, sigmoidal o gaussiana. Cada una tendrá sus ventajas y desventajas (Akif Cifci, 2023):

- La función **sigmoide** es útil para problemas de clasificación binaria, pero puede llevar a una convergencia lenta durante el entrenamiento debido a sus propiedades de saturación.
- La función **tanh**, a pesar de ser similar a la sigmoide, es menos utilizada que esta.
- La función **ReLU** es rápida de computar y lleva a una convergencia más rápida durante el entrenamiento, pero puede sufrir del problema de las "neuronas muertas" (esto ocurre cuando la salida de una neurona es siempre 0 y sus pesos nunca se actualizan durante el entrenamiento).
- La función **leaky ReLU** es un buen compromiso entre la función ReLU y la función sigmoide, pero introduce un hiperparámetro adicional (α) que debe ser ajustado.

La figura que se muestra a continuación representa las principales funciones de activación con casos de uso y sus gráficas correspondientes.

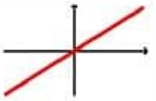
Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -½ \\ z + ½ & -½ \leq z \leq ½ \\ 1 & z \geq ½ \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

Ilustración 15. Tabla que ilustra las principales funciones de activación (Banoula, 2023)

El resultado de esta operación que relaciona la función de activación con la regresión lineal será el valor de salida del perceptrón.

2.4.2 CAPAS

El perceptrón por sí solo posee baja capacidad de procesamiento y su nivel de aplicabilidad es bajo. Sin embargo, al interconectar las neuronas entre sí somos capaces de potenciar esta capacidad de procesamiento y hacer uso del verdadero potencial de estas.

Tal como se mencionó anteriormente, una red neuronal artificial es un conjunto de algoritmos matemáticos que procesan información con el fin de encontrar relaciones no lineales entre un conjunto de datos. Estas operaciones se llevan a cabo dentro de las neuronas artificiales.

La distribución de neuronas dentro de la ANN se lleva a cabo de una manera jerarquizada. La información proveniente de fuentes externas a la red entra por la **capa de entrada**. Esta está compuesta por varias neuronas que recibirán las mismas variables de entrada y pasarán los resultados obtenidos tras procesar dicha información a la capa siguiente. Las **capas ocultas** son internas a la red y tanto la interconexión entre sus neuronas como el número de capas determinarán la tipología de la red. Por último, tenemos la **capa de salida** la cual transfiere la información que la red ha procesado hacia el exterior (Sánchez Anzola, 2015).

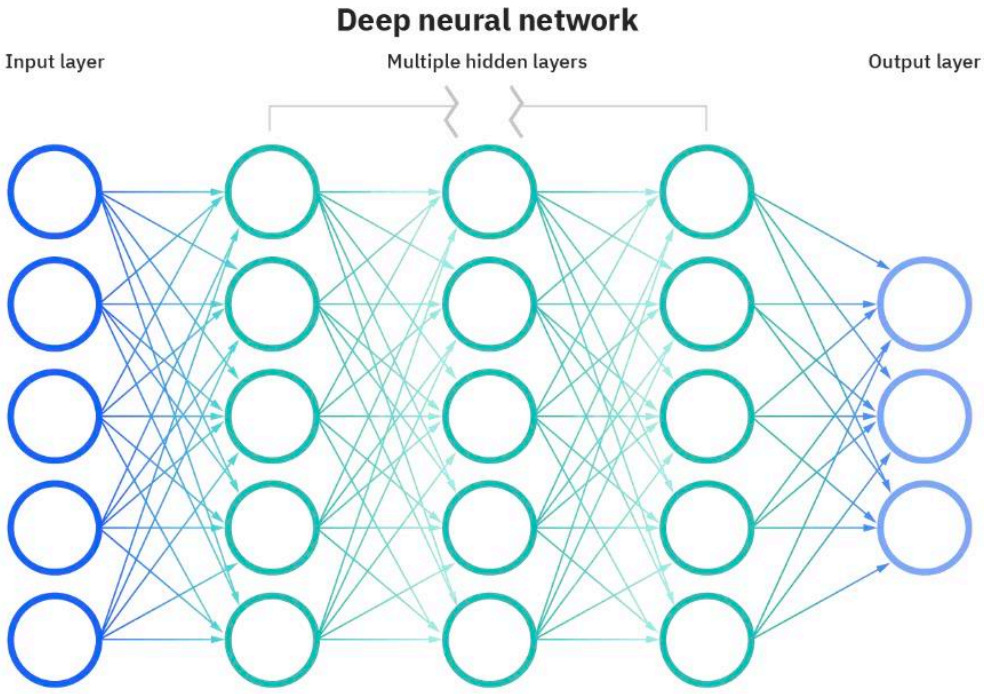


Ilustración 16. Estructura de una Red Neuronal Profunda (IBM, 2024)

2.4.3 PROCESO DE APRENDIZAJE MEDIANTE BACKPROPAGATION

El algoritmo de 'Backpropagation' se utiliza para entrenar redes neuronales y funciona en dos fases: una hacia adelante ('forward propagation') y una hacia atrás ('backward propagation').

Fase hacia adelante

Al iniciar la fase de entrenamiento, se realiza la propagación hacia adelante o 'forward propagation', en la que alimentamos la red con los datos de entrada, los cuales serán procesados por las distintas capas a medida que avanzan por la red. La capa de salida nos aportará una primera predicción del resultado.

Fase hacia atrás

A continuación, el algoritmo mide el error de salida de la red utilizando una función de pérdida que compara la salida deseada con la salida real de la red y devuelve una medida del error. El siguiente paso consiste en calcular cuánto contribuyó cada neurona de la capa anterior al error final. Este proceso se repetirá hacia atrás capa por capa hasta que todas las neuronas hayan recibido un error que describa su aportación relativa al error total.

Antes de comenzar la siguiente iteración, ajustaremos los pesos y sesgos de cada neurona en base al error que acabamos de calcular, para que la próxima vez que entrenemos la red el resultado esté más próximo al deseado (Sánchez Anzola, 2015).

Este método asegura que la red neuronal se ajuste adecuadamente a los datos de entrenamiento, reduciendo el error de predicción en cada iteración hasta que se alcance un nivel de error aceptable o se agote el número de iteraciones.

Consideraciones adicionales

- Es crucial inicializar aleatoriamente los pesos de conexión de todas las capas ocultas; de lo contrario, el entrenamiento fallará.

- En el caso de que todos los pesos y sesgos se inicializasen en cero, todas las neuronas en una capa dada serán idénticas y la backpropagation afectará a todas de la misma manera, lo que no permitirá una diversificación adecuada de las neuronas.

2.5 REDES NEURONALES CONVOLUCIONALES – CNN

La **red neuronal convolucional** o CNN, es una clase de red neuronal artificial especialmente eficaz a la hora de procesar datos de entrada bidimensionales, como las imágenes. Estas entran en la red en forma de vector de dimensiones: altura, anchura y número de canales de color (Una imagen a color consta de 3 canales: rojo, verde y azul).

La red multicapa intercala capas convolucionales, de agrupamiento ('pooling layers') y totalmente conectadas ('fully-connected layers') para transformar los datos de entrada (imágenes) en un valor de salida (como puede ser la clase de objeto detectado).

2.5.1 CAPA DE CONVOLUCIÓN

Las capas convolucionales toman un papel fundamental en el funcionamiento de las CNN's. El foco principal está en el uso de núcleos (o 'kernels') capaces de aprender y extraer características importantes de la imagen de entrada, *f*, como bordes, texturas y patrones. A continuación, se describe paso a paso la operación de convolución:

Primero, elegimos un filtro o 'kernel', matriz $p(n\ x\ n)$ que utilizaremos para modificar la imagen de entrada. Por ejemplo,

$$p = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

Dados *f* y *p* obtenemos los elementos de la matriz resultante *g,* mediante la siguiente ecuación:

$$g_{i,j} = \sum_{r=0}^{n} \sum_{s=0}^{n} p_{r,s} \cdot f_{i+r,j+s}$$

Se trata de multiplicar la plantilla o 'kernel' por la vecindad *Vn*, submatriz de tamaño *nxn* que se encuentra en la parte superior izquierda de la matriz *f*. Se realiza una multiplicación elemento a elemento entre 'kernel' y vecindad, sumando luego todos los resultados. El valor resultante se coloca en la posición correspondiente del mapa de características, *g*.

Por último, recorreremos la imagen repitiendo esta operación – de izquierda a derecha y de arriba hacia abajo – hasta generar una matriz de salida a la que denominaremos capa convolucionada (Gallego, 2018).

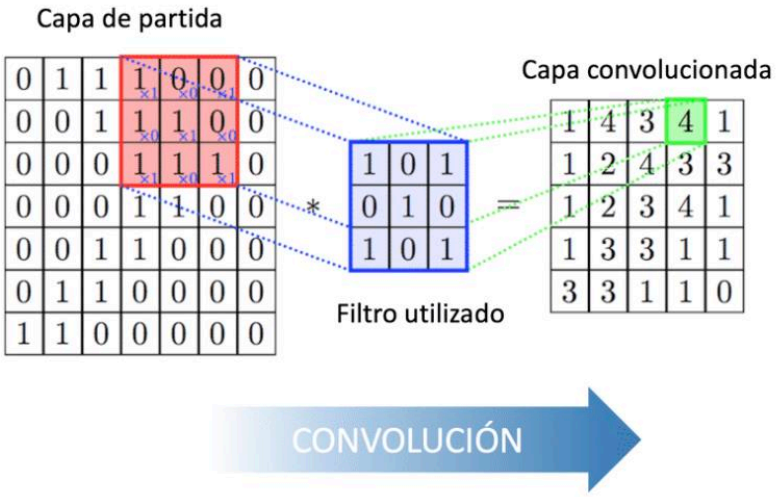


Ilustración 17. Operación de convolución (Calvo, 2017)

2.5.2 CAPA DE AGRUPAMIENTO

El objetivo de las capas de agrupamiento o 'pooling layers' es reducir gradualmente la dimensionalidad de la representación manteniendo las características más importantes detectadas por el filtro, para reducir el número de parámetros y, por ende, la complejidad computacional del modelo.

La capa de agrupamiento más utilizada en las CNN's es la '**maxpooling layer**'. Esta utiliza la función MAX para escalar la dimensionalidad de los mapas de activación. Consiste en

utilizar núcleos de agrupamiento de dimensión 2 × 2 aplicados con un intervalo de 2 a lo largo de las dimensiones espaciales de la entrada. De este modo, el mapa de activación se reduce al 25% del tamaño original, mientras que el volumen de profundidad mantiene su tamaño estándar (O'Shea & Nash, 2015).

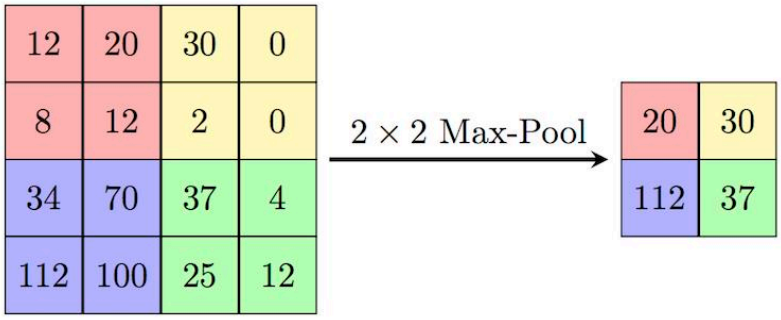


Ilustración 18. Capa ´max-pooling' con filtro 2x2 e intervalo = 2 (computersciencewiki, 2018)

2.5.3 FULLY-CONNECTED LAYER

Las capas completamente conectadas se encuentran en las últimas capas de la red, después de las capas convolucionales y de pooling. Su función principal es combinar las características extraídas por las capas anteriores para llevar a cabo en problema de clasificación o regresión (O'Shea & Nash, 2015).

En una tarea de clasificación, como es nuestro caso, la capa completamente conectada suele tener una neurona por cada clase posible. La salida de esta capa se pasa a través de una función softmax para convertirla en probabilidades de clase.

$$p(y = k|x) = \frac{e^{z_k}}{\sum_j e^{z_j}},$$

donde $p(y = k|x)$ es la probabilidad de que la entrada x pertenezca a la clase k, y z_k es la entrada ponderada a la neurona correspondiente a la clase k.

Por otro lado, en tareas de regresión, la capa completamente conectada puede tener una o más neuronas dependiendo del número de valores de salida que se necesitan predecir.

2.5.4 ESTRUCTURA DE LA CNN

La estructura típica de las capas dentro de una red neuronal convolucional es la siguiente:

1. La **capa de entrada** define el tamaño de las imágenes que la red procesará y se encarga de su normalización. Por defecto, esta capa normaliza las imágenes restando la media de las imágenes del conjunto de entrenamiento, centrando así los valores de los píxeles alrededor de cero.
2. Las **capas convoluciones**: son el núcleo de las CNN y realizan la operación de convolución sobre la imagen de entrada utilizando filtros (‘kernels’) para extraer características locales como bordes, texturas y patrones.
3. Las **capas de activación** aplican una función de activación a la salida de las capas convolucionales para introducir no linealidad en el modelo, permitiendo que la red aprenda representaciones más complejas. Entre las funciones más comunes se encuentran la ReLU, Sigmoide y tangente, de las que hemos hablado previamente.
4. Las **capas de agrupamiento** reducen la dimensionalidad de los mapas de características, disminuyendo la carga computacional y reduciendo el riesgo de sobreajuste. Entre los más utilizadas se encuentran las ‘max-pooling’ y ‘average pooling’.
5. Las **capas completamente conectadas** se encuentran al final de la red y transforman las características extraídas en la forma adecuada para la tarea de clasificación o regresión.
6. La **capa de salida** genera la predicción final de la red. En la tarea de clasificación, se utiliza la función ‘softmax’ para producir probabilidades de clase.

La razón por la que utilizamos múltiples capas de convolución es que la primera capa puede detectar características primitivas como líneas y curvas. A medida que se añaden más capas convolucionales, los mapas de características pueden reconocer formas más complejas.

Por otro lado, intercalar capas de convolución y de agrupación es esencial para controlar la carga computacional. Sin esta intercalación, la carga computacional crecería tanto que no sería posible resolver el problema de manera eficiente.

La ilustración que se muestra a continuación refleja la arquitectura típica de una red neuronal convolucional.

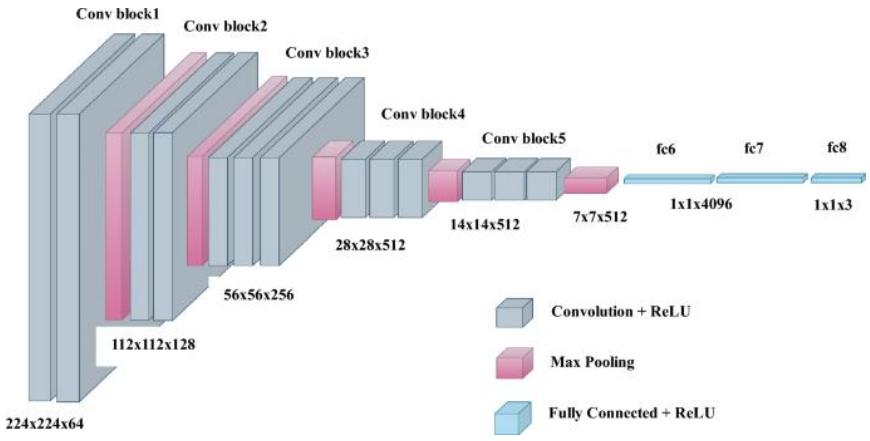


Ilustración 19. Estructura de una CNN (Dumakude & Ezugwu, 2023)

2.5.5 APRENDIZAJE SUPERVISADO

El **aprendizaje supervisado** es una técnica del aprendizaje automático (machine learning) donde un modelo es entrenado a partir de datos que previamente han sido etiquetados. Por ello, se dice que este método requiere de asistencia externa.

En el contexto de las CNN, las imágenes proporcionadas a la red tendrán asociadas una etiqueta que indicará la categoría a la que pertenecen. Por ejemplo, 'perro', 'coche' o 'persona'. Dividiremos el set de imágenes en tres bases de datos: entrenamiento, validación y prueba. Normalmente, en porcentajes de 75, 15 y 10 respectivamente.

Durante la **etapa de entrenamiento**, se pasa un set de imágenes que usaremos para calibrar los pesos de la red. El modelo aprenderá patrones gracias a la fase de entrenamiento, que

luego aplicará durante las fases de validación y prueba para realizar las predicciones y clasificaciones pertinentes.

La **etapa de validación** nos permitirá evaluar el rendimiento de nuestro modelo al pasarle imágenes a las que no tuvo acceso durante el entrenamiento.

La **etapa de prueba** se llevará a cabo una vez terminados los procesos de entrenamiento y validación, con el fin de evaluar el rendimiento del modelo obtenido (Mahesh, 2019).

El error en un modelo de machine learning viene dado por tres parámetros:

$$Error\ total = sesgo + varianza + tasa\ de\ error\ óptima$$

Mientras que hay una parte de este error que nunca se podrá reducir por completo, los errores debidos al sesgo y a la varianza se pueden evitar.

Si evaluamos simultáneamente el error de entrenamiento y validación podremos percibir el momento en el que el modelo entra en la etapa de sobreajuste o 'high variance'. Esta etapa se caracteriza por que el error de entrenamiento sigue disminuyendo, mientras que el de validación comienza a aumentar. Lo que ocurre es que el modelo es tan complejo que se ajusta demasiado a los datos, hasta el punto de dejar de ser un modelo generalizado. Por otro lado, si tanto el error de entrenamiento como el de validación son altos, el modelo estará subajustado. En otras palabras, carece de la información necesaria para resolver correctamente el problema o requiere un proceso de entrenamiento más prolongado (Mallick, 2017).

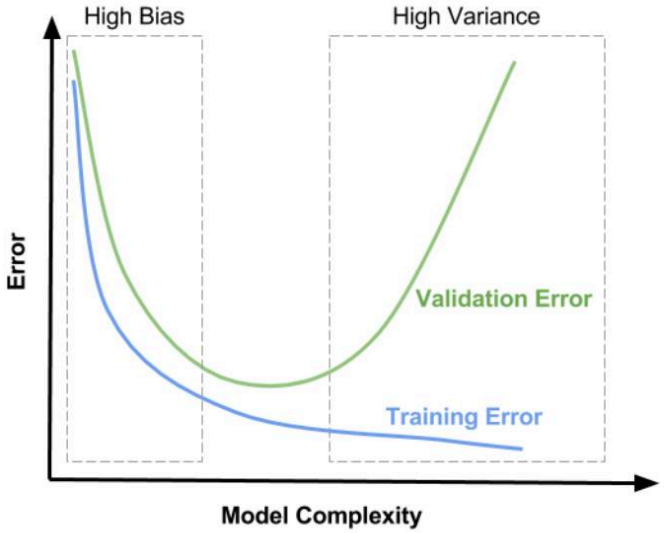


Ilustración 20. Gráfica Error / Complejidad del modelo (Mallick, 2017)

Capítulo 3. OPEN CV

En este capítulo se aborda la utilización de OpenCV, una biblioteca de software de código abierto, para el procesamiento de imágenes y visión por computadora. Se describen las funciones esenciales de OpenCV para la captura y análisis de imágenes y videos, así como su aplicación en la detección y seguimiento de objetos. Además, se detallan las implementaciones prácticas de estos métodos en entornos no controlados y se discuten las limitaciones y desafíos encontrados.

3.1 INTRODUCCIÓN A LA LIBRERÍA OPEN CV

Open CV ('Open Source Computer Vision Library') es una biblioteca de software de código abierto especializada en procesamiento de imágenes y visión por computadora. Esta fue desarrollada originalmente por Intel en junio del 2000 y ahora es mantenida por una comunidad de desarrolladores. Open CV proporciona una amplia gama de funciones para la captura, manipulación y análisis de imágenes y videos. Sus herramientas permiten realizar tareas como la detección de bordes, el reconocimiento facial, el seguimiento de objetos, y el procesamiento de video en tiempo real.

Open CV es compatible con sistemas operativos como Windows, Linux, macOS, Android e iOS. Se puede utilizar con diversos lenguajes de programación, incluyendo C++, Python, Java y MATLAB. Además, está optimizada para maximizar el rendimiento del hardware, tanto CPU como GPU.

La comunidad de usuarios y desarrolladores de Open CV es extensa, lo que facilita el acceso a recursos, tutoriales y ejemplos de código. Entre sus aplicaciones comunes se encuentran el reconocimiento facial, la visión para robots, el procesamiento de video, la realidad aumentada y la seguridad y vigilancia (Stallman, 2022).

La motivación del proyecto parte de la idea de desarrollar un sistema capaz de detectar y responder a posibles incidencias producidas en infraestructuras remotas a lo largo de nuestra

región. A continuación, se explicará cómo utilizar las herramientas que ofrece esta biblioteca para analizar secuencias de video, con el fin de monitorizar y detectar actividades sospechosas.

3.2 DETECCIÓN Y SEGUIMIENTO DE OBJETOS EN ENTORNOS NO CONTROLADOS

Para la creación de nuestro detector/seguidor de objetos, hemos utilizado una grabación llevada a cabo por una cámara de seguridad situada en el extremo de una infraestructura privada. Esta cámara graba a unos individuos que intentan acceder al recinto sin autorización en plena noche.

3.2.1 SEGUIDOR DE OBJETOS

Para implementar nuestro seguidor de objetos en Python utilizamos una clase a la que llamaremos 'Rastreador', cuyo objetivo es identificar y rastrear objetos en un espacio bidimensional, asignándoles identificadores únicos (IDs). Al inicializarse, la clase define dos variables: 'self.centro_puntos', un diccionario que almacena las posiciones centrales de los objetos detectados, y 'self.id_count', un contador para asignar IDs únicos.

El método principal, 'rastreo', recibe una lista de objetos representados como rectángulos definidos por sus coordenadas y dimensiones. Para cada objeto, se calcula su punto central. Luego, se verifica si el objeto ya ha sido detectado anteriormente, comparando su punto central con los almacenados en 'self.centro_puntos'. Esta comparación se realiza mediante el cálculo de la distancia euclidiana. Si la distancia es menor que el umbral, se asume que es el mismo objeto, se actualiza su posición y se le asigna su ID previo.

En cambio, si el objeto no ha sido detectado previamente, se le asigna un nuevo ID, se almacena su punto central en 'self.centro_puntos', y se incrementa 'self.id_count'. Después de procesar todos los objetos, se limpia el diccionario de puntos centrales para eliminar los IDs que ya no se utilizan, actualizándolo con solo los objetos presentes en la lista 'objetos_id'.

En resumen, esta función nos permite identificar y rastrear objetos a lo largo del tiempo mediante la asignación de identificadores únicos y la actualización continua de sus posiciones centrales.

3.2.2 DETECTOR DE INCIDENCIAS

Una vez creada la clase 'Rastreador', la cual se encargará del seguimiento de los objetos, utilizaremos esta clase junto con la biblioteca Open CV para implementar una función capaz de detectar objetos en movimiento que puedan causar incidencias. El objetivo final es identificar y rastrear estos objetos en movimiento a lo largo de un video, asignándoles identificadores únicos (IDs) para mantener su seguimiento a lo largo del tiempo.

Comenzamos importando la biblioteca Open CV y creando una instancia de la clase 'Rastreador'. A continuación, se abre el archivo de video que queremos procesar y se reduce su resolución para poder visualizarlo mejor. Seleccionamos una zona de interés que abarque tanto la zona privada como el límite de traspaso, para no agregar carga computacional innecesaria al procesamiento.

En un bucle continuo, se lee cada fotograma del video. Creamos una máscara donde los objetos en movimiento son representados por píxeles blancos y el fondo es negro, utilizando un sustractor de fondo que diferencia los objetos en movimiento del fondo estático. Posteriormente, se aplica un umbral binario para eliminar los píxeles grises de la máscara.

A continuación, se detectan los contornos de dichos objetos y se filtran aquellos que no superen un determinado umbral. Para cada contorno restante, se calculan las coordenadas y dimensiones de los rectángulos que los rodean. Estos rectángulos se dibujan en el fotograma original y se almacena su información en una lista de detecciones.

La lista de detecciones se pasa al método 'rastreo' de la instancia 'Rastreador', que asigna y actualiza IDs únicos para los objetos detectados. Finalmente, se muestra el video original con los rectángulos que delimitan los objetos en movimiento, permitiendo al usuario identificar si está ocurriendo algo fuera de lo habitual.

3.3 INCONVENIENTES/LIMITACIONES

Aunque este sistema es útil para aplicaciones de vigilancia, análisis de tráfico y cualquier otra situación en la que sea necesario identificar y rastrear objetos en movimiento de manera continua, existen varios inconvenientes que dificultan su aplicación a gran escala:

- Es esencial que la cámara permanezca estática en un punto fijo para asegurar la precisión del seguimiento.
- No es un proceso que se pueda automatizar por completo. Por ejemplo, es necesario seleccionar manualmente la zona de interés y ajustar los parámetros de umbralización, los cuales variarán según la resolución de la cámara, las condiciones de iluminación y la distancia a la que se encuentre de la zona de interés. Además, estos ajustes pueden requerir recalibración frecuente debido a cambios en las condiciones ambientales.
- Los fenómenos meteorológicos, como la lluvia, la nieve o la niebla, pueden afectar significativamente la visibilidad y la eficacia del sistema de seguimiento. Esto reduce la precisión y la fiabilidad de la detección de objetos en condiciones adversas.

A continuación, se muestra un ejemplo de cómo nuestro modelo se puede ver afectado por fenómenos meteorológicos como la lluvia. En este instante de la grabación la lluvia era tal que el modelo no fue capaz de filtrarla.

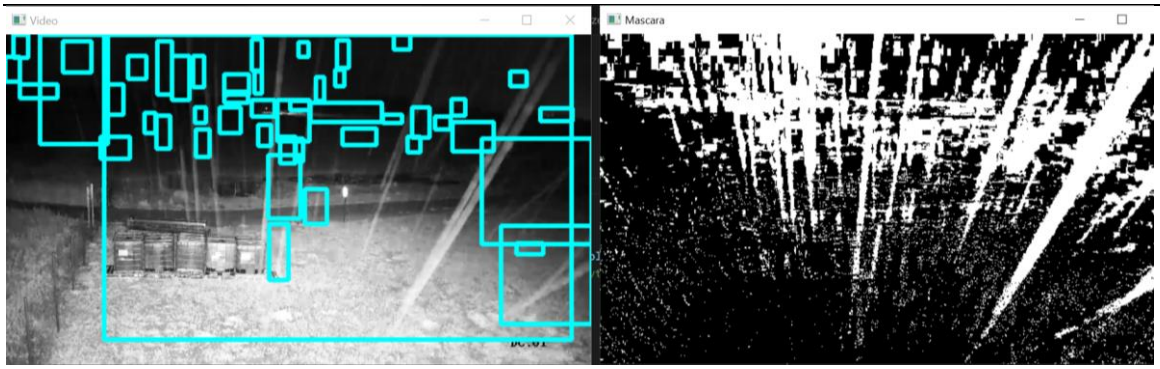


Ilustración 21. Fotograma de un video tomado por una cámara de seguridad en el que comienza a llover. A la izquierda, numerosos falsos positivos provocados por la lluvia. A la derecha, máscara que muestra el cambio en los píxeles respecto al fotograma anterior.

Otro de los problemas mencionamos previamente, es el de los reajustes necesarios ante cambios en la iluminación. En la imagen que se añade a continuación, se puede ver como los focos del coche iluminan el camino provocando falsos positivos debido a cambios en la intensidad luminosa de los píxeles.

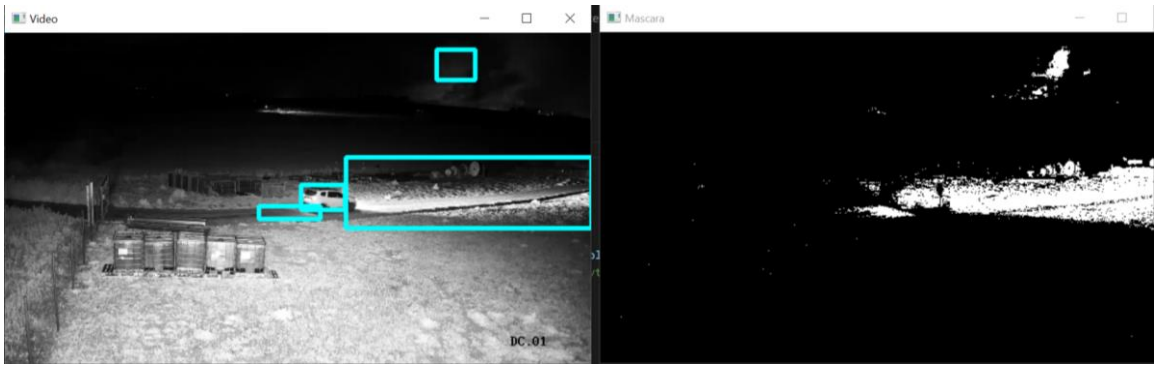


Ilustración 22. (Blanco, 2024)

3.4 CONCLUSIÓN

Aunque nuestro modelo actual proporciona una funcionalidad básica para la detección y seguimiento de objetos, presenta limitaciones significativas, especialmente bajo condiciones adversas como cambios en la iluminación o fenómenos meteorológicos. Estos problemas generan falsos positivos y reducen la precisión del sistema. Por tanto, es evidente la necesidad de explorar algoritmos más avanzados como YOLO (You Only Look Once), que ofrecen una mayor robustez y precisión en la detección de objetos, mejorando significativamente la eficiencia y fiabilidad de nuestro sistema de vigilancia.

Capítulo 4. SISTEMAS YOLO

En este capítulo, exploraremos los Sistemas YOLO (You Only Look Once) y su impacto en la detección de objetos en tiempo real. Comenzaremos introduciendo los fundamentos de YOLO, su arquitectura y cómo difiere de otros enfoques de detección de objetos. Aprenderemos sobre el proceso de predicción de YOLO y discutiremos sus ventajas y desventajas, así como las técnicas de post-procesamiento que mejoran su precisión. Finalmente, revisaremos las evoluciones más recientes de YOLO, destacando las mejoras en velocidad y precisión que han hecho de YOLO una herramienta esencial en la visión por computadora.

4.1 DETECCIÓN DE OBJETOS

La detección de objetos es una técnica en visión por computadora que permite identificar y localizar objetos dentro de una imagen o un video. A diferencia de la clasificación de imágenes, que solo asigna una etiqueta a toda la imagen, la detección de objetos no solo clasifica los objetos presentes, sino que también proporciona las coordenadas de los cuadros delimitadores que encierran dichos objetos. Se trata de una parte importante de muchas aplicaciones, como la vigilancia, los coches autónomos o la robótica.

Inicialmente, se utilizaban métodos computacionalmente costosos y lentos como las ventanas deslizantes o 'Sliding Window Object Detection'. Esto fue evolucionando a técnicas más avanzadas y eficientes como las redes R-CNN, Fast R-CNN y Faster R-CNN. Finalmente, con la aparición del algoritmo YOLO la detección de objetos en tiempo real se ha vuelto una realidad, combinando precisión y velocidad en un enfoque unificado y eficiente.

4.1.1 TWO-SHOT OBJECT DETECTION VS SINGLE-SHOT OBJECT DETECTION

La detección de objetos en dos pasadas y en una sola toma ofrece distintas estrategias para predecir la presencia y ubicación de objetos en imágenes. En la detección en dos pasadas, se

emplean dos fases de procesamiento de la imagen: en la primera fase se generan propuestas preliminares de ubicación de objetos, las cuales son refinadas y utilizadas para hacer predicciones finales en la segunda fase. Mientras que, la detección en una sola toma procesa la imagen de una vez para realizar todas las predicciones necesarias.

La detección en dos pasadas es más precisa que la detección en una sola toma, pero también es más demandante computacionalmente. En contraste, la detección en una sola toma es más eficiente en términos de recursos, aunque puede ser menos precisa, especialmente cuando se trata de objetos pequeños.

La elección entre estos métodos depende de las necesidades específicas de cada aplicación. La detección en una sola toma es ideal para aplicaciones que requieren tiempos de respuesta rápidos, como en entornos de tiempo real. Por otro lado, la detección en dos pasadas es preferible cuando la precisión es importante, sacrificando algo de eficiencia computacional por una mejor exactitud en la detección de objetos (Kundu, 2023).

Existen varias redes reconocidas para la detección de objetos utilizando dos pasadas, estas son las ‘Region-Based CNN’ entre las que se encuentran las R-CNN, Fast R-CNN y Faster R-CNN. Por otro lado, las redes más destacadas que emplean una sola pasada incluyen YOLO y SSD.

4.1.2 REGION BASED CNNS

El proceso de predicción de las CNN basadas en regiones, o 'Region Based CNN', es el siguiente. Primero, generan un conjunto de regiones candidatas dentro de la imagen o video, conocidas como "propuestas de regiones". Luego, una CNN clasifica cada una de estas regiones para determinar si contienen un objeto. Las regiones identificadas como contenedoras de objetos se procesan adicionalmente para refinar la ubicación del objeto y asignarlo a una clase específica (Mirkhan, 2023).

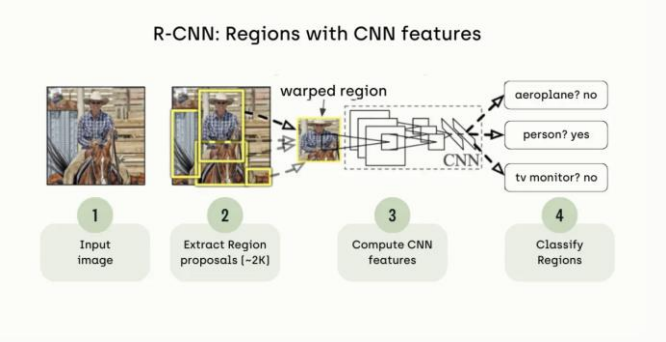


Ilustración 23. Esquema de funcionamiento de una CNN basada en regiones (Mirkhan, 2023)

Las CNN basadas en regiones pueden realizar la detección de objetos con precisión, pero tienen sus puntos débiles. Entre ellos se encuentran el alto coste computacional y la ineficiencia, lo que las hace poco prácticas para la detección en tiempo real en dispositivos de baja potencia. Además, requieren grandes conjuntos de datos etiquetados, cuya obtención es costosa y requiere mucho tiempo, limitando su adecuación para ciertas aplicaciones.

4.2 REDES YOLO

YOLO (You Only Look Once) es un algoritmo de detección de objetos que utiliza redes neuronales convolucionales para detectar objetos en tiempo real. Su capacidad de detección a alta velocidad se consigue procesando las fotos en una sola pasada y prediciendo al mismo tiempo las clases de objetos y las coordenadas de los cuadros delimitadores. Estos modelos se entrenan de principio a fin utilizando un gran conjunto de datos de imágenes etiquetadas y sus respectivos cuadros delimitadores.

La detección en una sola pasada permite a YOLO ser más rápido y eficiente que los algoritmos de dos pasadas, eliminando la necesidad de múltiples etapas y permitiendo su ejecución en tiempo real, aunque a costa de una menor precisión.

4.2.1 ARQUITECTURA Y PROCESO DE PREDICCIÓN DE YOLO

Antes de comenzar con el proceso de predicción del algoritmo **YOLO**, llevaremos a cabo la etapa de preprocesamiento en el que la imagen de entrada se redimensiona a un tamaño estándar y se normaliza para preparar los datos para la red neuronal.

A continuación, la imagen procesada pasa por una Red Neuronal Convolucional (CNN) para extraer sus características relevantes. Esta etapa convierte la imagen en un mapa de características que contiene información esencial sobre los patrones presentes en la imagen.

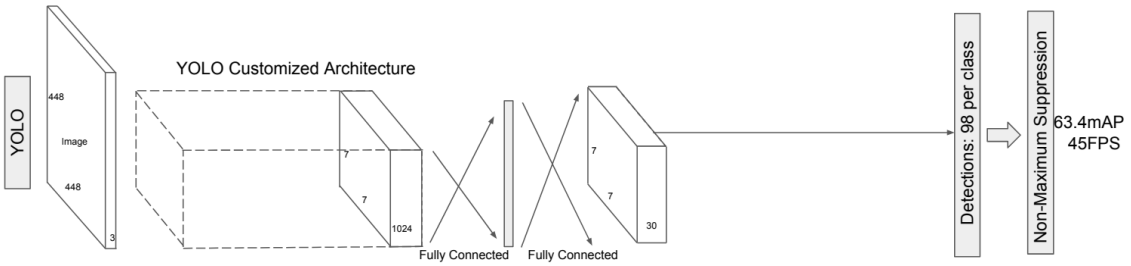


Ilustración 24. Arquitectura de la red YOLO (Liu, Anguelov, Erhan, & Szegedy, 2016)

El mapa de características se pasa a través de una serie de capas completamente conectadas que generan predicciones de coordenadas de los cuadros delimitadores (bounding boxes) y probabilidades de clase para cada celda de una cuadrícula en la imagen. Se divide la imagen de entrada en N cuadrículas de tamaño *S x S.* Cada una de estas cuadrículas se encarga de averiguar si contienen un objeto y en caso de haberlo localizar su posición (Mirkhan, 2023).

Si el centro de un objeto cae en una celda de la cuadrícula, esa celda es la responsable de detectar el objeto. Cada celda de la cuadrícula predice B recuadros delimitadores y puntuaciones de confianza para esos recuadros. Estas puntuaciones de confianza reflejan el grado de seguridad que tiene el modelo de que la caja contiene un objeto y el grado de precisión que considera que tiene la caja predicha.

Múltiples celdas que predicen el mismo objeto con distintos cuadros delimitadores pueden generar una gran cantidad de predicciones duplicadas. Para resolver este problema, se utiliza

la técnica de supresión de no máximos (Non-Maximum Suppression, NMS). Este método toma las puntuaciones de probabilidad asociadas a cada predicción y selecciona la más alta. Luego, se suprimen los cuadros delimitadores que tienen la mayor intersección sobre unión (Intersection over Union, IoU) con el cuadro delimitador actual de mayor probabilidad. Esto asegura que solo se mantengan las predicciones más precisas y reduce significativamente las redundancias y el número de falsos positivos, mejorando la calidad de la detección de objetos (Singh, 2023).

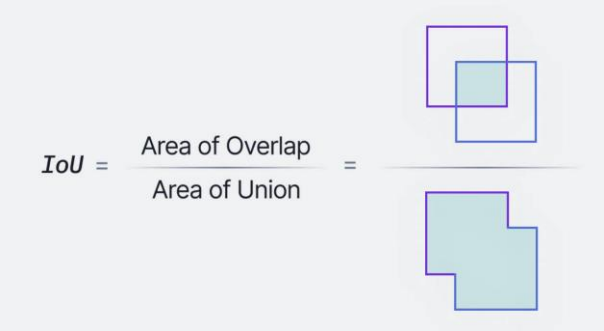


Ilustración 25. Fórmula de la intersección dividida por la unión (Kundu, 2023)

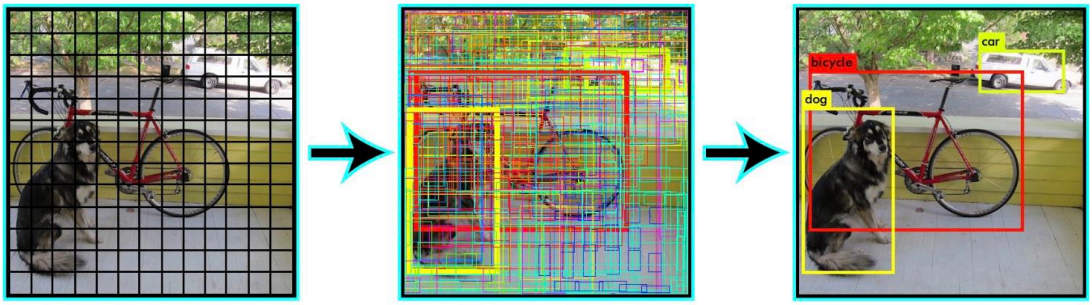


Ilustración 26. Cuadrícula a la izquierda, todas las predicciones en el centro, predicciones tras aplicar la técnica de supresión de no máximos a la derecha (Kurz, 2021)

4.2.2 MÉTRICAS DE EVALUACIÓN

4.2.2.1 Métricas de entrenamiento y validación

Utilizaremos tres formas de medir el error tras realizar los procesos de entrenamiento y validación.

- El error en la predicción de las cajas delimitadoras mide como de bien está prediciendo el modelo las posiciones y tamaños de los 'bounding boxes'.
- El error de clasificación mide cuán bien el modelo está clasificando los objetos dentro de las cajas delimitadoras. Mientras menor sea el error, mayor será la precisión en la clasificación.
- El error en la predicción de la distribución de la focalización durante el entrenamiento y la validación cuantifica la precisión en la predicción de la ubicación de los bordes de las cajas delimitadoras.

4.2.2.2 Métricas de Precisión y Evaluación

El 'Mean Average Precision' (mAP) es una métrica comúnmente utilizada para evaluar modelos de detección de objetos. Para entender cómo se calcula el mAP, primero necesitamos comprender los siguientes parámetros (Shah, 2022):

Matriz de confusión

Para crear una matriz de confusión, necesitamos considerar cuatro atributos clave:

- Verdaderos Positivos (TP): El modelo predice una etiqueta y coincide correctamente con la verdad del terreno.
- Verdaderos Negativos (TN): El modelo no predice la etiqueta y esta no forma parte de la verdad del terreno.
- Falsos Positivos (FP): El modelo predice una etiqueta que no es parte de la verdad del terreno (Error Tipo I).
- Falsos Negativos (FN): El modelo no predice una etiqueta que sí es parte de la verdad del terreno (Error Tipo II).

Estos atributos nos permiten evaluar la precisión del modelo, identificando tanto los aciertos como los errores en sus predicciones. Así, los elementos diagonales de la matriz de confusión representan clasificaciones correctas (TP y TN), mientras que los elementos no diagonales representan clasificaciones erróneas (FP y FN). Esta matriz es una herramienta esencial para analizar el rendimiento de un modelo de clasificación, proporcionando una visión detallada de dónde y cómo está fallando el modelo, lo que facilita la identificación de áreas específicas de mejora.

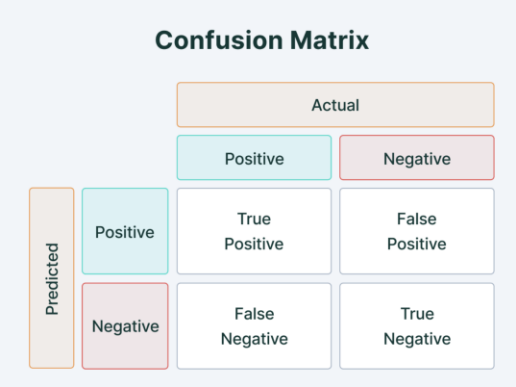


Ilustración 27. Esquema de la matriz de confusión (Shah, 2022)

Intersección sobre la Unión (IoU)

La intersección sobre la unión mide el solapamiento de las coordenadas del cuadro delimitador previsto con el cuadro real. Se calcula mediante la siguiente ecuación:

$$IoU = \frac{Área\ de\ solapamiento}{Área\ de\ unión}$$

Un IoU alto indica que las coordenadas del cuadro delimitador pronosticado se parecen mucho a las coordenadas del cuadro real.

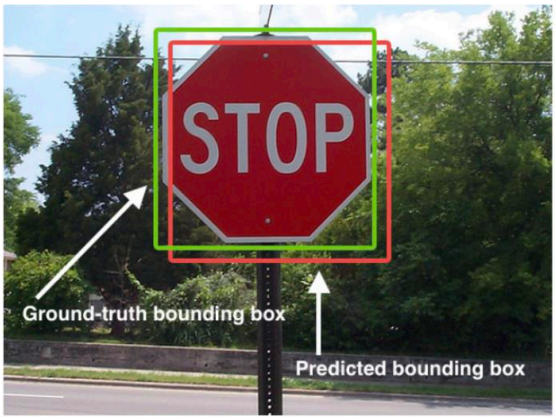


Ilustración 28. Fotografía que muestra la intersección sobre la unión del 'bounding' box real y el predicho (Shah, 2022)

Precisión

La precisión es la proporción de verdaderos positivos sobre el total de positivos predichos. En otras palabras, mide la capacidad del modelo para identificar correctamente un objeto cuando lo detecta. Es decir, la cantidad de predicciones correctas de objetos en comparación con todas las predicciones de objetos realizadas.

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

Por ejemplo, si un modelo de detección de objetos identifica correctamente 80 coches (TP) y erróneamente clasifica 20 objetos como coches cuando no lo son (FP), esto significa que el 80% de las veces que el modelo predice la presencia de un coche, lo hace correctamente.

Es importante saber que, en tareas de detección de objetos, la precisión se calcula utilizando un umbral IoU. Por ejemplo, si un objeto tiene una intersección sobre unión menor que un determinado umbral, se clasificará como falso negativo. En cambio, si la IoU es mayor que el umbral se considera verdadero positivo.

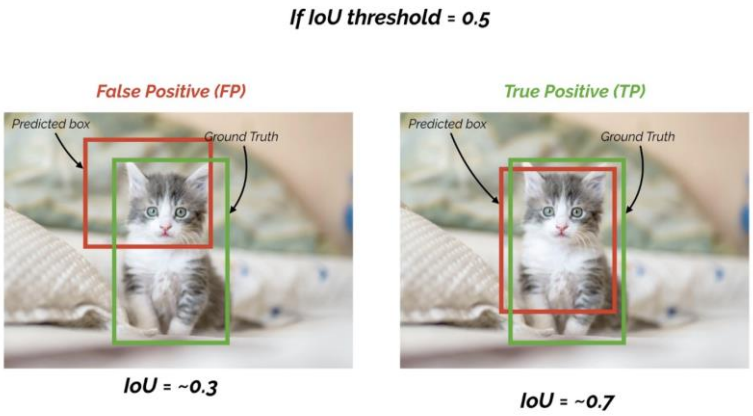


Ilustración 29. Cálculo del umbral de la IoU (Shah, 2022)

Sensibilidad

La sensibilidad o 'recall' es una métrica que evalúa la capacidad del modelo para detectar verdaderos positivos (TP) de entre todos los casos realmente positivos (TP + FN). Es decir, el 'recall' nos indica como de bien/mal el modelo encuentra los casos positivos verdaderos en comparación con el total de casos positivos que existen.

$$Sensibilidad = \frac{Verdaderos\ positivos\ (TP)}{Verdaderos\ positivos\ (TP) + Falsos\ negativos\ (FN)}$$

Esta métrica es crucial en contextos donde no detectar un caso positivo puede tener graves consecuencias, como en diagnósticos médicos o sistemas de seguridad, donde es esencial capturar la mayoría de los casos positivos para evitar fallos críticos.

Mean Average Precision (mAP)

A continuación, se resumen los pasos para calcular la Precisión Promedio (AP):

1. Generar las puntuaciones de la predicción utilizando el modelo
2. Convertir las puntuaciones de la predicción en etiquetas de clase
3. Calcular la matriz de confusión

4. Calcular las métricas de precisión y recall
5. Calcular el área bajo la curva de precisión-recall
6. Medir la precisión promedio.

El mAP se calcula encontrando la Precisión Promedio (AP) para cada clase y luego promediando estos valores entre el número de clases. Esta métrica incorpora un balance entre precisión y recall, considerando tanto los falsos positivos como los falsos negativos.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = the\ AP\ of\ class\ k,$$

$$n = the\ number\ of\ classes$$

La precisión promedio (AP) depende del umbral de IoU seleccionado. Este umbral determina lo estricta que será la evaluación de la coincidencia entre los cuadros delimitadores previstos y los reales. Para evitar la subjetividad que conlleva la elección de un umbral de IoU, se utiliza el mAP, que considera un conjunto de umbrales de IoU en el cálculo del AP. De esta manera, se calcula el AP para cada clase en diferentes umbrales de IoU y luego se promedian estos valores, proporcionando una evaluación más robusta y generalizada del rendimiento del modelo (Shah, 2022).

4.2.3 EVOLUCIÓN DEL ALGORITMO YOLO

Las diferentes versiones de YOLO han mostrado una evolución significativa en la detección de objetos en tiempo real. **YOLO v1** nació en 2016 e introdujo la detección en una sola pasada, permitiendo un procesamiento rápido al dividir la imagen en una cuadrícula y predecir simultáneamente las clases y las coordenadas de los objetos. Aunque innovador, tenía limitaciones en la precisión, especialmente con objetos pequeños y múltiples objetos cercanos.

YOLO v2 (YOLO9000, 2016) mejoró tanto la precisión como la velocidad al introducir anclas de diferentes escalas y normalización por lotes. Este enfoque permitía una mejor

adaptación a objetos de varios tamaños, aunque todavía presentaba desafíos con objetos muy pequeños y múltiples objetos muy próximos.

YOLO v3 surge en 2018. Este algoritmo usa Darknet-53 como pilar fundamental y añade pirámides de características para mejorar la detección de objetos pequeños a múltiples escalas. Estas mejoras hicieron que la red fuera más precisa y estable en comparación con sus predecesores, aunque a costa de una mayor complejidad y un incremento en el tiempo de procesamiento.

YOLO v4 (2020) implementó CSPNet y SPP, mejorando la eficiencia y la precisión. También introdujo la pérdida GHM para manejar mejor los datasets desequilibrados. Aunque estas mejoras incrementaron la precisión, también aumentaron la complejidad computacional y los requisitos de hardware necesarios para ejecutar el modelo de manera eficiente.

YOLO v5 fue desarrollado por Ultralytics en 2020. Incorporando 'EfficientDet' y técnicas avanzadas como cuadros delimitadores dinámicos y pooling piramidal espacial, consiguieron una mayor precisión y mejor generalización a diversas categorías de objetos, pero también implicó un aumento en los requisitos de hardware para entrenamiento y ejecución.

En 2022 llega **YOLO v6**, la cual utiliza EfficientNet-Lite para mejorar la eficiencia computacional y reducir el número de parámetros del modelo. Esto permitió que YOLO v6 lograra resultados de última generación al compararlos con otros algoritmos de detección de objetos, aunque con un incremento en la complejidad y los recursos necesarios para el entrenamiento.

YOLO v7 (2022) introduce nuevos tipos de cuadros limitadores y la función de pérdida focal, mejorando la detección de objetos pequeños y la precisión global. Con una alta resolución de procesamiento y una velocidad de 155 cuadros por segundo, YOLO v7 es ideal para aplicaciones en tiempo real, aunque su precisión sigue siendo menor que la de algunos detectores de dos etapas (Mirkhan, 2023).

En 2023, Ultralytics lanzó **YOLOv8**, la última versión de la serie, que presenta mejoras significativas en precisión, velocidad y versatilidad para la detección de objetos en tiempo real. YOLOv8 supera a sus predecesores al introducir varias mejoras significativas en la arquitectura del modelo, utilizando CSPDarknet53, una versión modificada del Darknet, como pilar fundamental de su arquitectura, además de un sistema de detección sin cuadros delimitadores, lo que optimiza el rendimiento y reduce la complejidad computacional. Además, esta versión es compatible con diversas plataformas de hardware, desde dispositivos edge hasta la nube, y soporta tareas avanzadas de visión por computadora, como segmentación de instancias, clasificación de imágenes y estimación de poses humanas. La integración con PyTorch y una interfaz de línea de comandos intuitiva facilitan su uso y experimentación, convirtiéndolo en una herramienta potente y adaptable para una amplia variedad de aplicaciones en tiempo real (Torres, 2024).

A continuación, se muestra una gráfica que compara el rendimiento de YOLOv8 con sus predecesores (YOLOv7, YOLOv6 y YOLOv5) en términos de precisión (mAP) y latencia. YOLOv8 muestra una mejora significativa, logrando mayor precisión con menos parámetros y menor latencia. En todas las métricas, YOLOv8 supera a las versiones anteriores, ofreciendo modelos más pequeños, rápidos y eficientes sin comprometer la precisión, lo que resalta su avance en la serie YOLO.

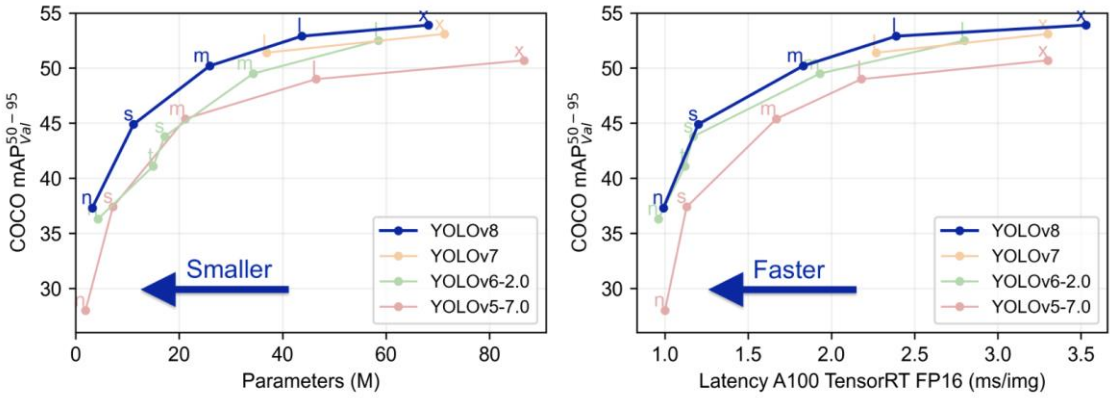


Ilustración 30. Rendimiento de las diferentes versiones del algoritmo YOLO utilizando el conjunto de datos MS COCO (Ultralytics/YOLOv8, 2023)

Capítulo 5. METODOLOGÍA PROPUESTA

En este capítulo, presentamos la metodología propuesta para desarrollar y evaluar nuestro modelo de detección de objetos utilizando la técnica de transferencia de aprendizaje. Esta metodología se estructura en varias etapas clave que aseguran un desarrollo eficiente y efectivo del modelo.

5.1 TRANSFERENCIA DE APRENDIZAJE

En el campo del aprendizaje automático, entrenar redes neuronales convolucionales (CNNs) desde cero puede ser costoso y lento, ya que requiere grandes cantidades de datos y recursos computacionales. La técnica de transferencia de aprendizaje consiste en reutilizar modelos pre-entrenados en grandes conjuntos de datos para tareas específicas, mejorando la eficiencia y precisión sin necesidad de empezar desde cero. Este enfoque acelera el desarrollo y permite obtener buenos resultados con menos datos y en menos tiempo.

A la hora de seleccionar el modelo, lo ideal es escoger uno cuya tarea original esté relacionada con la nueva tarea a realizar. Cuanto más relacionado esté, más útiles serán los conocimientos y habilidades del modelo para el nuevo entrenamiento.

Una vez seleccionado el modelo a utilizar, tendremos que configurarlo. Es importante saber que las primeras capas de una CNN tienden a aprender características generales como bordes, texturas y patrones, mientras que las capas más profundas capturan características más específicas de la tarea para la cual fue entrenada. Por ello, en la transferencia de aprendizaje, se suelen congelar las primeras capas (manteniéndolas fijas) y eliminar las capas finales para añadir nuevas capas que se adapten a la naturaleza de la nueva tarea (AWS, s.f.).

A continuación, entrenaremos el modelo con nuestra base de datos para que se adapte a la nueva tarea. Tras supervisar y evaluar el rendimiento del modelo durante el entrenamiento, se procede a ajustar los hiper-parámetros para mejorar aún más los resultados. A diferencia de los pesos, los hiper-parámetros no se aprenden de los datos. Están preestablecidos y

desempeñan un papel crucial a la hora de determinar la eficiencia y eficacia del proceso de entrenamiento.

Para entrenar nuestro modelo hemos decidido utilizar el algoritmo YOLO debido a su capacidad para detectar objetos a gran velocidad. La plataforma en línea GitHub nos ofrece una gran cantidad de recursos para trabajar con YOLO. Entre los repositorios más destacados se incluyen versiones como YOLOv3, YOLOv5, YOLOv7, y la más reciente, YOLOv8. Estos repositorios contienen código fuente, modelos pre-entrenados, instrucciones detalladas y ejemplos para entrenar, validar y desplegar estos modelos.

En el siguiente capítulo se compararán los resultados obtenidos con la versión YOLOv8n del algoritmo, ajustando los hiper-parámetros para optimizar su rendimiento.

5.1.1 FLUJO DE TRABAJO

Aunque cada aplicación de red neuronal es única, el proceso de transferencia de aprendizaje generalmente sigue estos pasos:

1. Recopilación de datos: Reunir una base de datos relevante para el problema.
2. Preprocesamiento: Definición de los cuadros delimitadores conocidos como 'bounding boxes' mediante herramientas de etiquetado para cada objeto en las imágenes.
3. División de datos: Separar los datos en conjuntos de entrenamiento, validación y prueba.
4. Configuración del algoritmo: Establecer las opciones del algoritmo de entrenamiento para optimizar el rendimiento (hiper-parámetros).
5. Entrenamiento de la red: Entrenar la red neuronal con los datos de entrenamiento.
6. Validación: Evaluar los resultados utilizando el conjunto de validación.
7. Evaluación final: Analizar el rendimiento del modelo con el conjunto de prueba.

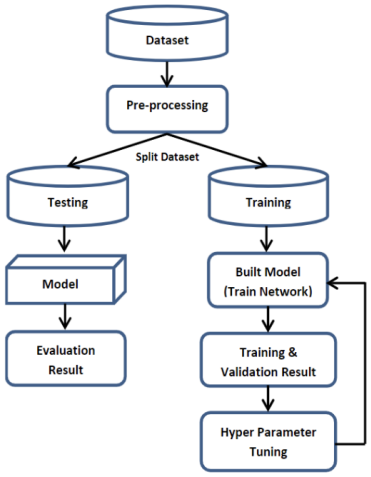


Ilustración 31. Flujo de trabajo a la hora de reentrenar una red neuronal convolucional

5.2 LIBRERÍAS UTILIZADAS

En el ámbito del aprendizaje profundo y la visión por computadora, el uso de herramientas avanzadas ha simplificado enormemente el desarrollo de modelos complejos. Python, con su rica variedad de librerías, ofrece a los desarrolladores una plataforma poderosa para crear modelos de alta precisión y eficiencia de manera rápida. Estas librerías proporcionan los recursos necesarios para gestionar datos, entrenar modelos, y visualizar resultados, optimizando el proceso de creación y ajuste de modelos de visión por computadora.

En el desarrollo del modelo de detección de objetos, varias librerías juegan un papel crucial. La librería 'os' facilita la gestión de directorios y rutas de archivos, permitiendo una organización eficiente de los datos y modelos. 'Numpy' es fundamental para realizar operaciones matemáticas y manejar matrices, aspectos esenciales en el procesamiento de imágenes. Por otro lado, la librería 'cv2' (OpenCV) permite la manipulación y procesamiento de videos, incluyendo la captura y escritura de fotogramas.

El núcleo del desarrollo se centra en la librería 'ultralytics', que simplifica la implementación de las diferentes versiones del algoritmo YOLO, especialmente de su última versión YOLOv8. Esta librería está diseñada para facilitar diversas tareas de visión por computadora,

como la detección de objetos, segmentación, clasificación, estimación de poses y seguimiento de objetos en tiempo real, todo dentro de un entorno unificado. Gracias a 'ultralytics', es posible entrenar nuestro modelo con un conjunto de datos personalizado, y luego aplicarlo para detectar y rastrear objetos en videos sacados de cámaras de seguridad.

Para la visualización de resultados y el monitoreo del entrenamiento, utilizamos 'TensorBoard'. Esta herramienta, integrada en la librería 'TensorFlow', permite visualizar en tiempo real las métricas del entrenamiento, facilitando así la evaluación del rendimiento del modelo y los ajustes necesarios para optimizarlo.

En resumen, las librerías utilizadas permiten desarrollar y entrenar nuestro modelo de manera eficiente, manejando grandes volúmenes de datos y facilitando tanto el procesamiento de imágenes como la visualización de resultados. Estas herramientas son esenciales en el desarrollo de modelos de visión por computadora, permitiendo abordar problemas complejos con soluciones avanzadas y efectivas.

5.3 IMPLEMENTACIÓN

5.3.1 BASE DE DATOS

La base de datos se ha creado a partir de imágenes reales tomadas de internet y contiene 1350 imágenes a color y en blanco y negro, divididas en dos clases: personas y coches. Es crucial balancear los datos para asegurar que el número de imágenes para cada clase sea el mismo, evitando que la red aprenda más de una categoría que de otra, lo cual podría afectar su rendimiento.

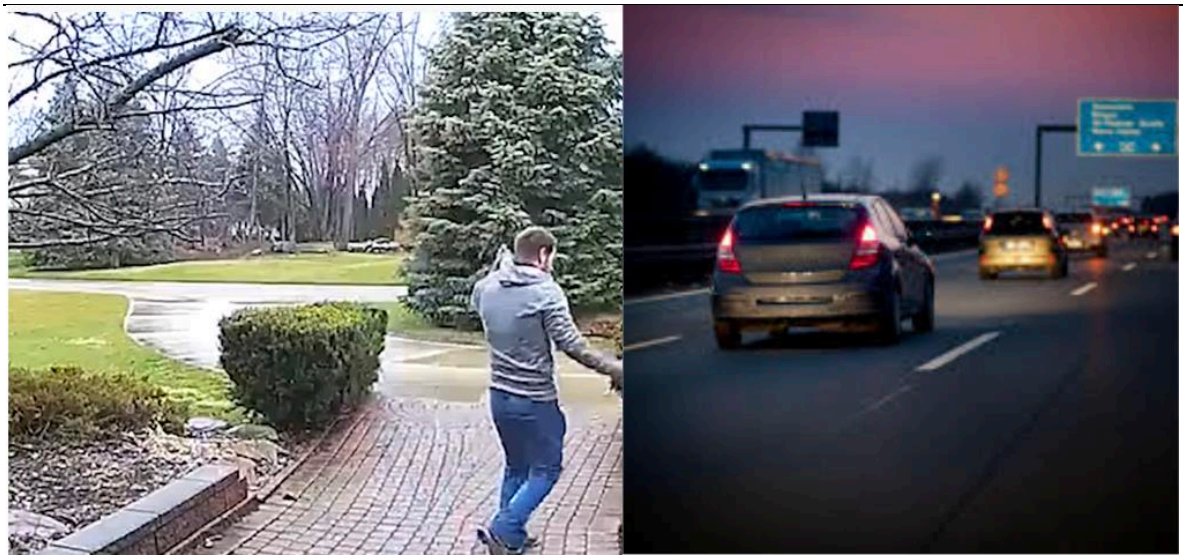


Ilustración 32. Clase 0: persona, clase 1: Coche

5.3.2 PREPROCESAMIENTO

Antes de entrenar el modelo, es necesario definir las áreas específicas donde se encuentran los objetos en las imágenes, proporcionando a la red la información necesaria para reconocer y localizar estos objetos. Para ello, hemos utilizado 'MakeSense', una herramienta en línea que facilita el etiquetado de imágenes para la creación de datasets de entrenamiento en proyectos de visión por computadora. Los usuarios pueden subir sus imágenes y utilizar su interfaz para dibujar y definir cuadros delimitadores alrededor de los objetos de interés. Además, nos permite exportar las anotaciones en formato *.txt* para su uso en el entrenamiento del modelo.

Sin etiquetas precisas, el modelo no sería capaz de asociar correctamente las características visuales con las clases de objetos, resultando en un rendimiento deficiente en la detección de objetos durante el entrenamiento.

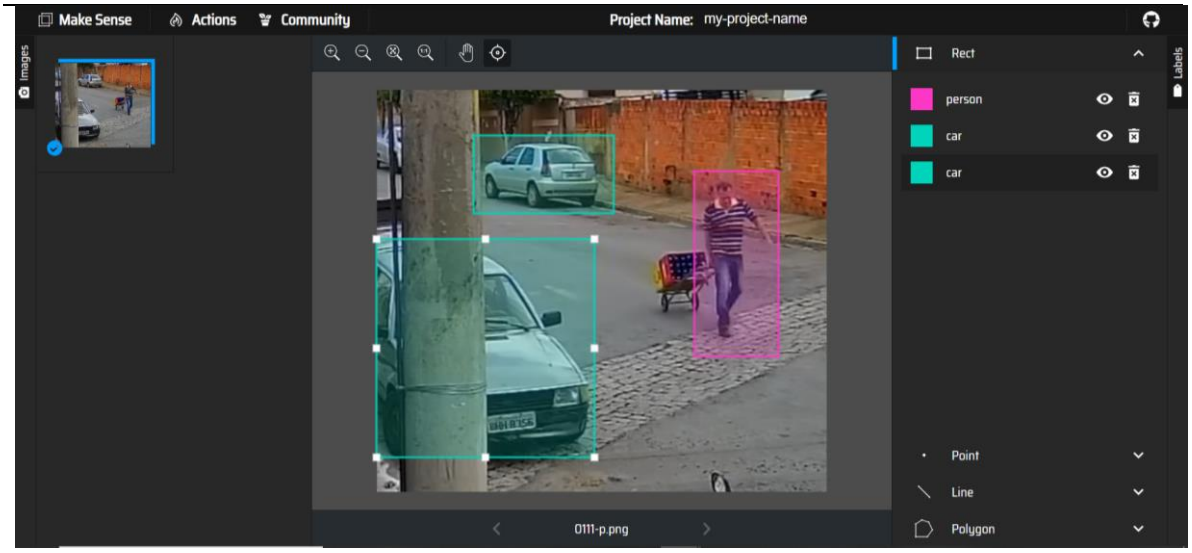


Ilustración 33. Interfaz de la herramienta 'MakeSense' utilizada para la definición de los cuadros delimitadores

```
1 0.360384 0.214612 0.266354 0.168950
1 0.239664 0.651826 0.479328 0.527397
0 0.800580 0.418950 0.178925 0.404110
```

Ilustración 34. Contenido del archivo exportado en formato *.txt*

5.3.3 DIVISIÓN DE DATOS

Dividimos la base de datos aleatoriamente en un 75% para entrenamiento y un 25% para validación, resultando en 1003 imágenes para entrenamiento y 334 para validación. Posteriormente, probaremos el modelo utilizando grabaciones de incidentes en infraestructuras remotas, así como flujos de video en tiempo real.

5.3.4 HIPER-PARÁMETROS

Los hiper-parámetros son variables cuya configuración no se puede derivar directamente de los datos, pero son fundamentales para el entrenamiento y aprendizaje eficiente del modelo. Estos parámetros determinan cómo se optimiza el modelo durante el entrenamiento, influyendo en su comportamiento y capacidad de aprendizaje. Una correcta configuración de los hiper-parámetros puede mejorar significativamente el rendimiento del modelo, mientras que una mala configuración puede causar problemas como el sobreajuste o el subajuste, afectando negativamente la eficiencia y efectividad del aprendizaje (MathWorks, s.f.).

El uso de '**mini-batches**', o pequeños lotes, permite dividir los datos de entrenamiento en partes manejables para cada iteración del entrenamiento. Esto facilita el cálculo del error y el gradiente, haciéndolo más eficiente y menos costoso computacionalmente. En cada iteración se utiliza un 'mini-batch' diferente, lo que permite que el error y el gradiente calculados sean aproximaciones representativas del conjunto completo de datos de entrenamiento. El tamaño del 'mini-batch' es un hiperparámetro clave que afecta la eficiencia del proceso de aprendizaje del modelo.

El número de épocas o '**epochs**' nos indica cuántas veces se ha procesado el conjunto completo de datos de entrenamiento durante el entrenamiento de la red neuronal. Un número adecuado de épocas asegura que la red tenga suficiente exposición a los datos para aprender, pero sin llegar a sobreajustar.

El '**learning rate**' o tasa de aprendizaje es un hiper-parámetro que determina el tamaño de los pasos que el algoritmo de optimización da para minimizar la función de pérdida en cada iteración. Una tasa de aprendizaje alta puede acelerar la convergencia, pero puede saltar sobre los mínimos locales. En cambio, una tasa de aprendizaje baja asegura una convergencia más precisa, aunque más lenta. Ajustar correctamente la tasa de aprendizaje es esencial para evitar el subajuste y el sobreajuste, y técnicas como la reducción gradual pueden mejorar significativamente la optimización del modelo.

5.3.5 PROCESO DE ENTRENAMIENTO Y VALIDACIÓN

Para el entrenamiento y validación del algoritmo YOLO, hemos creado un cuaderno de Google Colab titulado 'TrainYOLOv8nCustomDataset.ipynb'. Optamos por 'Google Colab' en lugar de entornos como 'VSCode' o Anaconda debido a las ventajas significativas que ofrece: acceso gratuito a GPUs potentes, lo que acelera el proceso de entrenamiento; integración fluida con Google Drive para un almacenamiento y acceso sencillos a los datos y resultados; y la ausencia de necesidad de configuración local, evitando problemas de compatibilidad de dependencias debido a diferentes versiones de paquetes y bibliotecas.

Evaluaremos el rendimiento de nuestra red YOLOv8 reentrenada utilizando 'TensorBoard'. Esta herramienta, mencionada previamente, permite monitorear y visualizar métricas del entrenamiento, ofreciendo gráficos detallados de la pérdida, la precisión, histogramas de activaciones y distribuciones de parámetros. Para ello, configuraremos 'TensorBoard' para registrar y visualizar estas métricas a lo largo del proceso de entrenamiento, facilitando así una evaluación exhaustiva y continua del rendimiento del modelo.

5.3.6 EVALUACIÓN FINAL

Por último, evaluamos el rendimiento del modelo utilizando un conjunto de prueba compuesto por varios videos capturados por cámaras de seguridad en ubicaciones remotas durante incidentes reales. El objetivo es que el modelo sea capaz de detectar tanto coches como personas en áreas de acceso restringido. Adicionalmente, probaremos nuestro modelo utilizando la cámara del ordenador para realizar detecciones en tiempo real, verificando así su eficacia en condiciones de uso práctico.

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se llevarán a cabo diversas pruebas experimentales para evaluar el rendimiento del modelo YOLOv8n bajo diferentes configuraciones de hiper-parámetros. Se presentarán tablas que detallan los parámetros utilizados en cada experimento y los resultados obtenidos en términos de precisión y eficiencia. Además, se analizarán las métricas de entrenamiento y validación para identificar las configuraciones óptimas y evitar problemas como el sobreajuste. Posteriormente, se evaluará el modelo utilizando videos grabados por cámaras de seguridad para simular situaciones reales de vigilancia, verificando su capacidad de detección y seguimiento de objetos en condiciones dinámicas. Finalmente, se evaluará el modelo en condiciones de video en tiempo real, proporcionando una visión completa de su rendimiento y robustez en aplicaciones prácticas de vigilancia y seguridad.

6.1 PRUEBAS EXPERIMENTALES

Llevamos a cabo el proceso experimental asignando diferentes valores a cada hiper-parámetro con el fin de evaluar el rendimiento del modelo en diferentes condiciones y determinar los valores óptimos para cada uno de ellos. De esta manera, se pretende obtener una red neuronal con una tasa de precisión elevada y eficiente en términos de tiempo y recursos.

La siguiente tabla presenta los parámetros utilizados en cada uno de los experimentos realizados. Se han variado el tamaño del lote (‘batch size’), el número de épocas de entrenamiento (‘epochs’), y la tasa de aprendizaje (‘learning rate’).

Test	Batch size	# of epochs	Learning rate
1	16	30	0.01
2	16	60	0.01
3	16	90	0.01
4	32	30	0.0001
5	32	60	0.0001
6	32	90	0.0001

Tabla 1. Tabla que recoge los valores usados para cada hiper-parámetro en los diferentes ensayos

Un mayor número de épocas permiten que el modelo entrene durante más tiempo, lo que puede ayudar a alcanzar un mejor ajuste en los datos de entrenamiento. Sin embargo, después de un cierto número de épocas, el modelo puede comenzar a sobre ajustarse ('overfitting'). Por ello, será crucial monitorear las métricas de validación para identificar el punto en el cual el modelo comienza a sobre ajustarse y detener el entrenamiento en ese momento para evitar el 'overfitting'.

A mayor tamaño de lote, las actualizaciones de gradiente serán más estables, pero también requerirá más memoria y recursos computacionales. Por otro lado, un tamaño de lote más pequeño puede hacer que el modelo aprenda más rápido, aunque con más ruido en las actualizaciones. Experimentar con tamaños de lote intermedios como 16 o 32 puede ayudar a encontrar un equilibrio adecuado entre la velocidad de aprendizaje y la estabilidad del entrenamiento.

Una tasa de aprendizaje alta puede hacer que el modelo converja más rápidamente, pero puede saltar sobre mínimos óptimos. En cambio, una tasa de aprendizaje baja puede resultar en una convergencia más estable pero más lenta. Utilizar técnicas de programación de la tasa de aprendizaje ('learning rate scheduling') puede ser beneficioso, ya que permiten reducir la tasa de aprendizaje de manera gradual durante el entrenamiento, mejorando así la precisión y estabilidad del modelo.

6.2 PROCESO DE EVALUACIÓN

En una primera instancia valoraremos los resultados obtenidos tras los procesos de entrenamiento y validación según las métricas estudiadas en el capítulo 4. Una vez definidos los modelos con mejores resultados, se probarán todos los ensayos frente a los videos grabados por las cámaras de seguridad que capturaron incidencias. El modelo que mejor realice la detección y seguimiento de la persona/coche, según el video, será el que utilicemos para las pruebas con flujos de video en tiempo real. El objetivo final de nuestro trabajo será obtener un modelo capaz de reconocer posibles incidencias antes de que estas lleguen a ocurrir.

6.2.1 EVALUACIÓN SEGÚN LAS MÉTRICAS

Tras pre-entrenar la red YOLOv8n con los diferentes hiper-parámetros que se muestran en la tabla presentada en el apartado anterior, 'ultralytics' nos proporciona un directorio de ejecución llamado 'runs' que contiene los pesos finales del modelo, un archivo con la configuración de los hiperparámetros y, lo más importante, un archivo CSV con los datos detallados de las métricas de entrenamiento y validación por época.

A continuación, se definen brevemente las métricas:

- La pérdida o error de caja ('box_loss') mide la discrepancia entre las cajas predichas y las cajas verdaderas.
- La pérdida o error de clasificación ('cls_loss') mide la discrepancia entre las etiquetas de clase predichas y las etiquetas verdaderas.
- La pérdida o error de regresión de distribución ('dfl_loss') se utiliza para mejorar la precisión de la predicción de las cajas.
- La precisión (precisión) mide la proporción de verdaderos positivos sobre el total de positivos predichos.
- La sensibilidad ('recall') mide la proporción de verdaderos positivos sobre el total de positivos reales.

- La mAP50 es la media de la Precisión Promedio ('mean Average Precision') a un umbral IoU del 50%.
- La mAP50-95 es la media de la Precisión Promedio en un rango de umbrales IoU de 0.5 a 0.95 (con pasos de 0.05).

Para comparar la evolución de los distintos ensayos, se adjuntan las siguientes gráficas que muestran la precisión y el error de clasificación del modelo a lo largo del proceso de entrenamiento y validación.

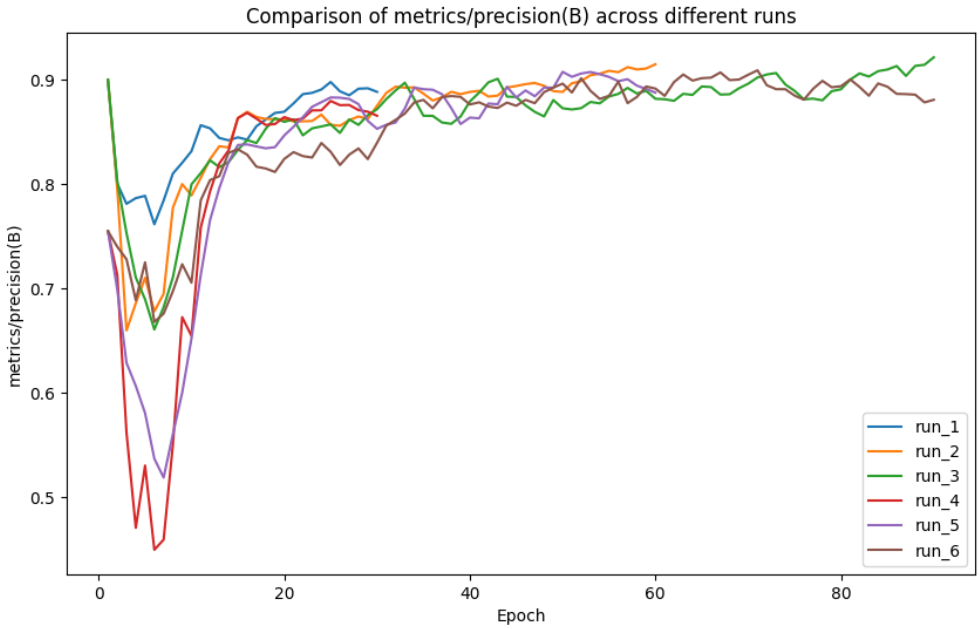


Ilustración 35. Evolución de la precisión de los modelos a lo largo de las épocas

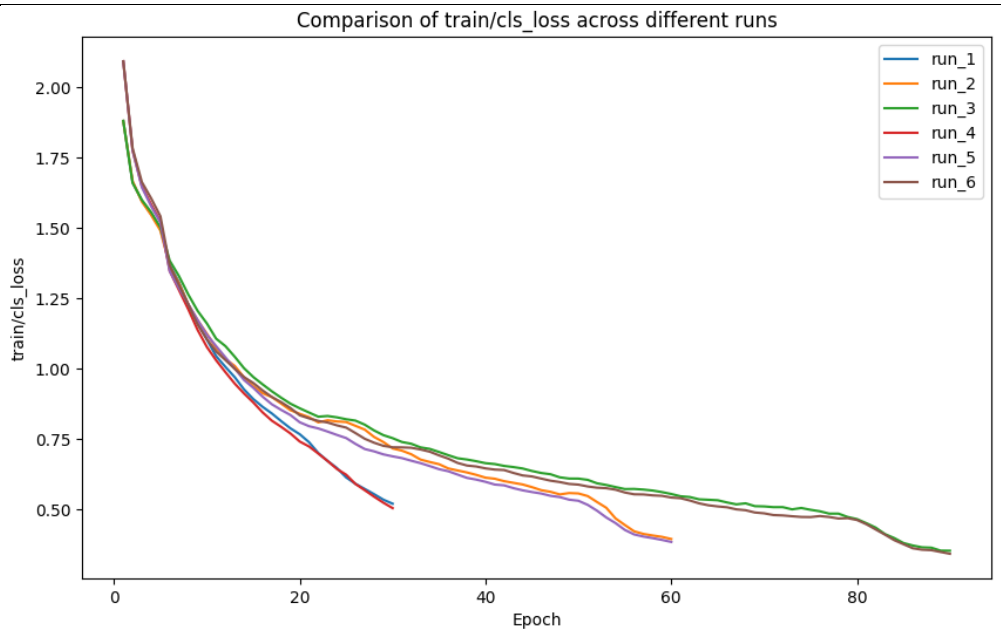


Ilustración 36. Evolución del error de entrenamiento de los modelos a lo largo de las épocas

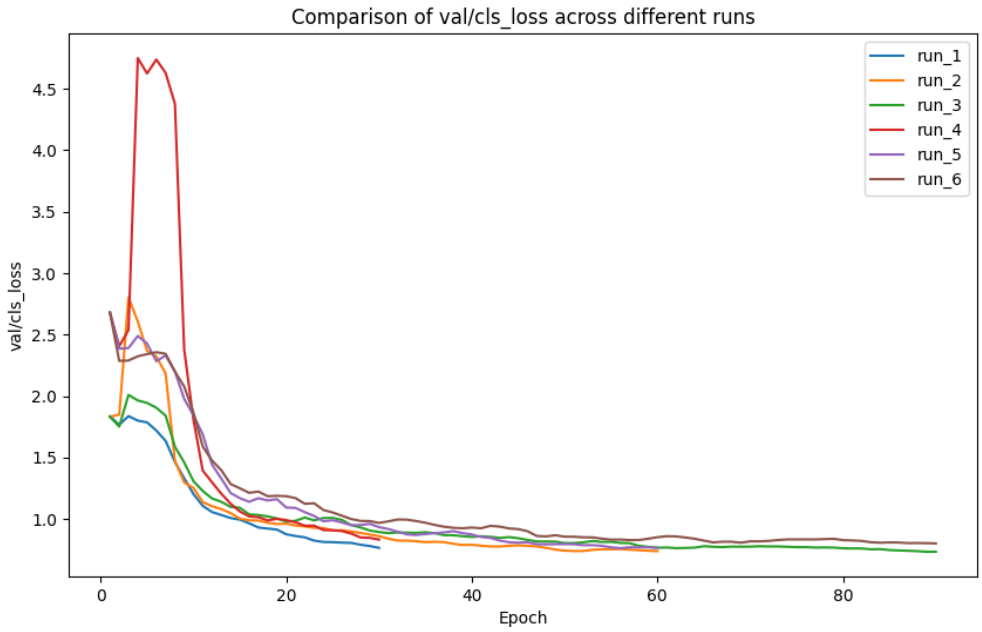


Ilustración 37. Evolución del error de validación de los modelos a lo largo de las épocas

Aunque el modelo aprende más rápido en algunos ensayos que en otros debido a la diferencia en tamaño de lote y tasa de aprendizaje, todos alcanzan resultados muy parecidos. Por ello, a la hora de evaluarlos nos basaremos en los resultados obtenidos en la última época.

A continuación, se adjunta una tabla con dichos resultados, los cuales procedemos a comentar.

Test	Train			Validation			Metrics			
	box_loss	cls_loss	dfl_loss	box_loss	cls_loss	dfl_loss	precision	recall	mAP50	mAP50-95
1	0.782	0.482	1.096	1.294	0.749	1.579	0.891	0.787	0.870	0.556
2	0.674	0.383	1.032	1.327	0.739	1.638	0.926	0.786	0.884	0.549
3	0.611	0.357	1.005	1.322	0.731	1.677	0.937	0.767	0.881	0.557
4	0.764	0.476	1.072	1.334	0.820	1.568	0.892	0.787	0.871	0.556
5	0.666	0.367	1.018	1.378	0.759	1.625	0.875	0.821	0.877	0.553
6	0.594	0.339	0.982	1.407	0.801	1.706	0.888	0.810	0.866	0.545

Tabla 2. Errores de entrenamiento y validación finales de los ensayos realizados; métricas de los modelos finales

De la etapa de entrenamiento podemos ver como a medida que entrenamos el modelo con un mayor número de épocas disminuye la pérdida o error. Además, se puede apreciar una ligera disminución de las pérdidas en los modelos entrenados con un 'batch size' mayor y 'learning rate' menor (ensayos 4, 5 y 6), respecto de los entrenados para un 'batch size' menor y 'learning rate' mayor (ensayos 1, 2, y 3).

Test	Batch size	# of epochs	Learning rate	Train		
				box_loss	cls_loss	dfl_loss
1	16	30	0.01	0.782	0.482	1.096
2	16	60	0.01	0.674	0.383	1.032
3	16	90	0.01	0.611	0.357	1.005
4	32	30	0.0001	0.764	0.476	1.072
5	32	60	0.0001	0.666	0.367	1.018
6	32	90	0.0001	0.594	0.339	0.982

Tabla 3. Errores de entrenamiento finales de los ensayos realizados

En la etapa de validación, a diferencia del comportamiento del error caja y de regresión de distribución durante el entrenamiento se observa que, al entrenar el modelo con un mayor número de épocas, el error no mejora, sino que incluso llega a aumentar. Sin embargo, las pruebas que utilizan un menor tamaño de lote y una mayor tasa de aprendizaje (pruebas 1, 2 y 3) muestran mejores resultados en comparación con las demás. El error de clasificación sí que mejora con el número de épocas, excepto en el caso de la prueba 6, en el que parece que el modelo comienza a sobreajustarse, ya que el error es mayor que en la prueba anterior (prueba 5), la cual tiene 30 épocas menos.

Test	Batch size	# of epochs	Learning rate	Validation		
				box_loss	cls_loss	dfl_loss
1	16	30	0.01	1.294	0.749	1.579
2	16	60	0.01	1.327	0.739	1.638
3	16	90	0.01	1.322	0.731	1.677
4	32	30	0.0001	1.334	0.820	1.568
5	32	60	0.0001	1.378	0.759	1.625
6	32	90	0.0001	1.407	0.801	1.706

Tabla 4. Errores de validación finales de los ensayos realizados

A continuación, observamos las métricas que nos proporciona el modelo.

Test	Batch size	# of epochs	Learning rate	Metrics			
				precision	recall	mAP50	mAP50-95
1	16	30	0.01	0.891	0.787	0.870	0.556
2	16	60	0.01	0.926	0.786	0.884	0.549
3	16	90	0.01	0.937	0.767	0.881	0.557
4	32	30	0.0001	0.892	0.787	0.871	0.556
5	32	60	0.0001	0.875	0.821	0.877	0.553
6	32	90	0.0001	0.888	0.810	0.866	0.545

Tabla 5. Métricas de los modelos finales

La precisión, proporción de verdaderos positivos sobre el total de positivos predichos, es bastante parecida para los entrenamientos realizados con 30 épocas. Sin embargo, esta precisión mejora a medida que incrementamos el número de épocas en modelos entrenados con un menor tamaño de lote y mayor tasa de aprendizaje (pruebas 1, 2 y 3). Mientras que, empeora a medida que incrementamos el número de épocas en modelos entrenados con un mayor tamaño de lote y menor tasa de aprendizaje (pruebas 5, 6 y 7).

De la sensibilidad, proporción de verdaderos positivos sobre el total de positivos reales, observamos lo contrario. Siendo mejores los resultados en las pruebas de mayor tamaño de lote y menor tasa de aprendizaje (pruebas 5, 6 y 7).

Esto se debe a que un modelo puede tener alta precisión, pero baja 'recall', o viceversa. Por ejemplo, si un modelo es muy conservador y solo detecta objetos cuando está muy seguro, tendrá alta precisión, pero puede perder muchos objetos, resultando en baja 'recall'. Por el contrario, si un modelo detecta muchos objetos, incluidos muchos falsos positivos, tendrá alta 'recall' pero baja precisión.

Por ello, para identificar que modelo obtiene unos mejores resultados utilizaremos la precisión media promedio, 'mAP', ya que se basa en la integración del área bajo la curva 'precision-recall'. Esta curva muestra la relación entre precisión y sensibilidad a diferentes niveles de confianza 'IoU' y en todas las clases de objetos. Proporciona una evaluación detallada de cómo el rendimiento del modelo varía con el umbral de confianza, permitiendo una comprensión más profunda del 'trade-off' entre precisión y 'recall'.

El 'mean Average Precision' ('mAP') al 50% es la precisión media calculada a un umbral de intersección sobre unión ('IoU') del 50%. Mientras que el mAP50-95 es una métrica más estricta que considera un rango de umbrales de 'IoU' que van desde el 50% hasta el 95% con pasos del 5%. Proporciona una evaluación más completa del rendimiento del modelo en la detección de objetos a diferentes niveles de coincidencia.

Los ensayos que han obtenido mejores resultados en base al mAP50 han sido el 2 y el 3, con un 88.4% y 88.1% respectivamente. Mientras que, los ensayos con mejores resultados en base al mAP50-95 son el 3 y en 4, con un 55.7% y 55.6% respectivamente.

A primera vista, puede que los resultados para el mAP50-95 nos parezcan muy bajos, sin embargo, si vamos a la web de 'ultralytics', encontramos que el valor medio del mAP50-95 para la red YOLOv8n en los datos de validación está en torno al 37.3%. Por lo que nuestro modelo trabaja mejor que la media.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Ilustración 38. Tabla de características de las diferentes versiones de YOLOv8 (Ultralytics/YOLOv8, 2023)

Entre los diferentes tamaños de YOLOv8, se decidió utilizar el modelo nano, al requerir una menor carga computacional. Sin embargo, el 'mAP' mejora a medida que aumenta el tamaño del modelo, siendo YOLOv8x el más preciso.

6.2.2 EVALUACIÓN DEL MODELO APLICADO A LOS VIDEOS

Aunque los resultados obtenidos al reentrenar el modelo sean aparentemente buenos, durante el entrenamiento y la validación, esto no implica un buen funcionamiento del modelo en la vida real. Por ello, aunque hayamos definido los ensayos que ofrecen mejores resultados en función de las métricas, vamos a probar todos ellos frente a videos grabados por cámaras de seguridad que capturaron incidencias para simular qué hubiese ocurrido en un entorno real.

La tabla que se muestra a continuación recoge el porcentaje de detección del objeto a lo largo del video. Si nos fijamos, nos daremos cuenta de que los algoritmos que mejor detectan los objetos no concuerdan con los que obtuvieron los mejores resultados en el apartado anterior.

La tabla que se muestra a continuación recoge el porcentaje de detección del objeto a lo largo del video. Al analizarla, observamos que los algoritmos que mejor detectan los objetos en los videos no coinciden con los que obtuvieron los mejores resultados en el apartado anterior. Esta discrepancia puede deberse a que el entorno controlado del entrenamiento y la validación no siempre refleja la complejidad y variabilidad de los escenarios reales capturados en los videos. Además, la optimización durante el entrenamiento puede llevar al modelo a sobreajustarse a los datos de entrenamiento y validación, reduciendo su efectividad en condiciones no vistas previamente.

Test	Batch size	# of epochs	Learning rate	Percentage of detection			
				Video 1 (Class 1 - car)	Video 2 (Class 1 - car)	Video 3 (Class 0 - person)	Average
1	16	30	0.01	42.60%	19.69%	37.60%	33.30%
2	16	60	0.01	31.90%	20.18%	36.00%	29.36%
3	16	90	0.01	23.81%	12.80%	39.60%	25.40%
4	32	30	0.0001	42.51%	22.09%	36.80%	33.80%
5	32	60	0.0001	39.92%	25.72%	38.40%	34.68%
6	32	90	0.0001	45.30%	23.20%	40.20%	36.23%

Tabla 6. Porcentaje de fotogramas en los que el objeto fue detectado a lo largo de diferentes videos para cada ensayo

Volviendo a los resultados de los ensayos, por un lado, podemos ver cómo el modelo entrenado con distintos hiper-parámetros es capaz de detectar los objetos en todos los casos. Sin embargo, los ensayos con diferentes configuraciones de hiper-parámetros muestran variaciones en la precisión de detección, con algunos ensayos superando a otros en rendimiento. Por otro lado, dependiendo del video, el rendimiento del ensayo varía significativamente. Esta variabilidad en el rendimiento de los modelos según el video indica que los modelos no son lo suficientemente robustos a distintos ambientes.

¿Por qué ocurre esto? La variabilidad en el rendimiento del modelo de un ensayo concreto en diferentes videos puede deberse a que este no sea igualmente efectivo en diferentes ambientes. La diversidad y representatividad de los datos de entrenamiento son cruciales. Si

el conjunto de datos de entrenamiento no cubre adecuadamente las variaciones presentes en los videos de prueba, el modelo tendrá dificultades para generalizar.

Un modelo puede haber aprendido a detectar coches bien en condiciones de buena iluminación, pero puede fallar en escenas con poca luz o condiciones cambiantes. Por ello, es muy importante que el modelo sea expuesto a suficientes ejemplos de cada tipo de entorno (urbanos, rurales, interiores, exteriores), iluminación (buena/mala, natural/artificial) y condicientes ambientales (sol, lluvia, tormenta, niebla) durante el entrenamiento.

No solo es importante el ambiente, sino también la complejidad de la escena. Videos con mucho movimiento o escenas complejas pueden ser más difíciles de analizar correctamente. Además, diferentes ángulos de cámara y perspectivas pueden afectar la detección. Por ejemplo, un modelo entrenado predominantemente en vistas frontales puede no funcionar tan bien en vistas aéreas o laterales. Asimismo, un modelo que se haya ajustado demasiado a las características de objetos vistos de cerca puede tener dificultades para reconocer esos mismos objetos cuando están lejos.

En cuanto a la diferencia de rendimientos entre los diferentes ensayos, si el modelo ha sido entrenado durante demasiadas épocas o con una tasa de aprendizaje inadecuada, puede haberse sobreajustado a las características específicas de los datos de entrenamiento y, por lo tanto, no generaliza bien los nuevos datos de prueba.

6.2.2.1 Análisis de las detecciones en los videos

A continuación, analizamos las detecciones en los videos para identificar cuándo están fallando y así extraer conclusiones sobre cómo mejorar nuestro modelo.

Observamos que el modelo tiene dificultades para detectar objetos cuando están parcialmente tapados por otros objetos, afectados por una luz intensa (como los faros de un coche), cuando no aparecen en su totalidad en el fotograma o están demasiado lejos.

La siguiente figura, muestra como el modelo no es capaz de detectar el coche en el fotograma de la derecha al encontrarse parcialmente oculto detrás de la puerta.

Ilustración 39. A la izquierda, objeto detectado correctamente. A la derecha, objeto no detectado al estar parcialmente tapado por otro objeto

La siguiente figura, muestra como el modelo no es capaz de detectar el coche en el fotograma de la izquierda debido a la presencia de una luz brillante en el centro de la imagen, causando un contraluz significativo. Los algoritmos de detección pueden tener problemas para distinguir objetos en condiciones de alto contraste o deslumbramiento.

Ilustración 40. A la izquierda, objeto no detectado debido a la fuerte luz que causa un contraluz significativo. A la derecha, objeto detectado correctamente

La siguiente figura, muestra como el modelo no es capaz de detectar la persona en el fotograma de la izquierda, ya que está inclinada y parcialmente oculta detrás de un equipo. Esto puede hacer que el modelo tenga dificultades para detectar a la persona debido a la falta de una silueta completa y clara.

Ilustración 41. A la izquierda, objeto no detectado al no aparecer en su totalidad en el fotograma. A la derecha, objeto detectado correctamente

La siguiente figura, muestra como el modelo no es capaz de detectar el coche en el fotograma de la derecha, debido a que este se ha alejado considerablemente y ocupa una porción muy pequeña de la imagen. A mayor distancia, el coche ocupa menos píxeles y tiene menos detalles visibles, lo que dificulta su detección por el modelo.

Ilustración 42. A la izquierda, objeto detectado correctamente. A la derecha, objeto no detectado al encontrarse demasiado lejos

Para mejorar la capacidad de detección de nuestro modelo, debemos mejorar el conjunto de datos de entrenamiento y validación para incluir más variaciones en iluminación, ángulos y distancias puede ayudar a mejorar la robustez del modelo en situaciones similares. Para ello, es importante considerar las siguientes estrategias:

1. 'Data Augmentation':
   - Rotaciones y Traslaciones: Aplicar rotaciones y traslaciones a las imágenes para simular diferentes ángulos de cámara.
   - Ajustes de Iluminación: Modificar la iluminación en las imágenes para incluir tanto condiciones de alta como de baja luminosidad.
   - Ruido y Desenfoque: Añadir ruido y desenfoque para hacer el modelo más robusto a variaciones en la calidad de la imagen.
2. Entrenamiento en Diversos Entornos: Asegurarse de que el conjunto de datos incluya imágenes de una variedad de entornos, así como tomadas en diferentes condiciones climáticas para mejorar la adaptabilidad del modelo.
3. Entrenamiento con objetos parcialmente ocultos y utilizar técnicas de segmentación semántica para entrenar el modelo a reconocer partes de objetos.

6.2.2.2 Mejora de la visualización

La incapacidad del modelo para detectar el objeto en todos los fotogramas donde este aparece puede causar un efecto de parpadeo en el video. Para mejorar la estabilidad de las detecciones y reducir este parpadeo, implementaremos una lógica de suavizado temporal. La estrategia consiste en mantener un historial de detecciones y ajustar el umbral de detección según las detecciones recientes.

Para ello, se almacenará un historial de las últimas cinco detecciones por clase, permitiendo registrar las detecciones recientes. Si un objeto ha sido detectado en los cuadros anteriores con una puntuación promedio superior al umbral inicial, se reducirá el umbral de detección en un 10% respecto al valor previo. En caso de que el objeto no se detecte durante un período de tiempo, el umbral de detección se incrementará gradualmente hasta volver a su valor inicial.

Este enfoque ayuda a reducir el parpadeo al suavizar las detecciones basadas en la consistencia temporal, asegurando una visualización más fluida y estable en el video resultante.

Tras implementar esta nueva lógica, se obtuvieron los siguientes resultados:

Test	Batch size	# of epochs	Learning rate	New Percentage of detection				
				Video 1 (Class 1 - car)	Video 2 (Class 1 - car)	Video 3 (Class 0 - person)	New Average	Improvement rate
1	16	30	0.01	58.17%	21.48%	38.40%	39.35%	+6pp
2	16	60	0.01	49.49%	22.40%	37.60%	36.50%	+7pp
3	16	90	0.01	28.98%	16.98%	42.30%	29.42%	+4pp
4	32	30	0.0001	52.50%	26.28%	36.80%	38.53%	+5pp
5	32	60	0.0001	51.19%	27.82%	40.80%	39.94%	+5pp
6	32	90	0.0001	53.63%	27.32%	42.50%	41.15%	+5pp

Tabla 7. Porcentaje de fotogramas en los que el objeto fue detectado a lo largo de diferentes videos para cada ensayo tras el suavizado del umbral

Como se puede observar, la detección de los objetos ha mejorado considerablemente gracias al suavizado, incrementando el rendimiento de los modelos en un promedio de 5 puntos porcentuales (‘pp’).

El ensayo 6 es el que obtiene los mejores resultados en la detección de objetos en los videos, por lo que este modelo será el que se utilice para flujos de video en tiempo real.

6.2.3 EVALUACIÓN DEL MODELO APLICADO A FLUJOS DE VIDEO EN TIEMPO REAL

En la última etapa del proyecto, probamos el modelo YOLOv8n reentrenado en flujos de video en tiempo real utilizando la cámara del ordenador. El objetivo de esta fase es evaluar el rendimiento del modelo en un entorno dinámico y verificar su capacidad para detectar objetos de manera precisa y rápida.

Para llevar a cabo esta última evaluación hemos utilizado un entorno local, ya que en ‘Google Colab’ no es posible acceder directamente a la cámara web local del usuario debido a las restricciones de seguridad del navegador.

Para ello creamos un entorno virtual con Python 3.8.0 e instalamos el paquete 'ultralytics' en nuestro entorno local. Desarrollamos un script en Python que captura fotogramas de video en tiempo real desde la cámara del ordenador, procesa cada fotograma con el modelo YOLOv8n resultado del ensayo 6 para realizar detecciones y visualiza los resultados en una ventana gráfica.

Durante la prueba, el modelo mostró un desempeño sólido en la detección de objetos en tiempo real. A continuación, se presentan las observaciones clave:

- Velocidad de Procesamiento: La velocidad de procesamiento fue adecuada para un flujo de video en tiempo real. Aunque existe un ligero retraso debido a la capacidad de procesamiento limitada del ordenador, este no afecta significativamente a la detección y el seguimiento de objetos.
- Precisión de Detección: El modelo fue capaz de detectar con precisión personas y vehículos en la mayoría de los casos. La detección se mantuvo consistente incluso con movimientos rápidos y cambios en el entorno.
- Condiciones de Iluminación: El modelo mostró un buen rendimiento bajo condiciones de iluminación variada, aunque tuvo dificultades en escenas con iluminación extremadamente baja o contraluz intenso.
- Ángulos y Perspectivas: Se observó que el modelo mantiene una buena precisión en la detección desde diferentes ángulos de cámara y perspectivas. Aunque la capacidad de detección de personas empeora a distancias mayores.

A continuación, se adjuntan capturas de pantalla realizadas mientras el modelo se ejecutaba sobre la cámara del ordenador. Estas imágenes se tomaron en diferentes ambientes y con distintas condiciones lumínicas.

Ilustración 43. Capturas de pantalla realizadas durante la prueba de video en tiempo real en distintos ambientes

La prueba del modelo YOLOv8n reentrenado en flujos de video en tiempo real demostró su capacidad para operar de manera efectiva en un entorno dinámico. La detección precisa y rápida de objetos confirma que el modelo es adecuado para aplicaciones de seguridad y vigilancia en tiempo real. Sin embargo, se identificaron áreas para mejorar, como la robustez en condiciones de iluminación adversa y la detección a largas distancias. Estas observaciones proporcionan una base sólida para futuros trabajos y mejoras en el modelo.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

Resumen de Logros y Objetivos Cubiertos

En este proyecto, hemos desarrollado un sistema de detección y seguimiento de objetos utilizando librerías gráficas y técnicas avanzadas de visión por computadora, mediante el uso de Redes Neuronales Convolucionales ('CNN') y el algoritmo YOLO ('You Only Look Once'). Los objetivos principales se han cumplido al implementar y validar un modelo capaz de procesar flujos de video en tiempo real, detectando y clasificando objetos con una precisión notable.

Comparación entre Métodos Tradicionales y CNNs

Los métodos tradicionales de procesamiento de imágenes, como los filtros lineales y las técnicas de segmentación, han sido fundamentales en el desarrollo de la visión por computadora. Sin embargo, estos métodos presentan varias limitaciones. En nuestras pruebas, se observó que los métodos tradicionales no capturaban adecuadamente las relaciones complejas en los datos, lo que resultaba en un desempeño inferior en condiciones de variabilidad de iluminación, ángulos y condiciones ambientales. Además, requieren un preprocesamiento manual intensivo y no son tan robustos en entornos con alta variabilidad.

Por otro lado, las Redes Neuronales Convolucionales ('CNN'), y específicamente el algoritmo YOLO ('You Only Look Once'), han demostrado ser significativamente más eficaces. En nuestros experimentos, YOLO permitió la detección en tiempo real gracias a su arquitectura de una sola pasada. La capacidad del modelo para operar en tiempo real lo hace ideal para aplicaciones de vigilancia y monitoreo continuo, como se comprobó en las pruebas con videos de cámaras de seguridad. Esto permite una detección y seguimiento de objetos más robustos y confiables en diversos escenarios, superando claramente a los métodos tradicionales en términos de rendimiento y adaptabilidad.

Limitaciones y Robustez del Modelo

Aunque el modelo desarrollado ha mostrado ser eficaz en múltiples escenarios, se han identificado ciertas limitaciones en su robustez. La variabilidad en el rendimiento del modelo en diferentes videos indica que no es igualmente efectivo en todos los ambientes. Factores como la iluminación, la complejidad de la escena y la distancia de los objetos afectan significativamente la precisión de las detecciones.

El modelo detecta objetos con alta precisión en condiciones de buena iluminación, pero su desempeño disminuye en escenas con poca luz o fuentes de luz intensa que causan contraluz. Los diferentes ángulos de cámara y perspectivas también afectan negativamente la detección, debido a que el modelo ha sido entrenado principalmente con vistas frontales, traseras y laterales, pero no desde todos los ángulos. Además, el modelo presenta dificultades para reconocer objetos cuando están lejos, debido a la pérdida de detalles visuales críticos. Estas observaciones sugieren que el modelo requiere más entrenamiento con variaciones en iluminación, ángulos y distancias para mejorar su robustez en situaciones del mundo real.

Futuras Mejoras y Continuación del Proyecto

Para mejorar la capacidad de detección y la robustez del modelo, es fundamental ampliar y diversificar el conjunto de datos de entrenamiento y validación. Incluir más variaciones en iluminación, ángulos y distancias puede hacer que el modelo sea más generalizable y efectivo en diferentes situaciones. Implementar técnicas de 'data augmentation', como rotaciones, traslaciones, ajustes de iluminación, adición de ruido y desenfoque, puede enriquecer el conjunto de datos y aumentar la capacidad del modelo para manejar condiciones variadas. Además, utilizar técnicas de segmentación semántica puede ayudar a que el modelo reconozca partes de objetos incluso cuando están parcialmente ocultos, mejorando así su precisión y fiabilidad.

Finalmente, integrar métodos de aprendizaje continuo permitiría que el modelo se adapte y mejore continuamente a medida que procesa más datos del entorno real, aumentando así su precisión y fiabilidad en aplicaciones de vigilancia y seguridad industrial.

Capítulo 8. BIBLIOGRAFÍA

Akif Cifci, M. (5 de February de 2023). *Neural Network.* Obtenido de Medium: https://themanoftalent.medium.com/neural-network-2b38cba18733

AWS. (s.f.). *What is Transfer Learning?* Obtenido de AWS: https://aws.amazon.com/what-is/transfer-learning/#:~:text=You%20can%20use%20transfer%20learning,for%20faster%20deployment%20into%20production.

Banoula, M. (May de 2023). *simplilearn.com*. Obtenido de https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron

Blanco, I. (Julio de 2024). *Código Trabajo Fin de Master*. Obtenido de GitHub: https://github.com/inesblanco00/202216600

Calvo, D. (20 de Julio de 2017). *DIEGO CALVO.* Obtenido de https://www.diegocalvo.es/red-neuronal-convolucional/convolucion/

computersciencewiki. (27 de February de 2018). Obtenido de Max-pooling / pooling: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

Dumakude, A., & Ezugwu, A. E. (2023). Automated COVID-19 detection with convolutional neural networks. Sci Rep 13, 10607 (2023). https://doi.org/10.1038/s41598-023-37743-4. *Nature*.

Gallego, A. J. (2018). *Bloque 3. Separación de regiones.*

Gerón, A. (2017). *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow.* O´Reilly.

IBM. (2024). *¿Qué es una Red Neuronal?* Obtenido de IBM: https://www.ibm.com/mx-es/topics/neural-networks

Kılıç, İ. (September de 2023). *Medium*. Obtenido de https://medium.com/@ilyurek/perceptron-model-the-foundation-of-neural-networks-4db25b0148d

Kundu, R. (17 de January de 2023). *YOLO: Algorithm for Object Detection Explained [+Examples]*. Obtenido de V7: https://www.v7labs.com/blog/yolo-object-detection

Kurz, L. (2021). *YOLO: You only look once*. Obtenido de cronn: https://blog.cronn.de/en/ai/2021/07/28/yolo-object-detection.html

Liu, W., Anguelov, D., Erhan, D., & Szegedy, C. (2016). *SSD: Single Shot MultiBox Detector*. arXiv:1512.02325v5.

Mahesh, B. (2019). *Machine Learning Algorithms- A Review*. International Journal of Science and Research (IJSR).

Mallick, S. (2017). *Open CV University*. Obtenido de BIG VISION: https://learnopencv.com/bias-variance-tradeoff-in-machine-learning/

MathWorks. (s.f.). Obtenido de https://es.mathworks.com/

Mirkhan, A. (2023). *YOLO Algorithm: Real-Time Object Detection from A to Z*. Obtenido de Kili: https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z#single-shot-object-detection

O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks*. arXiv:1511.08458v2 [cs.NE].

Oppermann, A. (n.d.). *Built in*. Retrieved from https://builtin.com/artificial-intelligence/ai-vs-machine-learning

Rojas Campo, J. (2022). *CCTV: Qué es, sus tipos y equipos que lo componen*. Obtenido de TECNOSeguro: https://www.tecnoseguro.com/faqs/cctv/que-es-cctv#:~:text=De%20acuerdo%20con%20la%20tecnolog%C3%ADa,alcance%20tecnol%C3%B3gico%20y%20su%20proyecci%C3%B3n

Sánchez Anzola, N. (9 de February de 2015). *Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento USD/COP spot intradiario.* Obtenido de ODEON: https://doi.org/10.18601/17941113.n9.04

Shah, D. (7 de Marzo de 2022). *Mean Average Precision (mAP) Explained: Everything You Need to Know*. Obtenido de v7labs: https://www.v7labs.com/blog/mean-average-precision#mean-average-precision-for-object-detection

Singh, S. (5 de January de 2023). *Labellerr.* Obtenido de https://www.labellerr.com/blog/why-is-the-yolo-algorithm-important/

Stallman, R. (2022). *Python with OpenCV3: Computer Vision Course for Beginners.*

Torres, J. (January de 2024). *YOLOv8 Architecture: A Deep Dive into its Architecture*. Obtenido de YOLOv8: https://yolov8.org/yolov8-architecture/

Ultralytics/YOLOv8. (2023). Obtenido de github: https://github.com/ultralytics/ultralytics

Vega, B. (2017). *biblus.es.* Obtenido de https://asignatura.us.es/imagendigital/Tema5-2_SegmentacionRegionesUmbralizacion.pdf

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.* Taiwan: Institute of Information Science, Academia Sinica, Taiwan.

ANEXO I: ALINEACIÓN CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE

Los Objetivos de Desarrollo Sostenible (ODS) creados por las Naciones Unidas, son un conjunto de 17 objetivos mundiales diseñados para "lograr un futuro mejor y más sostenible para todos". Este proyecto puede contribuir a la creación de un futuro más prometedor al estar alineado con varios de los objetivos propuestos para la Agenda 2030.

Este proyecto participa directamente en alcanzar el objetivo 9, también llamado: Industria, innovación e infraestructura. Mediante la propuesta de implementar soluciones de seguridad más avanzadas y eficaces para garantizar la integridad de los entornos físicos, se está contribuyendo al desarrollo de infraestructuras resilientes. Además, la aplicación de tecnologías emergentes como la Inteligencia Artificial y el Aprendizaje Automático como solución integral para abordar los desafíos ante los que nos encontramos en el campo de la seguridad promociona la innovación tecnológica

Por otro lado, el objetivo 11 está ligado directamente con este proyecto, pues busca que las ciudades y comunidades sean lo más sostenibles posibles. Tal y como se ha explicado anteriormente, se pretende crear un sistema capaz de detectar y responder a posibles incidencias en plantas eléctricas distribuidas por la región. La automatización de esta tarea no solo beneficiaría a las empresas del sector eléctrico, sino que también contribuiría a mejorar la seguridad y la eficiencia en áreas industriales, promoviendo comunidades más seguras y sostenibles.

Por último, aunque el enfoque principal del proyecto es la seguridad, al fortalecer la protección de las plantas eléctricas se garantiza un suministro de energía más fiable y estable. Esto contribuye indirectamente al ODS 7 al asegurar el acceso a una energía asequible, segura y sostenible para las comunidades locales y la industria en general.

En resumen, el proyecto no solo aborda la necesidad de seguridad en las infraestructuras críticas, sino que también promueve el desarrollo de infraestructuras resilientes,

comunidades seguras y el acceso a una energía sostenible, alineándose así con los ODS 9, 11 y 7, respectivamente.