UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Reconocimiento de personas con YOLOv8: Reducción de datos y evaluación del aumentado de datos.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Pacheco Millán, Andrés

Tutor/a: Paredes Palacios, Roberto

Cotutor/a externo: Mansanet Sandín, Jorge

CURSO ACADÉMICO: 2024/2025

# Resum

En els últims anys, l'aprenentatge profund ha revolucionat el camp de la visió per computador, permetent importants avanços en tasques com la classificació, segmentació i detecció d'objectes. No obstant això, un dels principals reptes a què s'enfronten aquestes tècniques és la seua elevada dependència de grans volums de dades etiquetades, la seua obtenció pot ser costosa i, en molts casos, inviable. Davant d'aquesta limitació, l'augment de dades es presenta com una estratègia efectiva per a millorar la capacitat de generalització dels models sense necessitat d'ampliar el conjunt de dades original.

Aquest treball se centra en el desenvolupament d'una tècnica automàtica d'augment de dades, que té com a objectiu reduir significativament el temps d'entrenament sense comprometre el rendiment del model. Per a això, s'ha dissenyat una metodologia que permet seleccionar de forma iterativa i eficient un subconjunt de transformacions que contribuïxen positivament al valor del mAP, emprant únicament el 10% de les imatges del conjunt original.

La tècnica ha sigut estudiada específicament en la tasca de detecció de persones, i ha sigut validada sobre el conjunt de dades PersonPath22, un conjunt d'imatges adequat per a aquesta finalitat. A més, s'ha emprat la sèrie de models YOLOv8 com a base per a la detecció.

S'ha implementat una sèrie de versions progressives de la tècnica, avaluant el seu impacte tant en precisió com en temps d'execució. La versió final aconsegueix mantindre valors de mAP pròxims al 70%. Els resultats obtinguts són molt similars als aconseguits amb tècniques com RandAugment, amb una diferència inferior al 2%, però amb un temps d'entrenament 2,5 vegades menor. En comparació amb models entrenats sobre el conjunt complet de dades i sense augment, el cost temporal es redueix fins a 20 vegades, millorant la precisió en quasi un 3%.

Per tant, la tècnica desenvolupada permet optimitzar el procés d'entrenament, reduint significativament els recursos necessaris sense comprometre la qualitat de les prediccions. Això la convertix en una solució pràctica i eficient, especialment adequada per a entorns on la disponibilitat de dades o la capacitat computacional són limitades.

**Paraules clau:** YOLOv8, detecció de persones, augment de dades, visió per computador, aprenentatge profund

---

# Resumen

En los últimos años, el aprendizaje profundo ha revolucionado el campo de la visión por computador, permitiendo importantes avances en tareas como la clasificación, segmentación y detección de objetos. Sin embargo, uno de los principales retos a los que se enfrentan estas técnicas es su elevada dependencia de grandes volúmenes de datos etiquetados, cuya obtención suele ser costosa y, en muchos casos, inviable. Ante esta limitación, el aumento de datos se presenta como una estrategia efectiva para mejorar la capacidad de generalización de los modelos sin necesidad de ampliar el conjunto de datos original.

Este trabajo se centra en el desarrollo de una técnica automática de aumento de datos, que tiene como objetivo reducir significativamente el tiempo de entrenamiento sin comprometer el rendimiento del modelo. Para ello, se ha diseñado una metodología que permite seleccionar de forma iterativa y eficiente un subconjunto de transformaciones que contribuyen positivamente al valor de mAP, empleando únicamente el 10 % de las imágenes del conjunto original.

La técnica ha sido estudiada específicamente en la tarea de la detección de personas, y ha sido validada sobre el conjunto de datos PersonPath22, un conjunto de imágenes que permite cumplir esta tarea. Además, como modelo base de detección se ha empleado la serie de modelos YOLOv8.

Se ha implementado una serie de versiones progresivas de la técnica, evaluando su impacto tanto en precisión como en tiempo de ejecución. La versión final logra mantener valores de mAP cercanos al 70 %. Los resultados obtenidos son muy similares a los obtenidos con técnicas como RandAugment, con una diferencia inferior al 2 %, pero con un tiempo de entrenamiento 2,5 veces menor. En comparación con modelos entrenados sobre el conjunto de datos completo y sin aumentado de datos, el coste temporal logra reducirse hasta 20 veces, logrando mejorar la precisión hasta casi un 3 %.

Por tanto, la técnica desarrollada permite optimizar el proceso de entrenamiento, reduciendo significativamente los recursos necesarios sin comprometer la calidad de las predicciones. Esto la convierte en una solución práctica y eficaz, especialmente adecuada para entornos donde los datos disponibles o la capacidad computacional son limitados.

**Palabras clave:** YOLOv8, detección de personas, aumentado de datos, visión por computador, aprendizaje profundo

# Abstract

In recent years, deep learning has revolutionized the field of computer vision, enabling significant progress in tasks such as classification, segmentation, and object detection. However, one of the main challenges faced by these techniques is their heavy reliance on large volumes of labeled data, which is often costly and, in many cases, unfeasible. To address this limitation, data augmentation emerges as an effective strategy to improve model generalization without the need to expand the original dataset.

This project focuses on the development of an automatic data augmentation technique aimed at significantly reducing training time without compromising model performance. To achieve this, a methodology was designed to iteratively and efficiently select a subset of transformations that positively impact the mAP score, using only 10% of the original images.

The technique was specifically studied in the context of person detection and validated using the PersonPath22 dataset, a collection of images suitable for this task. Furthermore, the YOLOv8 model series was used as the detection framework.

Several progressive versions of the technique were implemented, evaluating their impact on both accuracy and execution time. The final version maintains mAP values close to 70%. The results are very similar to those achieved with techniques such as RandAugment, with a difference of less than 2%, but with a training time reduced by a factor of 2.5. When compared to models trained on the full dataset without augmentation, the training time was reduced by up to 20 times, while improving precision by almost 3%.

As a result, the proposed technique optimizes the training process by significantly reducing resource requirements without compromising prediction quality. This makes it a practical and efficient solution, especially well-suited for scenarios with limited data availability or computational capacity.

**Key words:** YOLOv8, person detection, data augmentation, computer vision, deep learning

# Índice general

# Índice de figuras



# Índice de tablas

# CAPÍTULO 1
# Introducción

En los últimos años, el aprendizaje profundo ha revolucionado el campo de la visión por computador, permitiendo notables avances en distintas tareas como la clasificación, la detección y la segmentación de objetos. Esta revolución ha sido impulsada por un conjunto de factores como el incremento de la capacidad computacional, la aparición de grandes conjuntos de datos con un buen etiquetado y el desarrollo de arquitecturas sofisticadas, capaces de aportar cada vez mejores resultados. En estas tareas de la visión por computador, la detección de objetos ha recibido una gran relevancia, al permitir localizar e identificar instancias de objetos dentro de una imagen. Esta capacidad de detección es esencial en un amplio abanico de aplicaciones reales donde se vuelve crucial localizar el objeto, como la vigilancia, la robótica o la conducción autónoma.

Sin embargo, independientemente del gran funcionamiento y los resultados obtenidos por los modelos actuales, este tipo de sistemas se ve limitado por la necesidad de una gran cantidad de datos con un etiquetado de calidad. Es decir, no únicamente son necesarias imágenes con un buen etiquetado, además se debe contar con una gran cantidad de ellas. Entrenar modelos robustos requiere miles y miles de imágenes que sean capaces de cubrir todas las posibles variaciones que tendrán los objetos, con el objetivo de que el modelo haya visto gran parte de sus posibles variantes antes de comenzar con su etapa de inferencia y, de esta forma, poder alcanzar los resultados esperados. Esta dependencia de grandes conjuntos de datos puede representar un inconveniente en distintas aplicaciones, donde la obtención de estos datos no es una tarea sencilla, como puede ser la medicina. A su vez, la generación de estas imágenes etiquetadas al ser, mayormente, un proceso manual, supone un trabajo realmente costoso y muchas veces inviable.

Ante esto, las técnicas de aumento de datos han surgido como una solución efectiva que logran mejorar la capacidad de generalización y la robustez del modelo, sin necesidad de tener grandes conjuntos de datos para lograr buenos resultados. Esta técnica se basa en la aplicación de transformaciones a las imágenes disponibles, de forma que se generan nuevas muestras sintéticas a utilizar durante el entrenamiento. Algunas de estas transformaciones son rotaciones, escalados, ajustes de contraste, recortes o combinaciones entre imágenes. Gracias a la generación de estas variaciones, el modelo se vuelve más robusto, mejorando su capacidad de detección de ciertos patrones ante distintas situaciones. Por ejemplo, en caso de que no se vea la extensión completa del objeto o aparezca en una orientación distinta.

La aplicación manual de estas técnicas supone establecer magnitudes y un conjunto de transformaciones sin saber si pueden ser realmente buenas para el modelo. Para acabar con este problema, han surgido en los últimos años un conjunto de técnicas para la aplicación de aumentados que intentan automatizar este proceso de selección y combinación de transformaciones. Algunos ejemplos de estas técnicas son AutoAugment o

RandAugment, técnicas que logran encontrar combinaciones de transformaciones que optimizan el rendimiento del modelo.

## 1.1 Motivación

A pesar de los notables avances experimentados por las técnicas de visión por computador, una de sus mayores limitaciones es la dependencia de grandes volúmenes de datos. La aplicación de técnicas de aumento de datos automáticas se presenta como una solución que permite enriquecer los conjuntos de entrenamiento, sin necesidad de realizar el proceso manual de buscar las transformaciones a imágenes que obtienen los mejores resultados.

Estas soluciones automáticas, aunque sean efectivas, no están exentas de limitaciones. Por ejemplo, la técnica AutoAugment, necesita una costosa fase de búsqueda de políticas óptimas para su aplicación, lo que acaba reduciendo la eficiencia.

Este trabajo surge de la necesidad de optimizar el proceso de aumento de datos, no solo en términos de precisión, sino también en lo que respecta a la eficiencia temporal y la reducción de la dependencia de grandes conjuntos de datos. El objetivo es diseñar una técnica que sea capaz de generar resultados competitivos en métricas de precisión utilizando un conjunto de datos reducido, al mismo tiempo que reduzca significativamente los tiempos de ejecución en comparación con las técnicas automáticas existentes.

La motivación central de este trabajo, por lo tanto, es ofrecer una solución que mejore la eficiencia de las técnicas de aumento de datos sin sacrificar el rendimiento del modelo, permitiendo entrenar modelos robustos con menos datos y en menos tiempo.

## 1.2 Objetivos

El objetivo de este trabajo está centrado en desarrollar, implementar y evaluar una técnica de aumento de datos automática que permita mejorar la eficiencia del proceso de entrenamiento en tareas de detección de personas, manteniendo resultados comparables en precisión respecto a métodos estándar y de referencia. En concreto, se plantean los siguientes objetivos específicos:

- Reducir la cantidad de datos necesarios: conseguir valores de mAP similares a los obtenidos con el conjunto de datos completo, pero aplicando la técnica desarrollada a un 10 % del conjunto de imágenes original. Además, se busca que esta reducción en el volumen de datos se traduzca en una mejora significativa en el tiempo de ejecución del proceso de entrenamiento.
- Comparación con técnicas automáticas existentes: lograr un rendimiento en mAP comparable al obtenido con la técnica RandAugment aplicada sobre el 10 % del conjunto de imágenes, pero reduciendo los tiempos de ejecución.

## 1.3 Estructura de la memoria

A partir del contexto de la motivación y los objetivos del proyecto realizado, la memoria seguirá la siguiente estructura. Comenzando por el capítulo 2, se presentarán las tecnologías usadas, así como aquellas que han llevado al desarrollo de las tecnologías usadas, o que presentan muy buenos resultados en el mismo ámbito. En el capítulo 3, se

presentarán el conjunto de datos, la máquina y el modelo para realizar los experimentos, así como la resolución de los problemas surgidos durante su preparación. En el capítulo 4 se presentará la solución propuesta, con las fases por las que ha pasado la solución. La evaluación de los resultados obtenidos se encuentra en el capítulo 5. Las conclusiones del proyecto se presentarán en el capítulo 6. En el capítulo 7 se relacionará el proyecto realizado con los estudios cursados a lo largo de la carrera. Finalmente, en el capítulo 8 se proponen futuras líneas de mejora de la técnica, así como la investigación del funcionamiento de esta técnica en distintos contextos al elegido en este trabajo.

# CAPÍTULO 2
# Estado del arte

## 2.1 Detección de objetos

La visión por computador ha experimentado un gran avance en los últimos años, impulsado principalmente gracias al desarrollo de algoritmos cada vez más precisos y eficientes. Se basa en la capacidad por parte de las máquinas de interpretar el contenido visual del mundo, con el procesamiento de imágenes y vídeos.

En la visión por computador, existe un conjunto de tareas fundamentales para poder analizar el contenido visual. Este conjunto de tareas se podría representar en clasificación de imágenes, detección de objetos y segmentación.

La clasificación de imágenes se trata de la tarea más básica y se basa en determinar a qué clase contiene una imagen dada. En la figura 2.1 podemos ver un ejemplo de cómo funciona.

La detección de objetos se entiende como la capacidad de reconocer en qué zona de una imagen se encuentra un determinado objeto y, a su vez, saber distinguir correctamente la clase a la que pertenece el mismo. Es decir, nos permite reconocer los objetos que hay en una imagen y dónde se encuentran. En la figura 2.2 vemos un ejemplo de detección.

La tarea de segmentación se trata de la más compleja. Esta tarea, además de permitir la detección de los objetos, permite diferenciar entre las distintas partes de estos a nivel de píxel. Un ejemplo de segmentación se puede ver en la figura 2.3.


**Figura 2.1:** Ejemplos de clasificación de imágenes, con clase predicha, real y confianza

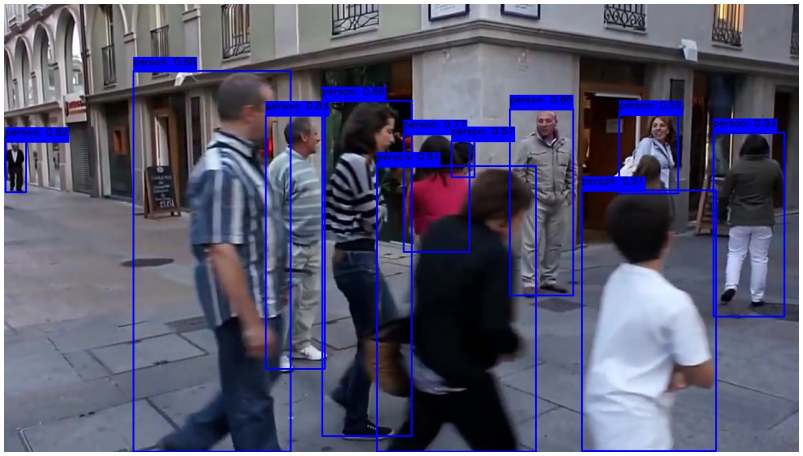

**Figura 2.2:** Ejemplo de detección de objetos

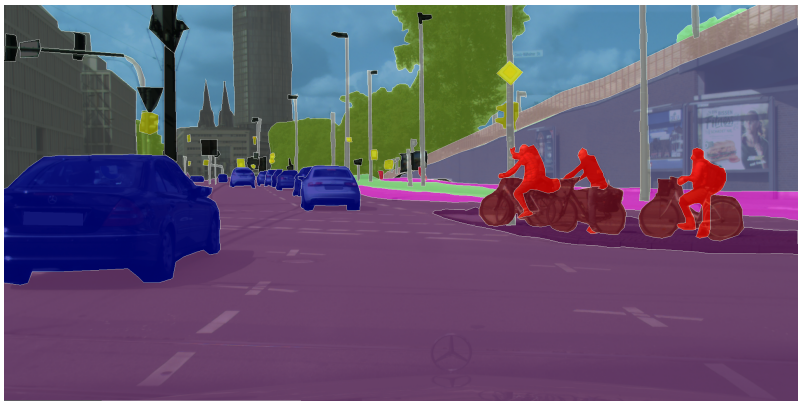

**Figura 2.3:** Ejemplo de segmentación

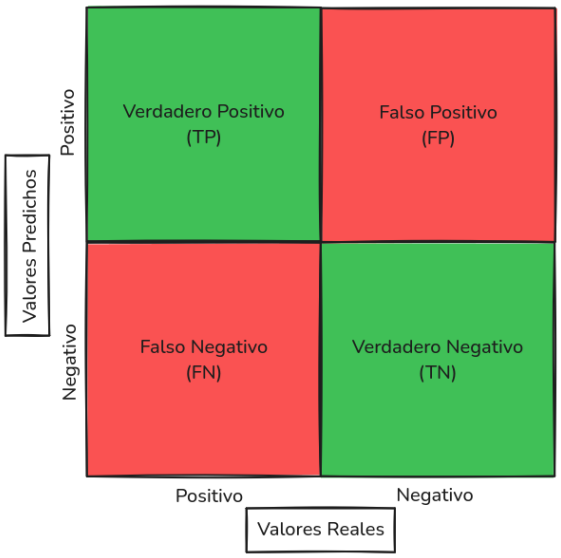


**Figura 2.4:** Matriz de confusión

### 2.1.1. Métricas

Las métricas de rendimiento nos permiten evaluar qué tan bien lo hace nuestro modelo. En relación a [29], para hablar de las métricas de evaluación en la detección de objetos, tenemos que hablar de IoU o *Intersection Over Union*, que representa el porcentaje de superposición entre la predicción del modelo y la etiqueta real, respecto a la unión de ambas.

A partir del IoU, al establecer un umbral, se pueden clasificar las predicciones como correctas o incorrectas. Además, se identifican los casos en los que el modelo no logra detectar un objeto que debería haber sido identificado. Esta clasificación da lugar a lo que se conoce como matriz de confusión, en la figura 2.4.

Con los términos de verdadero positivo, falso positivo, falso negativo y verdadero negativo, es posible definir las distintas métricas de rendimiento. Entre ellas destacan la precisión, el *recall* y el mAP.

#### Precisión

La precisión es una métrica que cuantifica el número de predicciones que son correctas respecto al total de predicciones. Indica cuán acertadas son las predicciones positivas emitidas:

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

#### Recall

El *recall* nos permite saber, sobre todas las instancias a detectar, la proporción de etiquetas correctas que se han llegado a detectar. Indica qué tan bien el modelo está detectando las instancias positivas que debía encontrar:

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

#### mAP (mean Average Precision)

Para entender el mAP o *mean Average Precision*, primero se debe entender el AP o *Average Precision*. El AP se trata de la medida que permite calificar la calidad del modelo para una determinada clase. Este se define como el área bajo la curva Precisión-Recall, la cual refleja cómo varía la precisión del modelo respecto al *recall* a diferentes umbrales de confianza:

$$\text{AP} = \int_0^1 \text{Precision}(r)\, dr$$

A partir de esta medida, se define el mAP como el promedio de los valores de AP calculados para todas las clases presentes en el conjunto de datos:

$$\text{mAP} = \frac{1}{K} \sum_{k=1}^{K} \text{AP}_k$$

Además, es habitual refinar este cálculo evaluando el desempeño del modelo en distintos umbrales del *Intersection over Union* (IoU), lo que permite obtener una evaluación más precisa sobre la calidad en la localización de las detecciones. En este caso, el mAP se calcula como el promedio de los valores de AP por clase en cada umbral:

$$\text{mAP} = \frac{1}{K} \sum_{k=1}^{K} \left( \frac{1}{T} \sum_{t=1}^{T} \text{AP}_{k,t} \right)$$

donde:

- $K$ es el número total de clases.
- $T$ es el número de umbrales de IoU considerados.
- $\text{AP}_{k,t}$ es el *Average Precision* para la clase $k$ y el umbral de IoU $t$.

### 2.1.2. Conjuntos de datos

Los modelos actuales de detección de objetos necesitan una gran cantidad de imágenes etiquetadas para lograr los resultados deseados. Por ello, surgen algunos conjuntos de datos, o *datasets*, ampliamente utilizados con este propósito.

#### MS COCO

Microsoft Common Objects in Context [59] es un conjunto de imágenes diseñado para la detección y segmentación. En su presentación, las imágenes son descritas tal que: "podrían ser fácilmente identificables por un niño de cuatro años", refiriéndose a objetos y escenas comunes y fácilmente identificables.

Este conjunto de imágenes incluye aproximadamente 330.000 imágenes, teniendo más de 200.000 anotadas. COCO define un total de 80 categorías de objetos. La partición estándar del conjunto se divide en:

- **Entrenamiento**: 118.000 imágenes.
- **Validación**: 5.000 imágenes.
- **Test**: 41.000 imágenes.

#### Pascal VOC

Pascal Visual Object Classes Challenge (VOC) [58] se utilizó ampliamente como referencia entre 2005 y 2012, de forma que se favoreciese el desarrollo de modelos de detección de objetos. Este conjunto sigue tratándose como una referencia clásica en la evaluación de los modelos de detección.

La versión más empleada es VOC 2007, que contiene un total de 9.963 imágenes, con aproximadamente 24.640 instancias de objetos. VOC define 20 clases de objetos, centradas en categorías comunes como personas, vehículos y animales. La partición del conjunto se divide en:

- **Entrenamiento**: 2.501 imágenes.
- **Validación**: 2.510 imágenes.
- **Test**: 4.952 imágenes.

#### Open Images

Open Images [61] se trata de un conjunto de imágenes muy amplio. Fue desarrollado por Google y ofrece anotaciones de alta calidad.

El conjunto contiene más de nueve millones de imágenes, de las cuales más de 1,9 millones están anotadas con cajas delimitadoras. En este conjunto se definen 600 clases de objetos. La partición principal del conjunto es:

- **Entrenamiento**: 1.743.042 imágenes.
- **Validación**: 41.620 imágenes.
- **Test**: 125.436 imágenes.

### 2.1.3. Redes neuronales convolucionales

Los primeros modelos de detección estaban basados en la extracción de características, como puede ser el modelo de Viola-Jones [1], el detector de histogramas de gradientes orientados (HOG)[2], o una extensión de este último, que se centra en la detección individual de las partes de un objeto para lograr así la detección completa de la instancia [3].

Estos modelos no alcanzaron gran interés debido a que no eran buenos con datos fuera del conjunto de entrenamiento. Sin embargo, la introducción de las redes neuronales convolucionales significó un cambio en este contexto.

Las redes neuronales convolucionales o CNNs se tratan de un tipo de red neuronal que, por medio de la aplicación de filtros convolucionales en la imagen de entrada, son capaces de detectar patrones y detalles en la imagen, generando así mapas de características.

Las bases para las primeras CNN fueron establecidas por Neocognitron en 1980 [4], pero no fue hasta 1998, con la aparición de LeNet [5] que se desarrolló la primera CNN funcional. La eficacia de estas redes se consolidó en 2012 con el desafío de ImageNet[62], que fue ganado por AlexNet [6], una red basada en CNNs.

Las CNN se caracterizan por estar formadas por tres tipos de capas:

- Capa convolucional: se basa en la aplicación de un conjunto de filtros que se desplazan sobre la imagen capturando un conjunto de patrones. Esto da lugar a la generación de un mapa de características.

- Capa de agrupación: nos permite reducir las dimensiones de los mapas de características, aportando invarianza a traslaciones. A pesar de la reducción, mantiene información de la posición en que aparecen las características de entrada. Algunos ejemplos comunes son *Global Average Pooling*, *Max Pooling* o *Average Pooling*.

- Capas *fully connected* o totalmente conectadas: serán las capas encargadas de interpretar las características obtenidas por las capas anteriores para realizar la clasificación final.

Según Sun *et al.*[7], los tipos de arquitecturas CNN para detección de objetos pueden clasificarse en dos grandes categorías: los modelos basados en regiones candidatas (o modelos en dos fases) y los modelos de clasificación/regresión directa (o modelos de una fase).

### Modelos en dos fases

Los modelos en dos fases comienzan generando propuestas de ubicación (o regiones candidatas), esto es, regiones que probablemente contengan un objeto. A partir de esta propuesta, se utiliza una capa de agrupación ROI, la cual permite convertir regiones de distintos tamaños en una de tamaño uniforme. Esta representación se emplea posteriormente para predecir la clase del objeto presente en cada región.

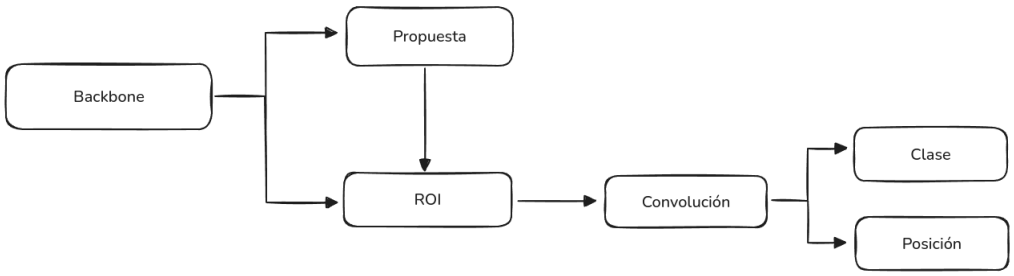


**Figura 2.5:** Estructura del modelo en dos fases

En este tipo de modelos entran:

- **R-CNN**: en 2014, Girshick *et al.*[33] propusieron un nuevo método para la detección de objetos, el cual consiguió mejorar los resultados de mAP en PASCAL VOC12 en más de un 30 %.

  El detector está compuesto por tres módulos. El primer módulo se basa en un generador de regiones candidatas que permite generar 2000 regiones candidatas mediante *Selective Search* [63]. Seguidamente estas regiones son pasadas a un segundo módulo. Cada región candidata se redimensiona a 227×227 píxeles y se introduce en una CNN que extrae sus características. Finalmente, el tercer módulo es un conjunto de SVMs lineales [34], uno por cada clase.

  Este modelo ofrece algunos problemas, entre estos encontramos el redimensionamiento de las imágenes a 227x227 píxeles, lo que puede conllevar largos tiempos de preprocesado en ciertas ocasiones. Además, la generación de regiones candidatas genera regiones solapadas o redundantes, lo que conlleva falta de eficiencia.

- **SPP-Net**: en 2015, He *et al.* propusieron SPP-Net[38]. Este surge ante el problema de las CNN que requieren un tamaño de entrada fijo. Esto puede conllevar generar distorsiones en la imagen, afectando el rendimiento del modelo.

  SPP-Net resuelve el problema al ser capaz de generar una salida uniforme, independientemente del tamaño de imagen de entrada. Con este modelo, solo se necesita la extracción del mapa de características una única vez, acabando con la idea de generación para cada región candidata. Esto es así gracias a la combinación de *Spatial Pyramid Matching* o SPM[39] junto a redes neuronales convolucionales.

  Este modelo mejoró el mAP en Pascal VOC 2007 a 59,2 % respecto al 58,2 % del modelo R-CNN, además de reducir significativamente los tiempos de detección en GPU.

- **Fast R-CNN**: en 2015, Girshick [35] propuso Fast R-CNN, una versión más rápida y eficiente que R-CNN. Este sistema acababa con el entrenamiento separado de cada uno de los módulos de R-CNN. Fast R-CNN unifica el proceso en una única red entrenable de principio a fin.

  Primero se extraen las características de la imagen mediante una CNN. A continuación, los vectores de características junto a las regiones detectadas son enviados a una capa de agrupación ROI. El uso de esta capa permite la extracción de regiones de interés de tamaño variable. Para mejorar la eficiencia, se usa *Singular Value Decomposition* o SVD[36] en las capas *fully connected* que reciben la salida de la ROI, de esta forma se reduce la carga computacional sin una pérdida significativa de precisión. Finalmente, se hace uso de una *softmax* para la clasificación de las regiones.

  Este modelo consiguió un mAP del 70 % en PASCAL VOC 2007 frente al 58,5 % de R-CNN. A su vez, la velocidad de las detecciones pasó de ser de 47 segundos en R-CNN frente a los 0,32 segundos en Fast R-CNN.

- **Faster R-CNN**: en 2016, Ren *et al.* introdujeron Faster R-CNN[37]. Este acababa con la necesidad de ventanas deslizantes para la generación de regiones candidatas. En su lugar, aplicaba *Region Proposal Network* o RPN que se trataba de una CNN diseñada para generar estas regiones, reduciendo prácticamente a cero el coste y mejorando velocidad y precisión.

  Esta red permitía realizar detecciones a cinco fps (fotogramas por segundo) en GPU, reduciendo los tiempos a 198 ms por imagen, respecto a los 320 ms de Fast R-CNN. Por último, en términos de precisión, obtuvo un mAP del 73,2 % en VOC 2007.

- **Feature Pyramid Network(FPN)**: en 2017, Lin *et al.* propusieron *Feature Pyramid Network* o FPN[57], una arquitectura diseñada para mejorar las detecciones ante objetos en escala pequeña, así como mejorar la representación ante características multiescala.

  Los mapas de características de bajo nivel ofrecen menor información semántica pero con mayor resolución, los de alto nivel al contrario.

  FPN introduce una arquitectura en forma de pirámide con dos rutas. La ascendente se basa en una estructura convolucional, se centra en la generación de las representaciones. Mientras que la descendente, mediante conexiones laterales, se encarga de distribuir y fusionar las características extraídas en los distintos niveles de resolución.

  Gracias a esto, las representaciones de los mapas de características presentan una alta semántica y una alta resolución, mejorando la detección en distintas escalas.

  Su integración con Faster R-CNN permitió reducir los tiempos de detección a 165 ms, a la vez que incrementar el mAP.

### Modelos en una fase

Este tipo de modelos, al contrario de los anteriores, no necesita de regiones candidatas. Estos modelos son capaces de identificar la posición y clasificar al objeto en un único paso.

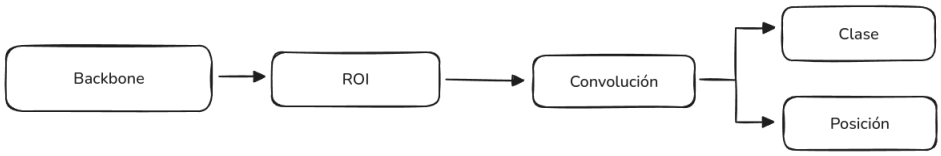


**Figura 2.6:** Estructura del modelo en una fase

Como ejemplos de modelos en esta clase, encontramos:

- **YOLO**: en 2016, Redmon *et al.* propusieron YOLO o *You Only Look Once* [56], para lograr obtener resultados que, a diferencia de los modelos como R-CNN que obtenía resultados con una mayor precisión, fuesen capaces de realizar detecciones rápidas. Esta serie de modelos permitía el uso en aplicaciones en tiempo real.

  Esta red hace todas las detecciones de la imagen al mismo tiempo. Además, para realizar la predicción de cada caja delimitadora o *bounding box* usa las características de toda la imagen. El modelo divide la imagen en una cuadrícula SxS, y cada una de las celdas es responsable de detectar un objeto si el centro del objeto cae en la celda. Por cada celda se predicen B cajas delimitadoras, acompañadas de una puntuación de confianza que indica la probabilidad de presencia de un objeto y la precisión de la caja predicha.

  La arquitectura del modelo se basa en GoogleNet[64], pero sustituye los módulos *inception* por capas de reducción 1x1 seguidas por capas convolucionales 3x3. La red en sí está formada por 24 capas convolucionales seguidas de dos capas *fully connected*. La red entrena en tamaños de 224x224 en las tareas de clasificación y de 448x448 en tareas de detección.

  Este modelo tiene problemas cuando aparecen objetos cercanos o objetos pequeños, mayormente en grupo. También encuentra problemas al tratar de entrenar con objetos no vistos anteriormente o escalas de objetos poco comunes.

  En 2016, Redmon *et al.* propusieron una segunda versión de YOLO, YOLO9000 o YOLOv2[55]. Este modelo buscaba mejorar los resultados de *recall*, reduciendo el error pero manteniendo la precisión que obtenía YOLO. Sin embargo, no se podía perder la velocidad que ofrecía su predecesor. La inclusión de ciertas técnicas permitió esto:

  - *Batch Normalization* [14] que acababa con la necesidad de otras técnicas de regularización ayudando a la convergencia.
  - El incremento de la resolución para clasificación, pasando a ser de 448x448.
  - La sustitución de las capas *fully connected* por cajas ancla o *anchor boxes*. Estas eran cajas predefinidas que ayudaban en la detección de los objetos, y así se facilitaba la predicción de las cajas delimitadoras, que aunque reducía la precisión, aumentaba el *recall* y permitía incrementar el número de detecciones.
  - El uso de *k-means clustering* [60] permitía determinar automáticamente el tamaño de las cajas ancla.
  - Se permitía el entrenamiento de la red en varios tamaños de imagen, de forma que fuese capaz de aprender a predecir en una variedad de tamaños desde 320 hasta 608, en saltos de 32.

En 2018, Redmon *et al.* introdujeron la tercera versión de YOLO, YOLOv3[54], una evolución de YOLOv2 que combina su arquitectura con un nuevo *backbone* llamado Darknet-53. Este modelo cuenta con 53 capas convolucionales que mejoran la detección de objetos en diferentes escalas utilizando detecciones en tres resoluciones. Los resultados son buenos usando el AP50, pero su rendimiento se ve afectado al ser evaluado con métricas más estrictas como AP50-95.

En 2020, Bochkovskiy *et al.* introdujeron el siguiente modelo YOLO, YOLOv4[49]. YOLOv4 cuenta con CSPDarknet53 como *backbone*, junto a SPP[47] como módulo adicional, PANet[53] como el cuello para agregar las características de distintas capas mejorando la detección de objetos pequeños y grandes, y YOLOv3 como la cabeza, beneficiándose del mecanismo de cajas ancla.

YOLOv4 mejora su rendimiento a partir de dos tipos de técnicas. *Bag of Freebies*, que son técnicas que mejoran la precisión sin incrementar la velocidad de inferencia. Por ejemplo, el uso de *Mixup* o la introducción de *Mosaic* como técnicas de aumento de dato. También el uso de *Bag of Specials*, los cuales son módulos que aportan mejoras significativas a costa de incrementar en cierta medida los tiempos de cómputo. Entre estos últimos módulos, entraría la unión del módulo SPP.

En 2020, el despliegue de YOLOv5 cambió el uso de Darknet por Pytorch, de forma que era más fácil su entrenamiento y despliegue, lo que incrementó su popularidad. Este modelo se basa en el modelo YOLOv4. Fue desarrollado por la comunidad y liderado por Ultralytics. Este modelo se consolidó como uno de los más prácticos debido a su facilidad de uso y la integración real del modelo.

Durante los años siguientes salieron algunos modelos YOLO, como YOLOX propuesto por Ge *et al.* en 2021 [15], YOLOv6 propuesto por Li *et al.* en 2022 [52] o YOLOv7 propuesto por Wang *et al.* también en 2022 [45]. Aunque no fueron desarrolladas por Ultralytics, introdujeron mejoras notables en velocidad y precisión.

Jocher *et al.* introdujeron en 2023 YOLOv8 [48], una significativa evolución en la serie de modelos YOLO. Si bien conserva conceptos de YOLOv5, introduce varias mejoras arquitectónicas y metodológicas. Fue entrenado con una mayor cantidad de datos y un número más amplio de clases, lo que aumentó su capacidad de generalización. En su arquitectura, la red hacía uso de FPN[57] que permitía generar mapas de características a distintas resoluciones, que junto al uso de PANet permitía reducir el problema de las detecciones a múltiples escalas.

Una de las innovaciones más destacadas de YOLOv8 fue la eliminación del uso de cajas ancla. A diferencia de versiones anteriores, que requerían múltiples salidas para estimar el tamaño de las cajas y su desplazamiento, YOLOv8 adopta un enfoque sin estas: utiliza una única salida para predecir directamente la ubicación del centro del objeto, así como su anchura y altura. Esto simplifica el proceso de predicción, reduce el número de cajas generadas durante la inferencia y acelera significativamente los posteriores procesados.

A pesar de la aparición de nuevos modelos YOLO como YOLOv9 desarrollado por Wang *et al.* [46] o YOLOv10 desarrollado por Li *et al.* [51], estos modelos no fueron desarrollados por Ultralytics.

El último de los modelos se trata de YOLOv11, desarrollado por Jocher *et al.* en 2024 [50]. El modelo es capaz de interpretar compleja información visual, gracias a la implementación de nuevos elementos como el bloque C3k2, SPPF y C2PSA. Este último incorpora mecanismos de atención que mejoran la detección en escenarios complejos al centrar la atención en regiones clave de la imagen.

- **SSD**: en 2016, Liu *et al.* publicaron *Single Short Detection* o SSD [40], diseñado para abordar el problema de la detección de objetos a múltiples escalas, una de las principales limitaciones de los primeros modelos como YOLO.

  Este modelo se basa en la estructura de VGG16 [41] que funciona como extractor de características, pero se añaden capas de convoluciones para poder realizar detecciones a distintas escalas. Finalmente, se aplica el algoritmo de *Non-Maximum Suppression* (NMS) [12] para eliminar cajas redundantes que se solapan significativamente, obteniendo así una salida más precisa y limpia.

  SSD logró alcanzar un mAP de 76,8 % en Pascal VOC 2007, además de incrementar la velocidad de detección a 22 fps, lo que lo convirtió en una solución funcional y eficiente en tareas en tiempo real.

- **RetinaNet**: en 2017, Lin *et al.* propusieron RetinaNet[42], el cual incluía pérdida focal y arquitectura FPN [57], tratando de mejorar el rendimiento de los detectores en una sola fase. Esta red buscaba reducir el desequilibro de clases en los modelos de una fase.

  La pérdida focal se introducía para reducir el peso que tenían los ejemplos sencillos durante el entrenamiento, permitiendo enfocar el aprendizaje en aquellos más complejos y mejorando la precisión. El uso de la arquitectura FPN mejora la detección de pequeños objetos, al poder conservar la resolución y la información semántica a un buen nivel.

  Las características generadas por la red se dividen en dos subredes, una para la clasificación y otra para obtener las cajas delimitadoras. Esta separación, junto con el uso de FPN y la pérdida focal, permite lograr una detección más precisa sin comprometer la eficiencia computacional.

### 2.1.4. Modelos basados en Transformers

Según [7] y [8], las arquitecturas basadas en Transformers han cobrado gran importancia en el campo de la visión por computador, gracias al mecanismo conocido como auto-atención que permite establecer dependencias entre datos distantes en una secuencia. Esto representa una ventaja significativa frente a arquitecturas tradicionales como las RNN [9] o las redes convolucionales, ya que los Transformers no requieren procesamiento secuencial ni operaciones de convolución.

El mecanismo de auto-atención se basa en una estructura de un codificador junto a un decodificador. La parte codificadora se encarga de procesar la entrada generando una representación de esta, mientras que el resultado de este tratamiento es enviado a los decodificadores para producir la salida final.

Su popularidad creció inicialmente en entornos relacionados con el lenguaje humano, y no tomó una mayor popularidad en el ámbito de la visión hasta 2020 con la aparición de DETR [10] y *Vision Transformer* o ViT [11]. A su vez, estos dos modelos marcan una clara diferencia en el uso de modelos basados en Transformers en la visión por computador.

#### DETR

En 2020, Carion *et al.* propusieron DETR[10] como una nueva forma de abordar la detección de objetos, reemplazando completamente los componentes tradicionales de detección (como la necesidad de regiones candidatas y el *Non-Maximum Suppression*) con una arquitectura basada en transformers. Las arquitecturas clásicas de detección, como Faster R-CNN o RetinaNet, dependen de etapas manuales y estructuras complejas como

pirámides de características y procedimientos para el filtrado de resultados. DETR nace como una solución a estos enfoques altamente diseñados, proponiendo un modelo de detección más simple, capaz de encargarse de todo el proceso y que aprende directamente a predecir un conjunto de objetos.

DETR combina una red convolucional backbone (como ResNet) con un transformer codificador-decodificador. La CNN extrae mapas de características de la imagen, que luego son procesados por el codificador, convirtiéndolos en un conjunto de secuencias al que se le añade la información de la posición. El decodificador recibe un conjunto fijo de consultas de objeto que, a través de mecanismos de atención, aprenden a enfocarse en diferentes regiones de la imagen para predecir directamente las clases y las cajas delimitadoras. El número de salidas es fijo, por lo que si el número de objetos no llega a este número de salidas, se hace uso de la clase *no object*.

Aunque DETR mostró resultados competitivos en *datasets* como COCO, su principal limitación fue la lenta convergencia, necesitando más de 500 épocas de entrenamiento para alcanzar un rendimiento comparable a métodos clásicos.

#### ViT

En 2020, Dosovitskiy *et al.* propusieron ViT o *Vision Transformer* [11], modelo que introdujo por primera vez el uso completo de arquitecturas tipo transformer en tareas de visión por computador, es decir, sin recurrir a convoluciones. Surgió como una alternativa al uso de redes convolucionales, que hasta entonces dominaban el campo, mostrando que era posible prescindir completamente de convoluciones y obtener resultados competitivos usando únicamente mecanismos de auto-atención. El diseño se inspira en los transformers del procesamiento de lenguaje natural y parte de la hipótesis de que, con suficiente cantidad de datos y capacidad de cómputo, los transformers también pueden aprender representaciones visuales efectivas.

La arquitectura de ViT divide la imagen en ventanas de tamaños fijos (por ejemplo, de 16×16 píxeles), que se aplanan y se proyectan linealmente en vectores. A estos vectores se les añade la información de la posición y se procesan como una secuencia de *tokens* a través de un transformer. Además, se añade un *token* especial de clasificación cuya salida se utiliza para predecir la clase de la imagen.

ViT demostró que, cuando se entrena con grandes conjuntos de datos, puede superar a los modelos CNN como ResNet en precisión. Sin embargo, para funcionar bien, requiere grandes volúmenes de datos y potencia de cómputo.

## 2.2 Aumentado de datos

Las redes neuronales son modelos que necesitan una gran cantidad de datos para funcionar adecuadamente, es decir, contra más hayan visto en su fase de entrenamiento, mejor tienden a funcionar. El problema reside en lograr tener esta gran cantidad de datos correctamente etiquetados. El proceso de etiquetado acaba siendo manual en caso de querer asegurarnos su correcto funcionamiento. Además, se debe tener en cuenta la elevada cantidad de recursos necesaria para poder mantener todos estos datos.

En [13], se trata el aumento de datos como una técnica que busca añadir robustez ante conjuntos de datos escasos, de forma que sin necesidad de añadir nuevas imágenes se logren buenos resultados. Esta técnica se trata de una técnica de regularización, las cuales tratan de reducir el problema del sobre-ajuste (el modelo tiende a no funcionar bien con datos nuevos y no vistos), es decir, busca mejorar la capacidad de generalizar

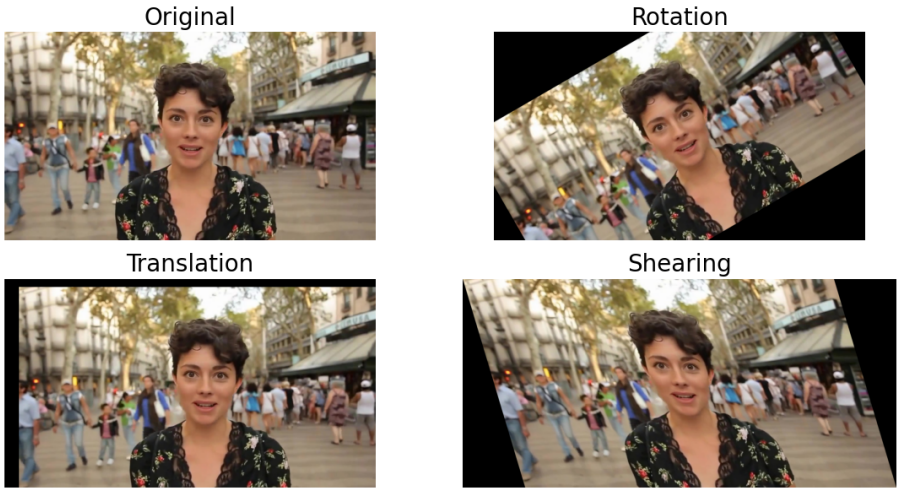


**Figura 2.7:** Ejemplos de transformaciones geométricas

por parte del modelo. Otras técnicas de regularización serían la normalización *batch* [14] o el DropOut [16].

### 2.2.1. Algunas transformaciones

Según [17], podemos establecer una división entre los distintos tipos de transformaciones básicas a la imagen. Primeramente, tenemos aquellas orientadas a la manipulación de las imágenes (geométrica o no geométrica), mientras que la segunda clase de transformaciones está destinadas a borrar ciertas zonas de la imagen o sustituirlas. Además, hay transformaciones orientadas a combinar varias imágenes en una sola.

#### Manipulaciones geométricas

Como su propio nombre indica, este tipo de transformaciones se basa en cambios geométricos en la imagen, como podría ser la orientación o la posición. En esta clase entran la rotación de la imagen (*Rotation*), la traslación (*Translation*) o el cambio de perspectiva (*Shearing*). En la figura 2.7 se pueden ver ejemplos de estas.

#### Manipulaciones no geométricas

Este tipo de transformaciones se basa en la modificación de las características visuales. En esta clase entran transformaciones como dar la vuelta a la imagen (*Flipping*), el recortado para centrarse en ciertas zonas (*Cropping*), la introducción de ruido (*Noise injection*), la manipulación de los canales de color (*Color Space*), cambios de brillo, contraste o saturación (*Jitter*) y, por último, aplicación de desenfoque, entre un conjunto de técnicas que entran en este grupo (*Kernel*). En la figura 2.8 se pueden ver estas transformaciones aplicadas.

#### Transformaciones de borrado

Estas se basan en la eliminación de ciertas zonas de la imagen, sustituyéndolas por colores base u otros patrones. En esta clase entran el rellenado de ciertas zonas de la imagen con colores base y tamaño fijo (*Cutout*), la eliminación de zonas con colores aleatorios o con ruido de tamaños variables (*Random erasing*), la división de la imagen en un conjunto


**Figura 2.8:** Ejemplos de transformaciones no geométricas


**Figura 2.9:** Ejemplos de aplicación de transformaciones de borrado

de cuadrados tapando una cantidad aleatoria (*Hide and seek*) o, por último, la aplicación de un patrón de rejilla ocultando regiones de la imagen (*GridMask*). La aplicación de estas transformaciones se puede ver en la figura 2.9.

#### Transformaciones combinatorias

Estas transformaciones se caracterizan por mezclar dos o más imágenes en una misma, creando nuevas imágenes que ayudan a generalizar el modelo. En estas destacan algunas, como la combinación de dos imágenes mediante una interpolación lineal (*Mixup*) o la unión de cuatro imágenes en una misma cuadrícula (*Mosaic*) introducida en YOLOv4[49]. Se pueden ver ejemplos de estas en la figura 2.10.

### 2.2.2. Técnicas de aumento automático de imágenes

Según [17], existen diversas técnicas automáticas de aumento de datos que han demostrado ser altamente efectivas en tareas de visión por computador. Entre ellas, destacan AutoAugment, RandAugment y AugMix, consideradas realmente importantes por


**Figura 2.10:** Ejemplos de transformaciones combinatorias

su impacto positivo en el rendimiento de los modelos y su capacidad de adaptación a diferentes tipos de datos.

### AutoAugment

AutoAugment, propuesto por Cubuk *et al.* [19], es una técnica de aplicación automática de aumento de datos que nace para eliminar la necesidad de la aplicación manual de estas transformaciones. La técnica está formada por un algoritmo de búsqueda y un espacio de búsqueda.

Esta técnica explora un espacio de búsqueda formado por políticas que a su vez están formadas por subpolíticas, las cuales son aplicadas a cada imagen. Una subpolítica se trata de dos técnicas individuales de aumento de datos, unidas a una magnitud para su aplicación, junto a la probabilidad de aplicarla. El objetivo será encontrar, concurrentemente, cinco subpolíticas para cada política para poder incrementar la flexibilidad.

En este caso, AutoAugment puede aplicar 16 transformaciones distintas. Además, la magnitud a aplicar puede tomar diez valores distintos y la probabilidad de aplicación, once. Por ello, al buscar dos transformaciones por cada vez que vaya a aplicar, el espacio de búsqueda se vuelve del estilo de $(16 \times 10 \times 11)^2$. Como el objetivo de la técnica es encontrar cinco subpolíticas para aumentar la diversidad, el espacio de búsqueda será de $(16 \times 10 \times 11)^{10}$ o del orden de $10^{32}$.

El algoritmo de búsqueda tiene dos componentes: un controlador, el cual es una RNN que genera políticas de aumento, y un algoritmo para el entrenamiento del controlador. Debido al espacio de búsqueda, la RNN tiene 30 *softmax* para predecir la decisión en cada caso. Esto es así ya que tiene que encontrar cinco subpolíticas y dos operaciones, siendo cada operación la transformación, la magnitud y la probabilidad de aplicación.

El entrenamiento de la RNN se basa en el desempeño de un modelo auxiliar, siguiendo aprendizaje por refuerzo [31]. El controlador genera cinco subpolíticas y este modelo auxiliar es entrenado con ellas. Cuando acaba el entrenamiento, se evalúa sobre el conjunto de validación, y el resultado se aplica para actualizar el controlador. Se puede ver el ciclo del entrenamiento en la figura 2.11. Al final del proceso, se seleccionan las cinco mejores políticas y se combinan en un conjunto de 25 subpolíticas que se utilizarán durante el entrenamiento final del modelo.

### RandAugment

RandAugment, también propuesto por Cubuk *et al.* [18], es una técnica que rompe con la dinámica de los métodos automáticos de aumento de datos. Esta clase de técnicas encontraba resultados SOTA (*State of the Art*), pero a costa de un alto coste computacional

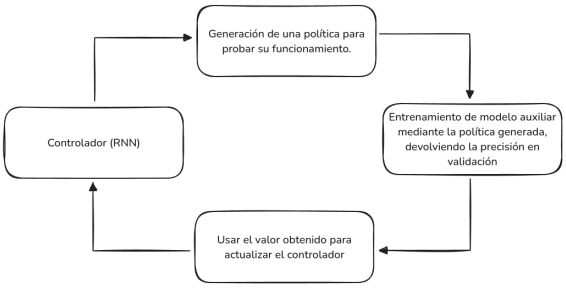


**Figura 2.11:** Proceso de búsqueda seguido por AutoAugment

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugment(N, M):
"""Generate a set of distortions.

   Args:
    N: Number of augmentation transformations to
        apply sequentially.
    M: Magnitude for all the transformations.
"""

  sampled_ops = np.random.choice(transforms, N)
  return [(op, M) for op in sampled_ops]
```

**Figura 2.12:** Aplicación de RandAugment en Python. Extraída de [18]

durante el proceso de búsqueda de aquellas técnicas de aumento de datos que mejor funcionen en el modelo. Para evitar tiempos demasiado altos, estas técnicas se ven obligadas a usar un conjunto menor de datos para realizar la búsqueda, lo que puede acabar en aplicaciones no óptimas en el conjunto de datos real.

RandAugment rompe con la necesidad de usar un conjunto de datos reducido para la búsqueda. Además, significa una drástica reducción en el espacio de búsqueda de los parámetros respecto al resto de técnicas.

En este caso, como se puede ver en la figura 2.12 únicamente necesita dos parámetros, pudiendo aplicar una combinación de K=14 transformaciones distintas. El parámetro N indica el número de transformaciones a aplicar, mientras que el parámetro M es la magnitud la cual se les aplicará. Esto significa que se pueden aplicar un total de $K^N$ políticas de aumento posibles. En el caso del parámetro M, se comprobó que no era necesario aplicar distintas magnitudes a cada transformación, sino que con un único valor se encontraban buenos resultados sin necesidad de añadir complejidad al proceso.

Comparando esta técnica respecto a AutoAugment en el conjunto de datos de COCO [30] sobre modelos ResNet, podemos ver los resultados de mAP y espacio de búsqueda en la tabla 2.1. Aunque los resultados obtenidos con RandAugment son ligeramente inferiores a los de AutoAugment en términos de mAP, el espacio de búsqueda requerido es considerablemente inferior, lo que representa una ventaja significativa en términos de eficiencia computacional.

### AugMix

AugMix, presentado por Hendrycks *et al.* [20], es una técnica que surge no solo para mejorar la precisión del modelo, sino también su robustez ante variaciones en los datos

| Model | Augmentation | mAP | Search Space |
|---|---|---|---|
| ResNet-101 | Baseline | 38,8 | 0 |
| | AutoAugment | 40,4 | $10^{32}$ |
| | RandAugment | 40,1 | $10^2$ |
| ResNet-200 | Baseline | 39,9 | 0 |
| | AutoAugment | 42,1 | $10^{32}$ |
| | RandAugment | 41,9 | $10^2$ |

**Tabla 2.1:** Comparación de técnicas de aumento sobre modelos ResNet y su espacio de búsqueda

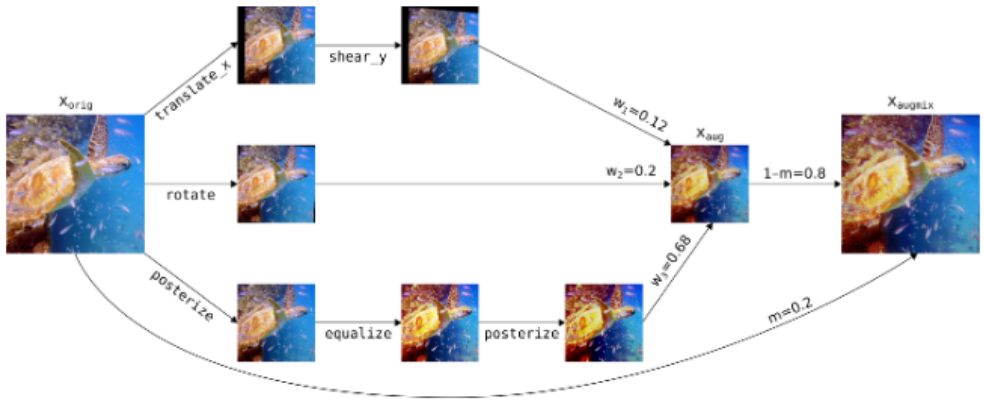


**Figura 2.13:** Ejemplo de proceso de generación de imágenes por AugMix. Extraído de [20]

de entrada. En particular, es importante cuando existen discrepancias entre las distribuciones del conjunto de imágenes en entrenamiento y durante la inferencia.

Esta técnica se diferencia del resto de técnicas porque combina varias secuencias de transformaciones en la imagen original. El proceso comienza generando varias versiones de una imagen mediante secuencias aleatorias de transformaciones seleccionadas de un conjunto de estas. Seguidamente, las versiones generadas se combinan en distinta proporción con la imagen original, lo que genera una única versión. En la figura 2.13 se puede ver este proceso. Para cada imagen, AugMix genera dos más que mantienen la estructura semántica original.

Aunque la aplicación de esta técnica suponga tiempos más altos de ejecución, los modelos son capaces de generalizar mejor y son más estables ante distintas entradas.

# CAPÍTULO 3
# Entorno experimental

Este capítulo tiene como objetivo definir y validar el entorno experimental sobre el cual se llevará a cabo el desarrollo y la evaluación de la técnica de aumento de datos propuesta. Para garantizar la fiabilidad y reproducibilidad de los experimentos, se detalla la máquina utilizada, el conjunto de datos seleccionado y el modelo base empleado. Asimismo, se describe el proceso de comprobación de dicho entorno mediante una primera ejecución de prueba, que permitió encontrar diversas incidencias. Finalmente, se describe la forma en la que estas incidencias fueron resueltas, de forma que los experimentos sobre la técnica propuesta fuesen correctos.

## 3.1 Configuración inicial

Comenzando por la máquina de trabajo, se eligió la Nvidia Jetson Orin Nano 8GB. Esta máquina está destinada a aplicaciones de robótica y permite la ejecución de modelos de aprendizaje profundo con un bajo coste computacional y económico. Por esto mismo, se trató de usar esta máquina tanto para el entrenamiento como para la inferencia. En la tabla 3.1 se pueden ver las especificaciones de la máquina.

| **Especificación** | **Detalles** |
|---|---|
| **Rendimiento de IA** | 67 TOPS |
| **GPU** | NVIDIA Ampere con 1024 CUDA cores y 32 Tensor cores |
| **Frecuencia máxima GPU** | 1020 MHz |
| **CPU** | 6 núcleos Arm Cortex-A78AE v8.2 64-bit<br>1,5MB L2 + 4MB L3 |
| **Frecuencia máxima CPU** | 1,7 GHz |
| **Memoria** | 8GB LPDDR5 de 128 bits (68 GB/s) |
| **Almacenamiento** | Soporte para tarjeta SD y NVMe externo |
| **Consumo de energía** | 7W–15W-25W |

**Tabla 3.1:** Especificaciones técnicas del módulo Jetson Orin Nano 8GB. Extraídas de [32]

Una vez escogida la máquina, se buscaron distintos *datasets* que permitiesen alcanzar el objetivo de la detección de personas. Tras la investigación se encontraron tres *datasets* que permitían cumplir con los objetivos del proyecto:

**Figura 3.1:** Imagen perteneciente al conjunto de datos PersonPath22

- CrowdHuman [21]: este conjunto de datos cuenta con 15.000 imágenes de entrenamiento, 4.370 de validación y 5.000 de test. Cuenta con 339.595 personas etiquetadas, lo que se traduce en aproximadamente unas 22,64 personas por imagen.
- PersonPath22 [23]: en [24] se expone este conjunto de datos. El 20 % de los datos proviene de otros *datasets* como son Pathtrack [27], Virat [26] y MEVA [25], mientras que el 80 % restante de los datos ha sido extraído directamente de sitios web.

  Contiene un total de 236 escenas divididas entre entrenamiento y test, estableciéndose el 60 % de las escenas como entrenamiento (lo que se traduce en 138 vídeos) y el 40 % restante en test (98 vídeos). Estas 138 escenas de entrenamiento contienen un total de 118.685 imágenes anotadas.

  Esta recopilación destaca por su gran diversidad en cuanto a escenas con distintos ángulos de cámara, condiciones medioambientales, iluminación y densidad de personas.
- Human Dataset V2 [22]: este conjunto de datos cuenta con un total de 13.660 imágenes, repartidas en 10.939 de entrenamiento, 1.571 de validación y 1.150 de test.

Tras una investigación individual de las distintas opciones, se decidió optar por el uso del PersonPath22 debido a la diversidad entre las imágenes, así como a la gran cantidad de imágenes con las que este cuenta. En la figura 3.1 se puede ver un ejemplo de imagen perteneciente a este conjunto.

Para obtener este *dataset*, es necesario seguir los pasos indicados en su repositorio oficial de Github [44].

En este conjunto de datos, las cajas de anotaciones pueden venir tanto en formato visible como amodal. En el formato visible, únicamente contendrán la zona del objeto que puede ser vista en la imagen, mientras que en el formato amodal se tratará de etiquetar la extensión completa del objeto, incluyendo aquellas zonas ocultadas por otro objeto. En la figura 3.2 se pueden ver las diferencias.

En este caso, se utilizarán las etiquetas proporcionadas en el conjunto de datos siguiendo el formato visible. A partir de dichas etiquetas, se extraerá la información relevante para el entrenamiento del modelo, que incluye el nombre de la clase correspondiente a cada objeto detectado, la resolución del vídeo de origen y los valores asociados a cada *bounding box*.

El conjunto de imágenes original contenía una amplia variedad de etiquetas que describían diferentes situaciones asociadas a la clase "persona", tales como persona, persona

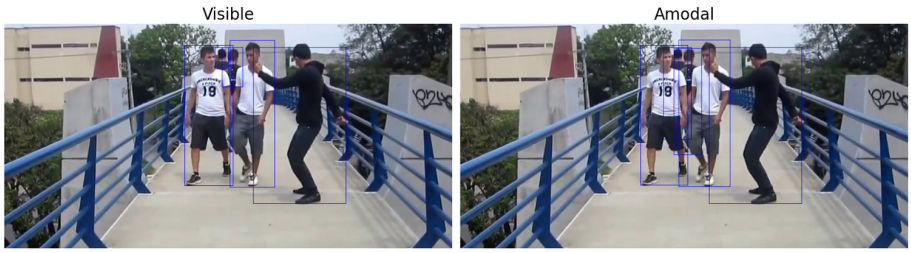


**Figura 3.2:** Comparativa entre los formatos visible y amodal

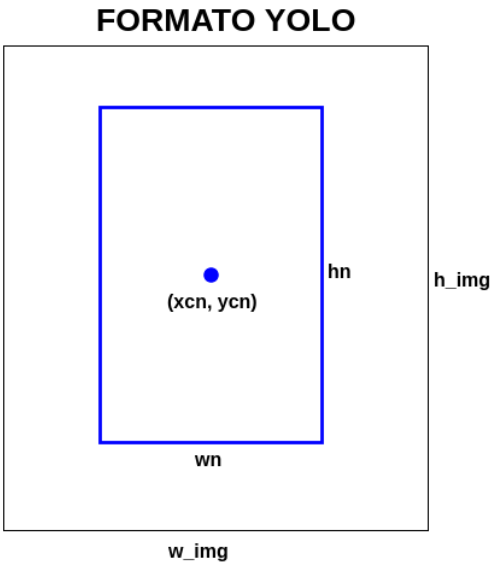


**Figura 3.3:** Bounding box en formato YOLO

de pie, persona en un vehículo, multitud, persona poco visible o reflejo, entre otras. Sin embargo, para los fines de este trabajo, se unificaron todas las categorías relevantes bajo una única clase general denominada persona. Como parte de este proceso de unificación, se descartaron aquellas etiquetas que no resultaban adecuadas para el entrenamiento del modelo, como multitud, persona poco visible o reflejo, ya que no representaban instancias individuales claramente delimitadas o no aportaban información demasiado útil para la detección de las personas.

Los *bounding boxes* originales son definidos mediante el mínimo valor del eje X, el mínimo del eje Y, el ancho y la altura. En cambio, el formato utilizado por YOLO representa las cajas mediante los valores normalizados del valor central del eje X, el valor central del eje Y, el ancho y la altura. En la figura 3.3 se puede ver un ejemplo de *bounding box* en formato YOLO.

A continuación, se procedió a seleccionar el modelo de detección de objetos más adecuado para la tarea. Tras analizar diversas alternativas, se optó por trabajar con la familia de modelos YOLO (*You Only Look Once*), ampliamente reconocida en el ámbito de la visión por computador por su eficiencia y precisión. En particular, se consideraron distintas versiones de la arquitectura, desde las más consolidadas hasta las más recientes. Aunque en los últimos meses han surgido versiones más avanzadas, como YOLOv11, que muestran mejoras en las métricas de evaluación, se decidió utilizar YOLOv8 debido a los resultados y el apoyo con el que cuenta por parte de la comunidad investigadora y desarrolladora. Es importante señalar que la aparición de nuevos modelos con mejores resultados no implica necesariamente que las versiones anteriores hayan quedado obsoletas. Como se observa en la tabla 3.2, YOLOv8 presenta un rendimiento competitivo

sobre el conjunto de datos COCO. Comparando los resultados con los de YOLOv11 en la tabla 3.3, pueden verse mejores métricas en el caso de YOLOv11, pero apenas significativas, siendo la diferencia de mAP de un 2,2 % en el peor de los casos. Por tanto, las métricas alcanzadas por YOLOv8 siguen siendo excelentes y lo convierten en una opción plenamente válida para este trabajo.

| Modelo | mAP (50-95) | Latencia (ms) | Parámetros (M) |
|---|---|---|---|
| YOLOv8n | 37,3 % | 80,4 | 3,2 |
| YOLOv8s | 44,9 % | 128,4 | 11,2 |
| YOLOv8m | 50,2 % | 234,7 | 25,9 |
| YOLOv8l | 52,9 % | 375,2 | 43,7 |
| YOLOv8x | 53,9 % | 479,1 | 68,2 |

**Tabla 3.2:** Métricas obtenidas por la serie de modelos YOLOv8. Extraída de [43]

| Modelo | mAP (50-95) | Latencia (ms) | Parámetros (M) |
|---|---|---|---|
| YOLO11n | 39,5 % | 56,1 | 2,6 |
| YOLO11s | 47,0 % | 90,0 | 9,4 |
| YOLO11m | 51,5 % | 183,2 | 20,1 |
| YOLO11l | 53,4 % | 238,6 | 25,3 |
| YOLO11x | 54,7 % | 462,8 | 56,9 |

**Tabla 3.3:** Métricas obtenidas por la serie de modelos YOLOv11. Extraída de [43]

Finalmente, fue necesario seleccionar una de las variantes disponibles dentro de la serie YOLOv8, la cual ofrece distintos modelos que varían en tiempo de detección, número de parámetros y rendimiento. Tras comparar las diferentes opciones en función de métricas clave como el número de parámetros, el mAP obtenido y la latencia durante la inferencia, se concluyó que la opción más adecuada para este trabajo era el modelo YOLOv8s. Esta elección responde a un compromiso entre precisión y eficiencia computacional, ya que YOLOv8s proporciona resultados competitivos con una carga computacional significativamente menor que las versiones más grandes, lo que facilita su uso en entornos con recursos limitados y acelera el proceso experimental.

## 3.2 Primera ejecución y problemas encontrados

Una vez definidos los distintos medios con los que poder realizar las pruebas, es decir, se contaba con la máquina, el conjunto de imágenes y con el modelo para el entrenamiento, se podía realizar una primera prueba. Este primer ensayo tenía como propósito principal comprobar que todos los componentes del sistema funcionaban correctamente de forma conjunta, y que el modelo base operaba según lo esperado. Además, esta ejecución inicial serviría como punto de partida para evaluar el comportamiento general del modelo, identificar posibles errores o limitaciones y planificar los ajustes o mejoras necesarias en fases posteriores.

Como se ha explicado antes, la máquina de trabajo era la NVIDIA Jetson Orin Nano 8GB. Esta máquina, a pesar de ofrecer buenos resultados en la inferencia, la parte del entrenamiento de los modelos es más complicada. Sus capacidades son limitadas, no permitiendo ejecuciones que demanden un elevado coste computacional.

La serie de modelos YOLOv8 cuenta con una gran cantidad de parámetros a editar por el usuario para realizar ejecuciones personalizadas. En este caso, nos centraremos en desactivar aquellos parámetros relacionados con el aumento de datos, de forma que

obtengamos cómo funciona el modelo únicamente usando las imágenes base. Además, teniendo en cuenta las limitaciones de la máquina, también se debían editar los parámetros para evitar sobrecargar la misma. Por ello, se estableció 384x384 píxeles como tamaño de imagen para el entrenamiento y un tamaño de *batch* de 16.

Debido a la elevada cantidad de imágenes que se tenían, se concluyó que establecer el número de épocas en diez sería suficiente para comprobar la eficiencia del modelo. De esta forma, se podrían identificar posibles problemas, obteniendo una primera aproximación sobre su eficiencia. El resultado de este entrenamiento puede visualizarse en la tabla 3.4.

| Precisión | Recall | mAP50 | mAP50-95 | Tiempo de ejecución |
|---|---|---|---|---|
| 75 % | 46 % | 54 % | 29 % | 21:43:00 |

**Tabla 3.4:** Resultado de la primera ejecución en la NVIDIA Jetson Orin Nano 8GB

Los resultados obtenidos fueron bastante poco favorables. Tomando como medida de referencia el mAP50, el cual es menos estricto comparándolo con el mAP50-95 obtenía valores realmente bajos. Este valor rondaba el 50 %, es decir, el modelo no era capaz de generar predicciones precisas y consistentes. La medida de *recall* tampoco alcanzaba estos valores, obteniendo valores por debajo del 50 % e indicándonos que no solo el modelo cometía errores ubicando a las personas, sino que ni siquiera era capaz de detectarlas en numerosos casos. Además de ello, debido a la alta cantidad de imágenes, los tiempos de ejecución acababan siendo realmente elevados.

A raíz de los resultados obtenidos durante la evaluación del modelo, se llevó a cabo un análisis exhaustivo del conjunto de imágenes utilizado en el entrenamiento y la validación. El objetivo principal de este análisis fue verificar la calidad de las anotaciones, comprobando si las etiquetas asignadas a las imágenes estaban correctamente definidas. Es decir, se quería comprobar si las anotaciones contaban con errores como *bounding boxes* mal posicionadas o personas sin anotar.

Gracias a este análisis, se encontraron errores notables en uno de los subconjuntos que componen PersonPath22. El subconjunto PathTrack era el responsable. En [27], se explica el proceso de etiquetado de este subconjunto, destacando que se recurre a un método de interpolación asistida para acelerar la anotación de trayectorias. Este proceso combina anotaciones manuales clave, realizadas en ciertos fotogramas, con la generación automática de trayectorias intermedias, basadas en la suposición de movimiento lineal entre puntos. De este modo, se reduce significativamente el tiempo necesario para etiquetar secuencias largas.

Este proceso, aunque permite generar grandes cantidades de imágenes anotadas, acaba añadiendo ruido al modelo mediante imágenes incorrectas. En la figura 3.4 se puede ver anotado en color rojo distintos errores, como distintos *bounding boxes* para una misma persona o etiquetas que no están asignadas a ninguna persona. En la figura 3.5 puede verse una imagen en negro sin ninguna persona que tiene dos etiquetas asignadas.

## 3.3 Cambios realizados

Al identificarse un número considerable de imágenes con problemas de anotación pertenecientes al subconjunto PathTrack, las cuales afectaban a la calidad de entrenamiento del modelo, se consideró que la mejor decisión era excluir completamente este subconjunto. Esta medida produjo una reducción del conjunto final de imágenes, a un total de 30.000 imágenes para entrenamiento, 5.000 para validación y 5.000 para test.

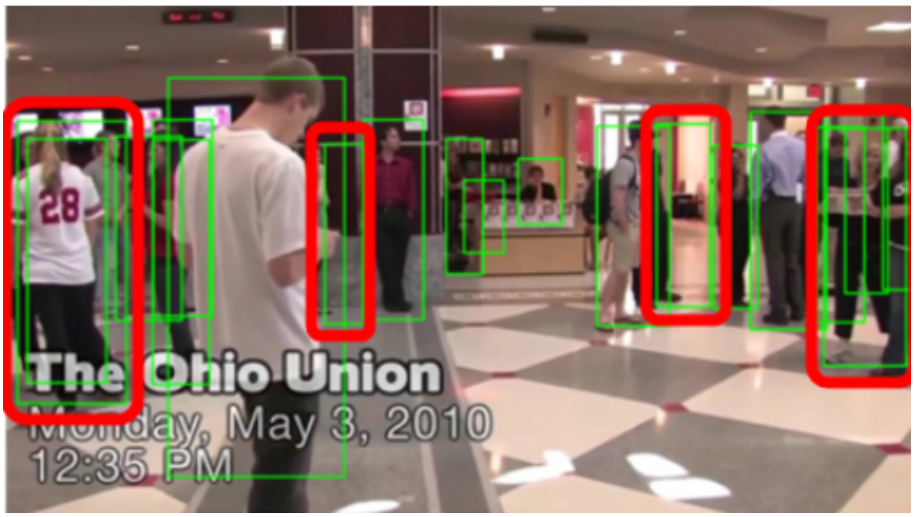


**Figura 3.4:** Imagen con etiquetas incorrectas

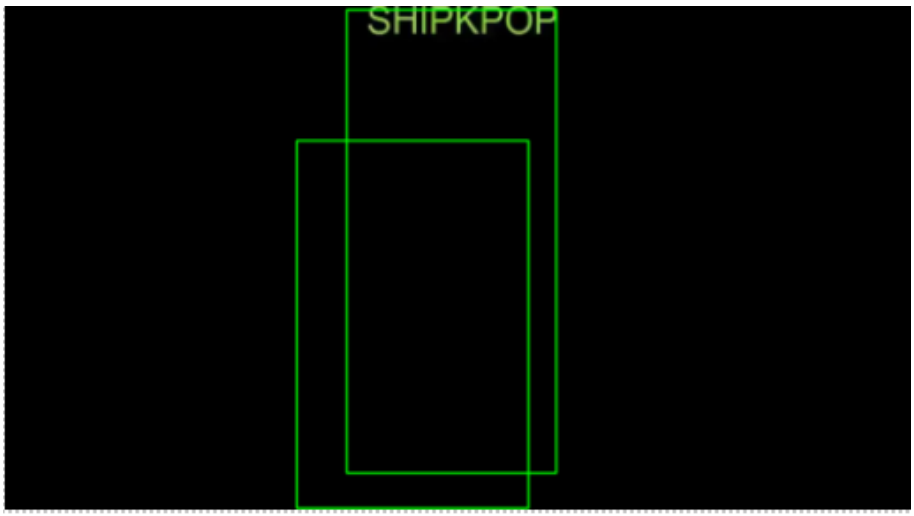


**Figura 3.5:** Imagen con etiquetas incorrectas

Inicialmente, la máquina empleada era una NVIDIA Jetson Orin Nano de 8 GB. Esta máquina fue reemplazada por una NVIDIA Jetson AGX Orin de 64 GB, un dispositivo considerablemente más potente tanto en términos de memoria como de capacidad de procesamiento. Esta mejora permitió incrementar la carga computacional soportada, facilitando el entrenamiento de modelos más complejos mediante configuraciones más demandantes. Las especificaciones de esta máquina se pueden ver en la tabla 3.5

| Especificación | Detalles |
|---|---|
| **Rendimiento de IA** | 275 TOPS |
| **GPU** | GPU de arquitectura NVIDIA Ampere de 2048 núcleos con 64 núcleos Tensor |
| **Frecuencia máxima GPU** | 1,3 GHz |
| **CPU** | CPU Arm® Cortex®-A78AE v8.2 de 12 núcleos y 64 bits 3 MB L2 + 6 MB L3 |
| **Frecuencia máxima CPU** | 2,2 GHz |
| **Memoria** | LPDDR5 de 64 GB y 256 bits 204,8 GB/s |
| **Almacenamiento** | eMMC 5.1 de 64 GB |
| **Consumo de energía** | 15 W-60 W |

**Tabla 3.5:** Especificaciones técnicas del módulo Jetson AGX Orin 64GB. Extraídas de [32]

Tras la implementación de estos dos cambios, se retomaron nuevamente las ejecuciones. En este caso, había dos cambios destacables. Primeramente, la reducción del conjunto de imágenes suponía que el modelo necesitaría una mayor cantidad de épocas para lograr converger obteniendo buenos resultados, pero a la vez supondría una reducción en los tiempos de ejecución. Por esto, se incrementó el número de épocas de 10 a 50. Segundo, el cambio de la máquina implicaba un entorno con una mayor capacidad, permitiendo ejecuciones más exigentes. A partir de esto, el tamaño de imagen durante el entrenamiento se pudo cambiar a 640x640, tamaño que usan los modelos YOLOv8 para ser preentrenados. En la tabla 3.6 se pueden ver los resultados del entrenamiento.

| Precisión | Recall | mAP50 | mAP50-95 | Tiempo de ejecución |
|---|---|---|---|---|
| 84,37 % | 59,51 % | 69,30 % | 43,85 % | 11:15:00 |

**Tabla 3.6:** Resultado de la primera ejecución en la NVIDIA Jetson AGX Orin 64GB

Cabe destacar una mejora notable en el rendimiento del modelo con respecto a la ejecución anterior. Tomando como métrica de referencia el mAP50, los resultados actuales alcanzan valores cercanos al 70 %, lo que representa un incremento de aproximadamente un 15 % en comparación con la versión anterior. Asimismo, la métrica de *recall* experimentó una mejora significativa, acercándose a un valor del 60 %, cuando anteriormente no era capaz de alcanzar el 50 %.

Además de los avances en precisión y detección, se registró una reducción considerable en los tiempos de entrenamiento, disminuyéndose a cerca de la mitad del tiempo necesario en la ejecución previa.

# CAPÍTULO 4
# Metodología propuesta

## 4.1 Solución propuesta

La solución propuesta se basa en el desarrollo de una técnica automática de aumento de datos. El objetivo principal de esta técnica es seleccionar, a partir de un conjunto predefinido de transformaciones, aquellos valores o configuraciones que resulten más beneficiosos para el rendimiento del modelo de detección. Para ello, la técnica debe incorporar un mecanismo de evaluación que le permita identificar qué transformaciones contribuyen de manera positiva al proceso de entrenamiento.

La validación de esta técnica se lleva a cabo en función del cumplimiento de los objetivos definidos en la sección 1.2. El primero de estos objetivos consiste en verificar que, ante una reducción significativa del conjunto de datos original, el modelo entrenado con la técnica propuesta sea capaz de mantener un rendimiento comparable al obtenido utilizando el conjunto completo de datos, así como al obtenido mediante otras técnicas automáticas de aumento de datos en el conjunto reducido. En este contexto, se considera que dos modelos presentan rendimientos similares si el valor de la métrica mAP difiere en un margen inferior al 3 %. Las variaciones dentro de ese rango no suelen ser significativas en aplicaciones prácticas. Para la validación de esta métrica se establece una época de referencia en la que comparar la ejecución de los distintos modelos, en este caso la época 23.

El segundo objetivo es la mejora en los tiempos de ejecución. La técnica desarrollada debe no solo mantener el rendimiento del modelo, sino también optimizar el tiempo requerido para el entrenamiento.

## 4.2 Diseño y desarrollo de la solución

Esta sección está enfocada en mostrar el trabajo realizado hasta alcanzar la solución final, partiendo desde una propuesta inicial y recorriendo todas las etapas intermedias. Se detallan las distintas aproximaciones probadas, los ajustes aplicados y las decisiones tomadas en función de los resultados obtenidos. Además, para facilitar la comparación entre versiones, todas las métricas de precisión se han contrastado tomando como referencia los resultados obtenidos en la época 23.

**RandAugment en el conjunto de imágenes**

Primeramente, se comprobó el funcionamiento del aumento de datos realizado por RandAugment tanto en el conjunto original como en un conjunto reducido de imágenes, formado únicamente por el 10 % del total de imágenes.

Con el objetivo de comprobar la eficacia del aumento de datos en este conjunto de imágenes, se procedió aplicando la técnica RandAugment estableciendo los siguientes parámetros: un tamaño de imagen de 640×640 píxeles y un tamaño de *batch* de 64. En la tabla 4.1 se pueden comprobar las diferencias obtenidas tras la aplicación de esta técnica.

| RandAugment | Precisión | Recall | mAP50 | mAP50-95 | Tiempo de ejecución |
|---|---|---|---|---|---|
| No | 82,63 % | 58,17 % | 67,04 % | 40,60 % | 11:15:00 |
| Sí | 86,24 % | 62,33 % | 71,87 % | 45,58 % | 11:15:00 |

**Tabla 4.1:** Métricas obtenidas tras la ejecución de RandAugment

Al comparar los resultados obtenidos en ambas configuraciones (tabla 4.1), se observa una mejora de entre un 4 % y un 5 % en las métricas de precisión. Esta diferencia resulta realmente significativa. Por tanto, se confirma que el uso de técnicas de aumento de datos, como RandAugment, contribuye de manera clara a aumentar la capacidad de generalización del modelo.

Posteriormente, se repitió el experimento utilizando únicamente el 10 % del conjunto de imágenes, con el objetivo de comprobar el comportamiento del modelo bajo una cantidad limitada de imágenes y mediante el uso de RandAugment. De esta forma, es posible comprobar cómo puede llegar a funcionar el modelo ante una limitación de imágenes, pero con la aplicación del aumento de datos, y definir el objetivo que nuestra técnica debería ser capaz de superar.

| Precisión | Recall | mAP50 | mAP50-95 | Tiempo de ejecución |
|---|---|---|---|---|
| 85,18 % | 61,96 % | 70,85 % | 44,27 % | 1:20:00 |

**Tabla 4.2:** Métricas obtenidas tras la ejecución de RandAugment en un 10 % de las imágenes

En la tabla 4.2 se presentan los resultados obtenidos, los cuales se comparan directamente con los mostrados en la tabla 4.1. Al observar el desempeño del modelo reducido frente al modelo completo sin técnicas de aumento de datos, se aprecia una mejora en la métrica de mAP cercana al 4 %, lo cual es especialmente relevante si se considera la reducción del tamaño del conjunto de imágenes. Por otro lado, al compararlo con el modelo completo que sí incorpora RandAugment, se detecta únicamente una pérdida del 1 % en mAP. A pesar de esta pequeña diferencia, su impacto en el rendimiento práctico es mínimo.

Donde realmente se aprecia una diferencia significativa es en los tiempos de ejecución. En este caso, el modelo reducido logró operar hasta ocho veces más rápido que los modelos anteriores.

**Primera versión de la técnica**

A continuación se trató de definir una técnica de aumento de datos no sólo que fuese competitiva respecto al modelo original sin aumento de datos, sino que también fuese competitiva respecto al modelo reducido aplicándole RandAugment. Como se ha ex-

plicado en los objetivos, la idea era mantener las métricas de precisión reduciendo los tiempos de ejecución. A partir de esto, se comenzó a definir la misma.

Se trataba de una búsqueda iterativa, en que a partir del estudio de un conjunto de transformaciones para el aumento de datos, se trataría de encontrar la mejor combinación de estas y, de esta forma, mejorar el rendimiento del modelo.

Por ejemplo, en el caso de establecer una serie de transformaciones a estudiar, se trataría de buscar el mejor valor de estas entre cero y uno con saltos de 0,1 comenzando en cero. Con el resto de transformaciones a cero, se comenzaría estableciendo la primera transformación a 0,1. A partir del valor que se obtuviese, se aplicaría un valor de 0,2 y se volvería a repetir el entrenamiento. En caso de que en esta segunda iteración, el modelo empeorase, se mantendría el valor anterior previo a empeorar y se comenzaría la evaluación con la siguiente transformación de la misma forma. En esencia, la evaluación de cada transformación se realiza teniendo en cuenta los resultados obtenidos por las anteriores, permitiendo así construir de forma progresiva una configuración óptima.

En este caso, se aplicaron un conjunto de transformaciones que podrían afectar positivamente al entrenamiento de nuestro modelo. A continuación, se detallan los nombres de estas transformaciones, según como se utilizan en la configuración de YOLOv8 [43], junto con una breve descripción de su función:

- Mosaic: combinación de cuatro imágenes.
- Translate: traslada la imagen horizontal y/o verticalmente.
- FlipLR: gira la imagen horizontalmente.
- Scale: amplia parte de la imagen, de forma que se reduzca el campo de visión.
- Hsv_H: ajuste del tono de la imagen.
- HSV_S: ajuste de la saturación de la imagen.
- HSV_V: ajuste del brillo de la imagen.

Una vez establecida la base teórica, se realizó la implementación práctica. En este caso, se establecieron un número de diez épocas para la evaluación de cada valor de cada transformación.

| Precisión | Recall | mAP50 | mAP50-95 | Tiempo de ejecución |
|---|---|---|---|---|
| 85,89 % | 62,63 % | 71,44 % | 44,85 % | 8:30:00 |

**Tabla 4.3:** Métricas obtenidas tras la ejecución de la técnica iterativa

En la tabla 4.3 se muestran los resultados obtenidos. En este caso, se observa que, en comparación con el modelo original sin aumentado de datos presentado en la tabla 4.1, las métricas de precisión son mejores, con una mejora de mAP aproximada del 4 %. No obstante, aunque los tiempos de ejecución se ven reducidos con respecto al modelo original, al compararlos con el modelo reducido con RandAugment de la tabla 4.2, siguen siendo considerablemente más altos, llegando a ser aproximadamente seis veces superiores.

Por tanto, esta aproximación no resulta adecuada debido al elevado coste computacional que implica la búsqueda individual de los valores óptimos para cada transformación. Incluso aplicando una estrategia de reducción de complejidad, como realizar

incrementos de 0,2 en los valores y limitar la búsqueda a tres épocas por transformación, seguidas de diez épocas adicionales para garantizar la convergencia, los tiempos de ejecución siguen siendo considerables, alcanzando aproximadamente unas tres horas de duración.

### Segunda versión de la técnica

Con el fin de alcanzar los resultados temporales que ofrece el método de RandAugment en el conjunto de datos reducido, se modificó el planteamiento en el que se acepta que una transformación va a entrar en la solución. Con este objetivo, se sustituye la búsqueda iterativa por una búsqueda binaria.

De esta forma, la decisión de aplicar una transformación se reduce a aplicar la transformación o no aplicarla. Es decir, en lugar de buscar el valor de la transformación que mejor solución aporta, se parte con un valor prefijado, y con este se puede determinar si la transformación debe entrar en la solución o no. De este modo, se simplifica considerablemente el espacio de búsqueda y se acelera el proceso de evaluación.

La técnica que se aplica sigue un procedimiento estructurado, diseñado para seleccionar las transformaciones que mejor deben funcionar. En primer lugar, se debe asignar el valor fijo con el que se evalúa cada transformación. Además, en esta técnica se usan tres épocas para evaluar cada transformación y diez épocas finales para asegurar la convergencia del modelo.

El proceso comienza ejecutando el modelo sin aplicar ninguna transformación. El valor mAP obtenido por la ejecución es almacenado. A continuación, se evalúa cada una de las transformaciones de forma individual. Únicamente aquellas transformaciones que, aplicándose individualmente, generan una mejora superior al 0,1 % en el valor mAP respecto al modelo base, se consideran candidatas para la solución final.

Una vez identificadas las transformaciones candidatas, estas se incorporan una a una, de mejor valor a peor valor obtenido, hasta que la inclusión de una nueva transformación provoque un deterioro en el rendimiento del modelo. En ese punto, se detiene la búsqueda de transformaciones, y se ejecutan, con las transformaciones previas a la detención, un total de diez épocas para asegurar la convergencia del modelo.

| Precisión | Recall | mAP50 | mAP50-95 | Tiempo de ejecución |
|---|---|---|---|---|
| 85,44 % | 62,24 % | 70,39 % | 43,24 % | 1:35:00 |

**Tabla 4.4:** Métricas obtenidas tras la ejecución de la técnica binaria con 0,25

Los resultados de la aplicación de la técnica con un valor fijo de 0,25 se pueden observar en la tabla 4.4. En relación con las métricas obtenidas, puede observarse que los valores de mAP obtienen resultados comparables con la técnica de RandAugment, dentro del rango aceptable. Además, los tiempos de ejecución también son cercanos a los que ofrece esta técnica, en este caso itera 15 minutos más lento. Sin embargo, los resultados no cumplen los objetivos esperados, es decir, no es capaz de suponer una mejora temporal significativa.

### Tercera versión de la técnica

Partiendo de la técnica anterior, se buscó una solución que consiguiese reducir los tiempos. Para lograrlo, la búsqueda de las transformaciones a aplicar se realizó sobre el

1 % de las imágenes respecto al total de imágenes originales, lo que permitió una evaluación preliminar más sencilla.

A diferencia de como se procedía con la implementación anterior, en esta ocasión no se descartaban las transformaciones que no superaban el rendimiento del modelo base. En este caso, no era necesario evaluar cómo funcionaba el modelo sin aplicarle transformaciones, sino que se almacenaba el valor de mAP obtenido para cada transformación evaluada de forma individual, sin descartar ninguna.

Una vez recopilados los resultados, se comenzaba una segunda etapa con un 10 % de los datos. En esta fase, estas transformaciones se evaluaban una a una de mejor a peor, según los resultados obtenidos anteriormente.

La evaluación consistía en añadir una transformación al mejor modelo actual (o al modelo base si se trata de la primera). Si la incorporación de esta transformación producía una mejora igual o superior al 0,2 % en el valor de mAP, se continuaba añadiendo la siguiente transformación según el orden establecido. En caso contrario, se interrumpía la incorporación de nuevas transformaciones, y se daba inicio a la fase final de convergencia, utilizando el modelo con el mejor rendimiento alcanzado hasta el momento.

Al igual que en la técnica anterior, las evaluaciones tomaban tres épocas, mientras que la fase de convergencia serían diez épocas. Además, el valor establecido para las transformaciones era de 0,25.

| Precisión | Recall | mAP50 | mAP50-95 | Tiempo de ejecución |
|---|---|---|---|---|
| 85,65 % | 62,02 % | 70,21 % | 43,21 % | 0:53:00 |

**Tabla 4.5:** Métricas obtenidas tras la ejecución de la técnica binaria con reducción de imágenes

El resultado de la ejecución de la técnica puede comprobarse en la tabla 4.5. En este caso, los tiempos de ejecución resultan significativamente más favorables, logrando una mejora de aproximadamente 25 minutos respecto al método RandAugment, mientras que el valor de mAP se mantiene en niveles similares, lo que confirma la eficiencia del nuevo enfoque sin comprometer el rendimiento del modelo.

#### Última versión de la técnica

Se trató de reducir aún más los tiempos de ejecución, con el objetivo de marcar una diferencia más significativa respecto al modelo reducido con RandAugment. Para ello, se introdujo una ligera modificación en la técnica previamente descrita. En esta nueva versión, la segunda fase, en la que originalmente se reemplazaba el 1 % de las imágenes por un 10 % para evaluar las transformaciones de forma secuencial desde la mejor hasta la peor, se llevó a cabo utilizando también solo el 1 % de los datos. Este ajuste permitió acelerar dicha fase del proceso y, por tanto, reducir el tiempo total de ejecución de la solución.

| Precisión | Recall | mAP50 | mAP50-95 | Tiempo de ejecución |
|---|---|---|---|---|
| 85,24 % | 61,36 % | 69,74 % | 42,73 % | 0:34:00 |

**Tabla 4.6:** Métricas obtenidas tras la última técnica

En la tabla 4.6 se puede comprobar que, en este caso, la técnica propuesta presenta una diferencia notable respecto a RandAugment, logrando reducir el tiempo total de ejecución en aproximadamente 50 minutos, mientras mantiene resultados comparables

en el resto de métricas, incluyendo el mAP, lo que muestra una mejora significativa en términos de eficiencia sin sacrificar rendimiento.

# CAPÍTULO 5
# Evaluación de los resultados

En este apartado se presentan y analizan los resultados obtenidos a lo largo de las distintas ejecuciones realizadas durante el desarrollo del proyecto. El objetivo principal es comprobar si la técnica automática de aumento de datos propuesta cumple con los criterios establecidos en los objetivos del trabajo.

## 5.1 Comparativa de las distintas técnicas desarrolladas

Durante el proceso de desarrollo se implementaron diversas versiones de la técnica, cada una con ajustes y mejoras orientadas a aumentar su eficiencia sin comprometer significativamente la precisión del modelo. Mediante sucesivos experimentos, se pudo evaluar cada una de las versiones en rendimiento y eficiencia. En la tabla 5.1 puede verse la comparativa de resultados.

Los resultados muestran una evolución positiva: mientras que las primeras versiones ofrecían métricas de precisión más altas, su coste computacional era elevado. En cambio, la versión final logró reducir el tiempo de entrenamiento aproximadamente 16 veces respecto a la primera aproximación, manteniendo un rendimiento similar en términos de mAP. Aunque las métricas de precisión de la versión final no son las más altas obtenidas, las diferencias entre las versiones son de menos de un 2 % de mAP, lo que implica que en la práctica se acaban obteniendo resultados muy similares.

| | **mAP50** | **Tiempo de ejecución** |
|---|---|---|
| Primera aproximación | 71,44 % | 8:30:00 |
| Segunda aproximación | 70,39 % | 1:35:00 |
| Tercera aproximación | 70,21 % | 0:53:00 |
| Última aproximación | 69,74 % | 0:34:00 |

**Tabla 5.1:** Comparativa de resultados entre los desarrollos de la técnica

## 5.2 Resultados y comparación con métodos de referencia

Además del análisis detallado de las versiones internas desarrolladas, se ha llevado a cabo una comparación con configuraciones comunes ampliamente utilizadas, con el objetivo de evaluar el rendimiento relativo de la técnica propuesta. Los resultados obtenidos

se recogen en la tabla 5.2, donde se incluyen tanto modelos entrenados con la totalidad del conjunto de datos como versiones reducidas al 10 %.

En el caso del conjunto reducido, la técnica desarrollada muestra una mejora temporal significativa. Concretamente, logra reducir el tiempo de ejecución en aproximadamente 2,5 veces en comparación con RandAugment, manteniendo un nivel de precisión prácticamente equivalente. La diferencia en mAP es inferior al 1 %, lo que indica un rendimiento muy similar en términos de detección.

Al ampliar la comparación con los modelos entrenados sobre el conjunto completo de datos, se observan diferencias algo mayores en la métrica de mAP. En concreto, la técnica propuesta supera en un 2,7 % el rendimiento del modelo completo que no emplea ningún tipo de aumento de datos. Por otro lado, si se compara con el modelo completo que sí incorpora RandAugment, se obtiene un valor de 2,1 % menos en mAP. No obstante, estas variaciones se consideran poco significativas en escenarios prácticos, donde márgenes inferiores al 3 % en este tipo de tareas suelen situarse dentro de los límites aceptables de rendimiento.

Lo más destacable es la notable mejora en los tiempos de entrenamiento. Frente a los modelos entrenados con el 100 % de los datos, la técnica propuesta permite una reducción de tiempo de hasta 20 veces, pasando de más de 11 horas a tan solo 34 minutos.

| **% Datos** | **Aumento de datos** | **mAP50** | **Tiempo de ejecución** |
|---|---|---|---|
| 100 % | No | 67,04 % | 11:15:00 |
| 100 % | RandAugment | 71,87 % | 11:15:00 |
| 10 % | RandAugment | 70,85 % | 1:20:00 |
| 10 % | Técnica propuesta | 69,74 % | 0:34:00 |

**Tabla 5.2:** Comparativa de resultados entre diferentes configuraciones

# CAPÍTULO 6
# Conclusiones

A lo largo de este trabajo se ha desarrollado y evaluado una técnica de aumento de datos automática orientada a tareas de detección de personas. El objetivo principal ha sido mejorar la eficiencia del proceso de entrenamiento, manteniendo niveles de precisión comparables a los obtenidos mediante métodos automáticos con el mismo objetivo.

Los resultados experimentales obtenidos permiten afirmar que los objetivos planteados se han cumplido satisfactoriamente. En primer lugar, la técnica propuesta ha demostrado ser capaz de alcanzar valores de rendimiento similares a los obtenidos con el conjunto de datos completo, utilizando únicamente un 10 % del total de imágenes. Esta reducción en el volumen de datos ha implicado, además, una mejora muy significativa en los tiempos de ejecución del entrenamiento, lo cual contribuye a una mayor eficiencia computacional.

En segundo lugar, al comparar el rendimiento de la técnica desarrollada con el de RandAugment se ha comprobado que son igualmente competitivos. Sin embargo, la técnica propuesta ofrece ventajas en términos de tiempo de entrenamiento, reduciendo notablemente la duración del proceso respecto a RandAugment, sin sacrificar precisión.

En resumen, se concluye que la técnica desarrollada constituye una solución útil y eficaz dentro del contexto de las tareas de detección de personas. Ha demostrado ser capaz de competir en rendimiento con técnicas automáticas de aumento de datos ampliamente utilizadas, ofreciendo además ventajas en términos de eficiencia computacional. Estos resultados refuerzan la utilidad de la propuesta y evidencian su potencial como alternativa válida frente a las técnicas existentes en escenarios donde la reducción de datos y tiempos de entrenamiento es prioritaria.

# CAPÍTULO 7
# Relación del trabajo desarrollado con los estudios cursados

A lo largo de los estudios realizados en el Grado en Ingeniería Informática, en este caso dentro de la rama de Computación, se han adquirido una serie de competencias fundamentales que han sido puestas en práctica durante el desarrollo de este Trabajo de Fin de Grado.

Para implementar la técnica propuesta ha sido necesario desarrollar una clase que guiase el funcionamiento del sistema a lo largo de sus diferentes fases. Los conocimientos necesarios para ello fueron adquiridos principalmente en asignaturas como Introducción a la Informática y a la Programación (IIP) y Programación (PRG), donde se establecieron las bases de la programación estructurada y la orientación a objetos.

Por otro lado, asignaturas como Interfaces Persona Computador (IPC), Ingeniería del Software (ISW) y Estructuras de Datos y Algoritmos (EDA) han sido fundamentales para garantizar la calidad y organización del código desarrollado. Estas materias han proporcionado herramientas para aplicar buenas prácticas en el diseño del software, así como para estructurar y manipular correctamente los datos necesarios para establecer los parámetros óptimos en el proceso de aumento de datos.

Dos asignaturas que han tenido un peso destacado en la comprensión de los fundamentos del proyecto han sido Sistemas Inteligentes (SIN) y Percepción (PER). Estas asignaturas introdujeron los primeros conceptos relacionados con el aprendizaje automático, sentando las bases necesarias para abordar el análisis y diseño de soluciones basadas en modelos de aprendizaje profundo.

No obstante, la asignatura que ha tenido un impacto más directo y significativo en el desarrollo del proyecto ha sido la asignatura de Aprendizaje Automático (APR). Esta materia permitió adquirir una sólida introducción a las redes neuronales convolucionales. Si bien durante el desarrollo del trabajo se ha profundizado en arquitecturas más avanzadas como YOLOv8, el contenido de la asignatura ha resultado esencial para comprender la estructura interna de estos modelos y abordar con criterio técnico las decisiones relativas al entrenamiento, evaluación y mejora del rendimiento.

# CAPÍTULO 8
# Trabajos futuros

A partir de los resultados obtenidos en este trabajo, se pueden abrir diversas líneas de desarrollo que podrían explorarse en futuras investigaciones.

En primer lugar, sería interesante evaluar el funcionamiento de la técnica en tareas distintas a la detección de personas, como la detección de objetos genéricos, la clasificación de imágenes o la segmentación semántica. Esto permitiría validar su aplicabilidad y efectividad en contextos más amplios dentro del ámbito de la visión por computador.

Otra mejora interesante sería aplicar la técnica en diferentes configuraciones de reducción de datos. En este caso se ha evaluado sobre un 10 % del conjunto de datos, pero sería beneficioso realizar la evaluación sobre un 5 % o un 1 %, y de esta forma conocer su límite y potencial ante una importante escasez de datos.

# Bibliografía


[1] Paul Viola i Michael Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pág. I–I, Kauai, Hawaii, EUA, 2001.

[2] Navneet Dalal i Bill Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pág. 886–893, San Diego, Califòrnia, EUA, 2005.

[3] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester i Deva Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, núm. 9, pág. 1627–1645, septiembre de 2009.

[4] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, vol. 36, núm. 4, pág. 193–202, abril de 1980.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio i Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86, núm. 11, pág. 2278–2324, enero de 1998.

[6] Alex Krizhevsky, Ilya Sutskever i Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Proceedings of the Neural Information Processing Systems (NIPS)*, vol. 25, pág. 1097–1105, diciembre de 2012.

[7] Yibo Sun, Zhe Sun i Weitong Chen. The evolution of object detection methods. *Engineering Applications of Artificial Intelligence*, vol. 133, pág. 108458, abril de 2024.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser i Illia Polosukhin. Attention is All You Need. *arXiv (Cornell University)*, vol. 30, pág. 5998–6008, junio de 2017.

[9] Robin M. Schmidt. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. *arXiv*, 2019.

[10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov i Sergey Zagoruyko. End-to-End Object Detection with Transformers. *Lecture Notes in Computer Science*, pág. 213–229, 2020.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit i Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations*, mayo de 2021.

[12] Jan Hosang, Rodrigo Benenson i Bernt Schiele. Learning non-maximum suppression. *arXiv preprint*, arXiv:1705.02950, 2017.

[13] Alhassan Mumuni i Fuseini Mumuni. Data augmentation: A comprehensive survey of modern approaches. *Array*, vol. 16, pág. 100258, noviembre de 2022.

[14] Sergey Ioffe i Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li i Jian Sun. YOLOX: Exceeding YOLO Series in 2021. *arXiv preprint*, arXiv:2107.08430, julio de 2021.

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever i Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, vol. 15, núm. 1, pág. 1929–1958, enero de 2014.

[17] Teerath Kumar, Alessandra Mileo, Rob Brennan i Malika Bendechache. Image Data Augmentation Approaches: A Comprehensive Survey and Future Directions. *arXiv*, pág. 2301.02830, enero de 2023.

[18] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens i Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, junio de 2020.

[19] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan i Quoc V. Le. AutoAugment: Learning Augmentation Strategies from Data. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pág. 113–123, junio de 2019.

[20] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer i Balaji Lakshminarayanan. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. *International Conference on Learning Representations*, abril de 2020.

[21] CrowdHuman. *CrowdHuman Dataset*. Disponible en: https://www.crowdhuman.org/

[22] Human v2. *Human Dataset v2 Object Detection Dataset and Pre-Trained Model*. Disponible en: https://universe.roboflow.com/human-v2/human-dataset-v2

[23] Webthemez. *PersonPath22*. Disponible en: https://amazon-science.github.io/tracking-dataset/personpath22.html

[24] Bing Shuai, Alessandro Bergamo, Uta Buechler, Andrew Berneshawi, Alyssa Boden i Joe Tighe. Large Scale Real-World Multi Person Tracking. *European Conference on Computer Vision (ECCV)*, Springer, 2022.

[25] Kevin Corona, Kyle Osterdahl, Robert Collins i Arthur Hoogs. MEVA: A Large-Scale Multiview, Multimodal Video Dataset for Activity Detection. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pág. 1060–1068, enero de 2021.

[26] Seonwook Oh, Arthur Hoogs, Abhijit Perera, Nalini Cuntoor, Chia-Chih Chen, Joon-Young Lee, Subhabrata Mukherjee, Jagannath Aggarwal, Hwann-Tzong Lee, Larry Davis i altres. A Large-Scale Benchmark Dataset for Event Recognition in Surveillance Video. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pág. 3153–3160, 2011.

[27] Santiago Manen, Matthias Gygli, Dengxin Dai i Luc Van Gool. PathTrack: Fast Trajectory Annotation with Path Supervision. *2017 IEEE International Conference on Computer Vision (ICCV)*, pág. 290–299, 2017.

[28] Ultralytics. YOLOv8 vs YOLO11: A Technical Comparison. *Ultralytics Documentation*, marzo de 2025. Disponible a: https://docs.ultralytics.com/es/compare/yolov8-vs-yolo11/#performance-comparison.

[29] Deval Shah. Mean Average Precision (mAP) Explained: Everything You Need to Know. *V7 Labs Blog*, sense data. Disponible a: https://www.v7labs.com/blog/mean-average-precision.

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick i Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv preprint arXiv:1405.0312*, 2015.

[31] Barret Zoph i Quoc V. Le. Neural Architecture Search with Reinforcement Learning. *International Conference on Learning Representations (ICLR)*, 2017.

[32] NVIDIA Corporation. *NVIDIA Jetson AGX Orin*. Disponible en: https://www.nvidia.com/es-es/autonomous-machines/embedded-systems/jetson-orin/.

[33] Ross Girshick, Jeff Donahue, Trevor Darrell i Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *arXiv preprint arXiv:1311.2524*, 2014.

[34] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt i B. Schölkopf. Support Vector Machines. *IEEE Intelligent Systems and their Applications*, vol. 13, núm. 4, pág. 18–28, 1998.

[35] Ross Girshick. Fast R-CNN. *arXiv preprint arXiv:1504.08083*, 2015.

[36] V. Klema i A. Laub. The Singular Value Decomposition: Its Computation and Some Applications. *IEEE Transactions on Automatic Control*, vol. 25, núm. 2, pág. 164–176, 1980.

[37] Shaoqing Ren, Kaiming He, Ross Girshick i Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv preprint arXiv:1506.01497*, 2016.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren i Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *Computer Vision – ECCV 2014*, pág. 346–361. Springer International Publishing, 2014.

[39] S. Lazebnik, C. Schmid i J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pág. 2169–2178, 2006.

[40] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu i Alexander C. Berg. SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016*, pág. 21–37. Springer International Publishing, 2016.

[41] Karen Simonyan i Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2015.

[42] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He i Piotr Dollár. Focal Loss for Dense Object Detection. *arXiv preprint arXiv:1708.02002*, 2018.

[43] Ultralytics. *Configuration*. Disponible en: https://docs.ultralytics.com/es/usage/cfg/, marzo de 2025.

[44] Amazon-Science. *GitHub - amazon-science/tracking-dataset*. Disponible en: https://github.com/amazon-science/tracking-dataset.

[45] Chien-Yao Wang, Alexey Bochkovskiy i Hong-Yuan Mark Liao. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pág. 7464–7475, junio de 2023.

[46] Chien-Yao Wang, I-Hau Yeh i Hong-Yuan Mark Liao. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *Lecture notes in computer science*, pág. 1–21, octubre de 2024.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren i Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, núm. 9, pág. 1904–1916, enero de 2015.

[48] Glenn Jocher, Ayush Chaurasia i Jing Qiu. Ultralytics YOLOv8. Versión 8.0.0, 2023. Disponible en: https://github.com/ultralytics/ultralytics.

[49] Alexey Bochkovskiy, Chien-Yao Wang i Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934, 2020.

[50] Glenn Jocher i Jing Qiu. Ultralytics YOLO11. Versión 11.0.0, 2024. Disponible en: https://github.com/ultralytics/ultralytics.

[51] Ao Wang, Hui Chen, Lihao Liu i altres. YOLOv10: Real-Time End-to-End Object Detection. arXiv preprint arXiv:2405.14458, Tsinghua University, 2024.

[52] Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu i Xiangxiang Chu. YOLOv6 v3.0: A Full-Scale Reloading. arXiv preprint arXiv:2301.05586, 2023.

[53] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi i Jiaya Jia. Path Aggregation Network for Instance Segmentation. arXiv preprint arXiv:1803.01534, 2018.

[54] Joseph Redmon i Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767, 2018.

[55] Joseph Redmon i Ali Farhadi. YOLO9000: Better, Faster, Stronger. arXiv preprint arXiv:1612.08242, 2016.

[56] Joseph Redmon, Santosh Divvala, Ross Girshick i Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. arXiv preprint arXiv:1506.02640, 2016.

[57] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan i Serge Belongie. Feature Pyramid Networks for Object Detection. arXiv preprint arXiv:1612.03144, 2017.

[58] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn i Andrew Zisserman. The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, vol. 88, núm. 2, pág. 303–338, septiembre de 2009.

[59] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár i C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science*, pág. 740–755, enero de 2014.

[60] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, núm. 14, págs. 281–297, 1967.

[61] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig i Vittorio Ferrari. The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale. *arXiv preprint arXiv:1811.00982*, pág. 1956–1981, julio de 2020.

[62] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li i Fei-Fei Li. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pág. 248–255, junio de 2009.

[63] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers i A. W. M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, vol. 104, núm. 2, pág. 154–171, abril de 2013.

[64] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke i Andrew Rabinovich. Going Deeper with Convolutions. *arXiv preprint*, arXiv:1409.4842, 2014.

# APÉNDICE A
# OBJETIVOS DE DESARROLLO SOSTENIBLE

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

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

Este Trabajo de Fin de Grado se centra en el desarrollo de una técnica automática de aumento de datos para tareas de detección de personas mediante modelos de visión por computador, concretamente sobre la arquitectura YOLOv8. A pesar de tratarse de un proyecto puramente tecnológico, es posible establecer vínculos directos e indirectos con varios Objetivos de Desarrollo Sostenible (ODS) definidos por las Naciones Unidas en la Agenda 2030. Las implicaciones de este trabajo no se limitan únicamente al rendimiento técnico o a la eficiencia computacional, sino que también impactan en ámbitos relacionados con la sostenibilidad, la accesibilidad tecnológica y el consumo responsable de recursos.

Uno de los ODS con mayor relación con el contenido de este trabajo es el **ODS 9: Industria, innovación e infraestructura**. Este objetivo promueve el desarrollo de infraestructuras sostenibles, la industrialización inclusiva y el fomento de la innovación tecnológica. La técnica desarrollada en este proyecto permite reducir significativamente la cantidad de datos necesarios para entrenar modelos de detección sin afectar de forma sustancial al rendimiento final. Al reducir el volumen de datos y el tiempo de entrenamiento requerido, se facilita la implementación de modelos complejos en dispositivos con recursos computacionales limitados o en entornos industriales donde la eficiencia y el coste energético son factores determinantes. Esto representa una contribución directa a la creación de soluciones tecnológicas más accesibles, escalables y sostenibles para una amplia gama de aplicaciones.

En segundo lugar, este trabajo guarda relación con el **ODS 11: Ciudades y comunidades sostenibles**. Los sistemas de detección de personas son fundamentales en múltiples aplicaciones urbanas: desde sistemas de videovigilancia hasta control de aforo en espacios públicos, o la gestión de emergencias. La eficiencia lograda con la técnica propuesta facilita que estos sistemas puedan desplegarse en ciudades de menor tamaño o en comunidades con limitaciones tecnológicas. Además, al reducir la dependencia de grandes conjuntos de datos etiquetados, se acortan los ciclos de desarrollo, lo que permite una mayor capacidad de adaptación ante situaciones cambiantes, como emergencias sanitarias, eventos multitudinarios o nuevas regulaciones.

Por otro lado, también se establece una conexión con el **ODS 12: Producción y consumo responsables**. En el contexto del aprendizaje automático, los grandes volúmenes de datos y el elevado coste computacional que suelen requerir los modelos más avanzados implican un importante consumo de energía y recursos. La técnica de aumento de datos desarrollada en este proyecto permite entrenar modelos eficaces con un subconjunto reducido de datos, lo que se traduce en una menor demanda computacional, menos consumo energético y menores costes asociados. Esta optimización, aunque de carácter técnico, está alineada con los principios de sostenibilidad en la producción y uso de tecnologías digitales.

Asimismo, puede establecerse una relación indirecta con el **ODS 13: Acción por el clima**. Aunque no es un objetivo explícito del trabajo, es conocido que el entrenamiento de modelos puede tener un impacto ambiental considerable, especialmente cuando se trata de modelos de gran escala que requieren semanas de entrenamiento en centros de datos. Reducir los tiempos de entrenamiento en más de un 90 %, como se ha logrado en algunos experimentos descritos en este proyecto, conlleva también una reducción proporcional en el consumo energético. Por tanto, esta investigación también puede interpretarse como una contribución al desarrollo de modelos de inteligencia artificial más sostenibles desde el punto de vista ambiental.