ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster Universitario en Big Data

USO DE TÉCNICAS DE TRANSFER LEARNING EN UN ENTORNO INDUSTRIAL. ANÁLISIS DEL BINOMIO EFICIENCIA-EMISIONES DE CO_2

Autor
María Javierre Moragues

Dirigido por
Dr. Emilio Martín Gallardo

Madrid
Enero 2025

María Javierre Moragues, declara bajo su responsabilidad, que el Proyecto con título **USO DE TÉCNICAS DE TRANSFER LEARNING EN UN ENTORNO INDUSTRIAL. ANÁLISIS DEL BINOMIO EFICIENCIA-EMISIONES DE CO_2** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2024/25 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: ..... Fecha: 17 / Enero / 2025

Autoriza la entrega:

El Director del Proyecto

Dr. Emilio Martín Gallardo

Fdo.: ..... Fecha: 17 / Enero / 2025

V. B. del Coordinador de Proyectos

Nombre del Coordinador

Fdo.: ..... Fecha: ..... / ..... / .....

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor Doña María Javierre Moragues **DECLARA** ser el titular de los derechos de propiedad intelectual de la obra: Uso de técnicas de transfer learning para garantizar la seguridad en un entorno industrial, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, los derechos de digitalización, de archivo, de reproducción, de distribución y de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

(a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.

(c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.

(d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.

(e) Asignar por defecto a estos trabajos una licencia Creative Commons.

(f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente*).

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

(a) Que la Universidad identifique claramente su nombre como autor de la misma

(b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

(c) Solicitar la retirada de la obra del repositorio por causa justificada.

(d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

(a) El autor se compromete a:

(b) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

(c) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

(d) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

(e) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 17 de Enero de 2025

ACEPTA

Fdo.: . . . . . .

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster Universitario en Big Data

USO DE TÉCNICAS DE TRANSFER LEARNING EN UN ENTORNO INDUSTRIAL. ANÁLISIS DEL BINOMIO EFICIENCIA-EMISIONES DE CO_2

Autor
María Javierre Moragues

Dirigido por
Dr. Emilio Martín Gallardo

Madrid
Enero 2025

Resumen

Hoy en día, y especialmente en el último año, la salud y la seguridad de las personas han cobrado especial importancia a nivel global. Esta preocupación se extiende al ámbito laboral, donde las condiciones de trabajo, los riesgos de cada puesto o las medidas de seguridad impuestas tienen un papel muy relevante en los temas mencionados. En entornos de tipo industrial dichos aspectos adquieren una gran importancia. Este escenario requiere de medidas más estrictas que garanticen la seguridad de los trabajadores, ya que los riesgos son más elevados debido al uso de maquinaria de elevada potencia que puede causar accidentes graves o incluso la muerte del trabajador. En este contexto es importante controlar tanto los procesos como a las personas que trabajan en ellos, ya que estas últimas son un factor importante a tener en cuenta para poder garantizar la seguridad.

La digitalización y los avances tecnológicos permiten utilizar técnicas basadas en inteligencia artificial que ayuden al desarrollo de herramientas que contribuyan a garantizar la seguridad. El objetivo es beneficiarse de los últimos avances para mejorar las medidas de seguridad a la vez que se busca que tenga un impacto reducido en la huella de carbono, continuando con el compromiso de REPSOL con las 0 emisiones.

Este proyecto ha hecho uso de técnicas de visión artificial para detectar objetos en imágenes. Concretamente, se ha utilizado un primer conjunto de datos que contiene imágenes de trabajadores en un entorno industrial con el objetivo de detectar equipos de protección personal y contribuir a minimizar los accidentes asociados a las actividades laborales. En el contexto actual, se ha utilizado la técnica de transfer learning, que consiste en hacer uso de redes neuronales previamente entrenadas, como YOLOv5 y YOLO11, y adaptarlas a un nuevo conjunto de datos. Este estudio pretender comparar el desempeño de YOLOv5, un modelo bien establecido, con YOLO11, un modelo de detección de objetos de última generación, para analizar los compromisos entre la precisión, los requisitos computacionales, emisiones de CO_2 y la escalabilidad de los datos. Al evaluar ambos modelos en un mismo conjunto de datos, se busca determinar si las mejoras incrementales en precisión de los modelos modernos justifican sus mayores demandas de recursos en aplicaciones prácticas, especialmente en entornos con restricciones de recursos. Con ello se pretenden conseguir los siguientes objetivos: reducir la huella de carbono de los modelos

y proporcionar una ventaja competitiva a las empresas mediante el uso de soluciones de inteligencia artificial, reduciendo el tiempo de entrenamiento, coste y el *time to market*. Para evaluar el comportamiento de las redes seleccionadas en diferentes escenarios, se ha utilizado un segundo conjunto de datos compuesto por imágenes de defectos presentes en acero laminado en caliente.

Para poder utilizar con éxito la técnica de la transferencia de aprendizaje hay que tener en cuenta, en primer lugar, el tamaño del conjunto de datos y la similitud entre las clases con las que se ha entrenado el modelo con anterioridad y las del nuevo conjunto de datos. Como se ha podido observar en los experimentos, de esto último depende en gran medida el rendimiento del modelo ya que, como se esperaba desde un principio, el primer conjunto de datos produce métricas mucho mejores que el segundo. Esto puede deberse a varias razones o incluso a una combinación de ellas. En primer lugar, sin llegar a tener clases en común con MS COCO2017, las clases del primer conjunto de datos son más similares a las que se han usado para entrenar tanto YOLOv5 como YOLO11, mientras que las del segundo conjunto son vastamente diferentes. En segundo lugar, el conjunto de datos de defectos en acero laminado en caliente no sólo contiene un número limitado de imágenes sino que, como consecuencia, no cuenta con un número suficientemente alto de instancias etiquetadas de cada clase como para asegurar un buen entrenamiento.

El primer lugar, era importante comparar el rendimiento de los modelos cuando se entrena el modelo entero sobre el nuevo conjunto de datos o cuando se hace sobre un modelo con capas congeladas. Ambos son tipos de transferencia de aprendizaje pero uno implica menos computación que el otro, concretamente al congelar capas se congelan pesos por lo que hay menos parámetros que calcular. Con este método se ha ahorrado tanto en tiempo de computación como en uso de recursos del sistema, con la desventaja de que el rendimiento también se ha visto reducido. Sin embargo, dado que la reducción en rendimiento ha sido pequeña, y aunque la reducción en tiempo de computación también lo ha sido, se ha decidido proceder a buscar los parámetros óptimos utilizando los modelos congelando ciertas capas. Todo esto es generalizable a los cuatro modelos diferentes de YOLOv5 y los cinco de YOLO11, donde la arquitectura se mantiene constante dentro de cada modelo, pero varía el número de parámetros según el mismo, y para los cuales se ha comprobado que, en general, a mayor número de parámetros, mayor rendimiento del modelo.

Uno de los parámetros configurables es el número de épocas (número de iteraciones) durante las que se va a ejecutar el modelo y se ha comprobado que éste depende completamente del conjunto de datos. También se han realizado experimentos de manera que se ponderen clases, es decir, que de una época a otra cobran más peso aquellas imágenes que contengan objetos para los cuales el modelo produce menor rendimiento. Se ha comprobado que esta técnica es útil cuando todavía queda margen de mejora en el modelo, si éste ha alcanzado en torno al máximo aprendizaje posible entonces la técnica no mejora los resultados. Una de las mayores ventajas de la familia de modelos YOLO es que está programada de manera que el tamaño de batch no afecte al rendimiento

del modelo. Por esta razón los resultados de los experimentos son reproducibles sin importar el hardware, una tarjeta gráfica menos potente requerirá de un tamaño de batch menor para funcionar pero se terminará obteniendo el mismo resultado, a costa de un tiempo de computación mayor que si se usara una tarjeta gráfica más potente en la cual se pueda usar un tamaño de batch mayor. Esto es especialmente interesante desde el punto de vista de las emisiones de CO_2. Un aspecto que habitualmente se pasa por alto es el coste medioambiental que supone el entrenamiento y la inferencia de modelos de inteligencia artificial. Si bien los modelos desarrollados por profesionales de machine y deep learning típicamente no son grandes productores ni emisores de CO_2, las grandes empresas acumulan no sólo un número elevado de modelos sino de mayor tamaño. Por ello dichas empresas, como REPSOL que está comprometida con las 0 emisiones, tienen que buscar nuevas formas de beneficiarse de los últimos avances tecnológicos a la par que buscar técnicas cuya huella de carbono tenga un impacto reducido. En este contexto, se ha encontrado que un tamaño de batch 3 veces mayor supone una reducción a la mitad del tiempo de ejecución y una reducción del 66 % de las emisiones de CO_2. Por otro lado, se ha ejecutado un modelo entrenado con los pesos optimizados y ese mismo modelo inicializado con pesos aleatorios, con el objetivo de calcular el tiempo y las emisiones que hubieran supuesto optimizar la arquitectura de la red. Se ha podido observar que para que el modelo con los pesos aleatorios obtenga el mismo resultado de rendimiento se necesita 4 veces más tiempo de computación, lo que se traduce en que las emisiones equivalen a casi cuatro veces las del primer modelo. Utilizando el enfoque propuesto, no sólo se reduce la huella de carbono sino que se reduce el coste económico y, al reducirse los tiempos de entrenamiento, también se reduce el tiempo en que llega el modelo al mercado, lo que a su vez puede proporcionar ventajas competitivas frente a otras empresas. Por lo tanto, este escenario demuestra que al utilizar transfer learning se han logrado los objetivos mencionados anteriormente. Dado el volumen de modelos que se diseñan y entrenan al día de forma global, la reducción de emisiones de CO_2 entre un modelo con pesos optimizados y uno entrenado desde cero es significativa, por lo que buscar técnicas que permitan reducir la huella de carbono debería ser una prioridad para las grandes empresas.

Asimismo, se ha comparado el uso, en términos de eficiencia y energía, de un modelo de última generación frente a un modelo ya establecido en la detección de objetos. En este caso, la elección de una arquitectura más moderna dependerá de la necesidad de negocio de obtener métricas más exactas a cambio de un gasto mayor, ya que se ha calculado que YOLO11 consume un 21 % más de energía para una mejora en la detección del 5 %.

Por último, se ha calculado tanto el tiempo como las emisiones que se hubieran dado de no utilizar la técnica de transfer learning y haber creado la red desde cero. Este análisis implica que al hacer uso de una red preentrenada las emisiones han sido 213.750 y 285.000 veces menores para YOLOv5 y YOLO11 respectivamente , lo que confirma que el uso de esta técnica permite cumplir los objetivos propuestos para este proyecto. Con ello se ha confirmado que se puede hacer uso de los últimos avances tecnológicos, proporcionando ventajas competitivas en forma de aplicación de software de inteligencia artificial pero

acercándonos al compromiso con las 0 emisiones que persigue REPSOL.

Abstract

Nowadays, and especially in the last year, the health and safety of people have become particularly important worldwide. This concern is extended to the workplace, where working conditions, risks pertinent to each position or the security measures play a very relevant role in the aforementioned issues. In industrial-type environments, these aspects acquire great importance. This scenario requires stricter measures to guarantee the safety of the workers, since the risks are higher due to the use of highly potent machinery that can cause serious accidents and even the death of the worker. In this context, it is important to control both the processes and the people who work in them, since the latter are an important factor to take into account in order to guarantee workplace safety.

The digitalization and the technological advances make it possible to use techniques based on artificial intelligence that help to develop tools that contribute to guaranteeing safety in an industrial-type environment. The objective is to take advantage of the latest advances to improve security measures while working to have a reduced impact on the carbon footprint to follow REPSOL's commitment to zero emissions.

Computer vision techniques have been used in this project in order to detect objects within images. In particular, a first dataset containing images with workers in an industrial-type environment has been used with the objective of detecting personal protection equipment so as to reduce the number of accidents that are caused by people not following safety rules. In the current context, a technique named transfer learning has been used, which consists on making use of a pretrained network, in this case YOLOv5 and YOLO11, and adapting it to another dataset. This study aims to compare the performance of YOLOv5, a well-established model, with YOLO11, a state-of-the-art model in object detection, to analyze the trade-off between precision, computational requirements, CO_2 emissions and data scalability. When evaluating both models on the same dataset, the aim is to determine whether the improvements in accuracy of modern models justify their higher resource demands in practical applications, especially in resource-constrained environments. The aim is to achieve the following objectives: reduce the carbon footprint of the models and provide a competitive advantage for companies through the use of artificial intelligence solutions, reducing training time, costs and *time to market*. In order to evaluate the performance of the selected network in different situations a second dataset

has been used, which contains images of defects on hot-rolled steel.

To be able to use transfer learning with success, it is necessary to take into account, at first, the size of the dataset and the similarity between the classes the model has been trained on previously and those of the new set of data. As it has been observed in the experiments, the performance of the model greatly depends on the latter since, as it was expected from the beginning, much better metrics are obtained for the first dataset than for the second one. This can be due to several reasons or even a combination of them. Firstly, without having classes in common with MS COCO 2017, the classes of the first dataset are similar to those used to train YOLOv5 and YOLO11 while those of the second set are vastly different. Secondly, the hot-rolled steel dataset not only contains a limited number of images but, as a consequence, does not have a high enough number of labeled instances of each class to ensure a good training.

Before anything else it was important to compare the performance of the models when training the entire model on the new dataset or when training it on a model with frozen layers. Both are types of transfer learning but one requires less computation, in particular the second one as when freezing layers weights are frozen so there are less parameters to calculate. This method has reduced both computing time and use of the system's resources, with the disadvantage that performance has also been reduced. However, given that the reduction in performance has been small, and although the reduction in computing time has also been small, it has been decided to search the optimal parameters using the models by freezing certain layers. This can be generalized to the four different YOLOv5 models and the five of YOLO11, where the model architecture remains the same for each model, but the number of parameters in each varies, for which, in general, the greater the number of parameters, the higher the performance of the model.

One of the parameters that can be tuned is the number or epochs (number of iterations) for which the model is going to be executed for and it has been confirmed that it is completely dataset dependent. Experiments have also been carried out in such a way that classes are weighted, that is, from one epoch to another images containing objects for which the model produces lower performance gain more weight. It has been proven that this technique is useful when there is still room for improvement in the model, if it has reached a plateau in learning then the technique does not improve the results. One of the biggest advantages of the YOLO family models is that they are programmed in such a way that the batch size does not affect the performance of the model. For this reason the results of the experiments are reproducible regardless of the hardware, a less powerful GPU will require a smaller batch size to work but will end up obtaining the same result at the cost of a greater computing time than if a more powerful GPU was used with a larger batch size. This is especially interesting from the point of view of CO_2 emissions. One aspect that is often overlooked is the environmental cost that arises when training and making inference with artificial intelligence models. Although the models developed by machine and deep learning professionals are not typically large producers

or emitters of CO2, large companies accumulate not only a large number of models but also larger ones. Therefore, these companies, such as REPSOL that is committed to zero emissions, have to look for ways of taking advantage of the latest technological advances while reducing the carbon footprint. In this context, it has been found that increasing the batch size by 3 entails a reduction by half of the training time and a 66 % in CO_2 emissions. On the other hand, a model using the optimized weights has been run and then the same model has been executed but with a random weight initialization, with the aim of calculating the time and emissions that optimizing the network's architecture would have entailed. It has been observed that the model with the random weights needs 4 times more computing time than when using optimized weights, which means that the emissions are equivalent to almost four times those of the first model. In addition, not only is the carbon footprint reduced but the economic cost is reduced and, by reducing training times, the time it takes for the model to reach the market is also reduced, which in turn can provide competitive advantages over other companies. Thus, this scenario shows that by using transfer learning the aforementioned objectives of the project have been achieved. Given the volume of models that are designed and trained daily worldwide, the reduction in CO_2 emissions between a model with optimized weights and one trained from scratch is significant, so looking for techniques that allow for the reduction of the carbon footprint should be a priority for large companies.

Furthermore, a comparison has been made between the efficiency and energy usage of a state-of-the-art model and that of a well-established one in object detection. In this case, the choice of a more modern architecture will depend on the business need to achieve more accurate metrics in exchange for higher costs, as it has been calculated that YOLO11 consumes 21 % more energy for a 5 % improvement in detection.

Lastly, the training time and carbon emissions that would have resulted in the case of not using transfer learning and instead creating the network from scratch have been calculated. This analysis implies that by using transfer learning the emissions have been 503.616 times smaller, which again confirms that the use of this technique enables the achievement of the objectives of the project. It has been confirmed that companies can make use of the latest advances, providing competitive advantages through the use of artificial intelligence but getting nearer to REPSOL's commitment with zero emissions.

Unless you try to do something beyond what you have already mastered, you will never grow.
Ralph Waldo Emerson

Agradecimientos

En primer lugar me gustaría agradecer al Dr. Emilio Martín Gallardo, mi tutor y director de este proyecto, el apoyo y la confianza puesta en mi. Gracias por enseñarme, por guiarme pero siempre dejando espacio para mis ideas y por la paciencia para responder mis mil y una preguntas, especialmente cuando son sobre detalles pequeños.

A la Dra. Julia Díaz, mi jefa dentro de REPSOL, gracias por hacer de las prácticas una excelente experiencia, por todo lo aprendido y en especial por tus consejos, tanto profesionales como personales.

A ambos, gracias por darme la libertad de elegir un trabajo de fin de máster que me ilusionara y que me acerca un poco más a lo que quiero hacer en mi carrera profesional.

A D. Carlos Morrás Ruiz-Falcó, por estar siempre dispuesto a echar una mano.

Y, por supuesto, agradecer a mi familia el apoyo que siempre me ha brindado. A mis padres y a mi hermana, gracias por los ánimos cuando a mí me han faltado , por ayudarme y empujarme a hacer lo que me gusta.

María Javierre Moragues

Madrid

17 de Enero, 2025

Índice

Índice de figuras

Índice de tablas

Acrónimos

ICAI	Instituto Católico de Artes e Industrias
IA	Inteligencia Artificial
ANN	Artificial Neural Networks
GMDH	Group method of data handling
CNN	Convolutional Neural Networks
RCNN	Region Based CNN
SPPNet	Spatial Pyramid Pooling Networks
YOLO	You Only Look Once
SSD	Single Shot MultiBox Detector
NIPS	Neural Information Processing Systems
C2PSA	(Cross Stage Partial with Spatial Attention)
SGD	Stochastic Gradient Descent
ADAM	Adaptive Moment Estimation
AUC	Area Under the Curve
AP	Average Precision
mAP	mean Average Precision
NLP	Natural Language Processing
SOTA	State of the art

Capítulo 1

Introducción

Hoy en día, la salud y la seguridad de las personas se han convertido en temas de especial interés y preocupación a nivel global. Éstos se extienden al ámbito del trabajo, donde se incluyen temas como riesgos y condiciones del entorno de trabajo o las medidas de seguridad del mismo. Según el Ministerio de Trabajo, en España, concretamente en el 2023, se produjeron más de 1.000.000 de accidentes laborales, de los cuales más de 600 fueron mortales.[24].

Ambos aspectos cobran especial importancia en un entorno industrial, donde los riesgos son mucho mayores que en otros entornos y cuyas medidas de seguridad deben ser más estrictas. Muchos de los procesos industriales de hoy en día conllevan el uso de maquinaria de elevada potencia que causan accidentes graves. Por ello, anticiparse y prevenir riesgos son elementos clave para poder hacer de todos los procesos y operaciones algo seguro. Una vez se identifican y evalúan los riesgos se pueden diseñar barreras e identificar tareas críticas para poder garantizar la seguridad de los trabajadores.

Sin embargo, no es suficiente con gestionar la seguridad de los procesos, también hay que gestionar a las personas que trabajan en ello. REPSOL es una de las mayores empresas del sector energético, comprometida con la transición energética y el desarrollo sostenible de la sociedad. Con siete refinerías en total persigue la ambición "cero accidentes" y apuesta por trabajar las actitudes y comportamientos de las personas para reducir los accidentes causados por factores humanos. Dicho compromiso se ve reflejado en las palabras del Consejero Delegado de REPSOL, Josu Jon Imaz.

"Sea cual sea el puesto o ubicación geográfica, todas las personas de nuestra compañía son responsables de su propia seguridad, así como de contribuir a la salud y desempeño ambiental individual y colectivo. En caso de conflicto, tienen la responsabilidad de elegir la seguridad, decisión que siempre estará apoyada por la Dirección".[25]

La seguridad es la prioridad número uno y por eso se trabaja a lo largo de todo el ciclo de vida de las actividades y procesos, identificando riesgos relevantes e innovando en herramientas y metodologías. Gracias a los avances tecnológicos y la digitalización de una gran cantidad de procesos se pueden crear nuevas herramientas que ayuden a mejorar las medidas de seguridad. Por ejemplo, se pueden usar técnicas de aprendizaje profundo[26;27] o visión artificial[28]. Esta última busca que los ordenadores sean capaces de interpretar imágenes de la misma forma que lo haría un ser humano. Sin embargo, esto plantea varios desafíos. El primero de ellos técnico, debido a la complejidad de desarrollar inteligencia artificial, y el segundo es medioambiental.Las técnicas de visión por ordenador han experimentado grandes avances en los últimos años, siendo capaz de superar incluso para ciertas tareas las capacidades cognitivas de los humanos[29]. Este aumento en las capacidades ha ido acompañado de un aumento en la necesidades de cómputo y del uso de energía en general. Un aspecto que a menudo pasa desapercibido, es la huella de CO_2 resultante de entrenar estos modelos de aprendizaje profundo tal y como muestra la tabla 1.1.

Consumption	**CO_2e (lbs)**
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model	
SOTA NLP model (tagging)	13
w/ tuning & experimentation	33,486
Transformer (large)	121
w/ neural architecture search	394,863

Tabla 1.1: Datos sobre la huella de carbono dejada por modelos de inteligencia artificial.[22]

En este contexto de digitalización y de compromiso con las 0 emisiones, REPSOL tiene que buscar nuevas formas de beneficiarse de los últimos avances tecnológicos que redundan en la seguridad de las personas y, a la vez, buscar técnicas cuya huella de carbono tenga un impacto reducido.

Por ello, este proyecto trata sobre la detección de objetos en entornos industriales para mejorar la seguridad. Concretamente se busca detectar en un conjunto de imágenes que proceden de un entorno industrial aquellas personas que llevan casos de seguridad y por otra parte aquellas que no lo llevan. A la vez, se busca utilizar una técnica, denominada transfer learning[30], que permita reducir la huella de carbono que dejan los algoritmos de inteligencia artificial, ya que uno de los objetivos de REPSOL es convertirse en una compañía de cero emisiones en 2050.

1.1. Motivación

Como ya se ha mencionado, la seguridad en el trabajo es un elemento crucial para cualquier empresa. En entornos industriales es aún más importante, ya que la falta de seguridad puede ocasionar accidentes de mayor gravedad que si se tratara de aquellos que pueden surgir en general en una oficina. Una de las motivaciones del proyecto es estudiar la viabilidad de técnicas de transfer learning para ayudar a mejorar la seguridad en un entorno industrial.

Desde otro punto de vista, transfer learning permite reducir el tiempo de computación del modelo, ya que la red neuronal ya ha sido entrenada y sólo hay que añadir una serie de capas para que se adapte a la nueva tarea y nuevo conjunto de datos. Esto es especialmente interesante desde el punto de vista de la huella de carbono que se produce al entrenar modelos de inteligencia artificial. Un modelo consume energía, ya que para ejecutarse depende tanto de software como de hardware. En general, este último consume energía las 24 horas del día, especialmente en modelos de deep learning. La huella de carbono aumenta con el número de horas de entrenamiento de un modelo, por lo que es importante diseñar los modelos de manera que sean lo más eficientes posibles en cuanto a energía se refiere. Esto implica diseñar el código de manera eficaz pero eficiente a la vez, evitando pasos y cálculos innecesarios. Por otra parte, al usar transfer learning se reduce el tiempo de computación, por lo que se reduce la huella de carbono de este proyecto.

1.2. Objetivos

Los objetivos que se persiguen con este proyecto son los siguientes:

- Reducir la huella de carbono de los modelos de machine learning
- Analizar el equilibrio entre innovación, escalabilidad de datos y eficiencia computacional en escenarios de negocio
- Reducir el coste y minimizar el time to market
- Proporcionar una ventaja competitiva a la empresa mediante el uso de software e inteligencia artificial

El primero de los objetivos ya ha sido mencionado en la motivación del proyecto. REPSOL es una empresa cuyo objetivo para el 2050 es convertirse en una compañía de cero emisiones, por lo que reducir al máximo posible la huella de carbono de los modelos de machine learning es una parte esencial para llegar a cumplir este objetivo.

Al comparar dos modelos de detección de objetos, uno consolidado y otro de última generación, se busca evaluar los compromisos en la evolución del Deep Learning. Los modelos modernos suelen requerir conjuntos de datos cada vez más grandes y más recursos de computación pero los resultados no siempre presentan mejoras sustanciales frente a modelos anteriores o similares.

Por otro lado, el uso de transfer learning reduce los tiempos de computación. Esto implica que también se reduce el periodo de tiempo que tarda un modelo de machine learning en ser puesto en producción una vez se ha comprobado su efectividad. Todo ello supone una reducción de costes para la empresa.

En 2013, el CEO de General Electric, Jeffrey Robert Immelt, comentó "Creemos que toda compañía industrial será una compañía de software"[31] y en 2019, el CEO de Microsoft afirmó que "Toda empresa es ahora una empresa de software.... Los avances en inteligencia artificial van a ser rápidos y frenéticos "[32]. Está claro que, en una era que cada vez es más digital, aquellos que avancen en software y soluciones de inteligencia artificial tendrán una ventaja competitiva importante sobre el resto. Esto se ve reflejado en las inversiones cada vez más fuertes que realizan las empresas en temas de digitalización, IoT (Internet Of Things, industria conectada)... REPSOL prevé seguir invirtiendo 150-160 millones de euros en proyectos de digitalización al año hasta 2025 debido al flujo de caja obtenido en años anteriores, donde éste supera a la inversión. Por eso este proyecto está enmarcado también dentro de este objetivo, seguir desarrollando soluciones de software que permitan a la empresa obtener una ventaja competitiva.

Capítulo 2

Estado de la cuestión

La tecnología está cada vez más presente en la vida diaria ,y para seguir el ritmo de las expectativas del consumidor, las empresas confían y desarrollan soluciones basadas en algoritmos que les proporcionen una ventaja competitiva. Durante las últimas décadas se han desarrollado soluciones tecnológicas cuyo objetivo es crear sistemas que puedan resolver una amplia variedad de tareas como, por ejemplo, reconocimiento automático de voz[33], coches autónomos[34] o reconocimiento facial[35]. Muchas de estas soluciones tienen como base Deep Learning[26], incluída la de este proyecto. Por ello en este capítulo se explican los avances más relevantes en la historia del Deep Learning hasta llegar a los métodos más actuales. Así mismo se exploran técnicas de visión artificial[28], en concreto detección de objetos[36;37].

Por último se explica la técnica de transfer learning[30] utilizada en este proyecto para hacer el proceso más eficiente en cuanto a tiempos de computación y contaminación por CO2.

2.1. Deep learning

Deep Learning, también conocido como aprendizaje profundo, es considerado una rama de la Inteligencia Artificial[38] que pretende, mediante redes neuronales, emular el funcionamiento del cerebro humano. Mediante múltiples capas de redes neuronales estos modelos pueden aprender una representación de los datos con múltiples niveles de abstracción que simulan el proceso de análisis y aprendizaje del cerebro. A pesar de que esta técnica se ha vuelto muy popular en las últimas décadas , los primeros avances se realizaron en los años 40.

En 1943, Walter Pitts y Warren McCulloch se basaron en la biología del cerebro

humano para crear un modelo que aproximara el comportamiento de las neuronas, desarrollando lo que se conoce como la neurona de McCulloch-Pitts[39]. Ésta estaba basada en un modelo lineal que toma varias entradas [X_1, X_2 ..., X_n] para las cuales se asignan ciertos pesos [W_1, W_2 ... W_n] y calcula una salida binaria (verdadero / falso) como muestra la ecuación 2.1. Dichos pesos no eran asignados por el modelo sino configurados a mano por McCulloch y Pitts.

$$f(x, w) = X_1W_1 + X_2W_2 + \cdots + X_nW_n \tag{2.1}$$

El siguiente avance significativo se produjo en 1957 cuando Frank Rosenblatt desarrolló el Perceptrón[40]. El objetivo era aprender de manera automática los pesos que en la la neurona de McCulloch-Pitts eran configurados a mano, si bien el perceptrón estaba destinado a ser una máquina eléctrica de clasificación de imágenes y no un programa de software. El hardware utilizaba potenciómetros para determinar los pesos y seguía el esquema de la figura 2.1. La desventaja principal fue que no pudo ser entrenado para reconocer otros tipos de patrones.

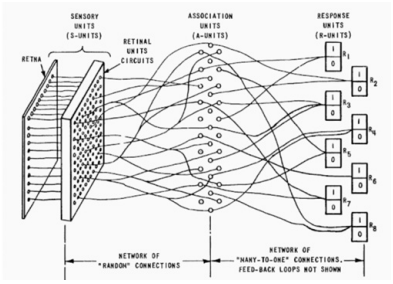


Figura 2.1: Funcionamiento del Perceptrón.[1]

No mucho más tarde, en 1960, Bernard Widrow y Edward Hoff desarrollaron ADALINE (Adaptive linear element)[2]. Parecida a un perceptrón, esta neurona podía adaptar los pesos durante la fase de aprendizaje basándose en la suma ponderada de las entradas, como muestra la figura 2.2.

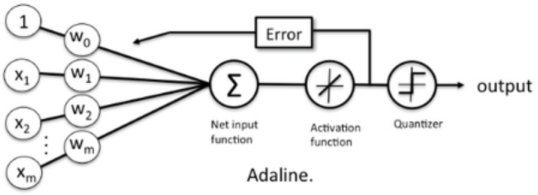


Figura 2.2: Arquitectura de Adaline.[2]

Un problema común a todas las técnicas que se han mencionado es que éstas no pueden resolver problemas de tipo XOR, lo que fue expuesto por Minsky y Papert en 1969[41].

Debido a este inconveniente y a la falta de potencia de los ordenadores el desarrollo de redes neuronales se paralizó durante un tiempo, lo que se conoce como el primer invierno de la IA.

El problema del XOR se arregló con la introducción de redes neuronales artificiales (ANN). Estas redes consisten en conectar capas formadas por varias neuronas cada una y en usar funciones de activación no lineales, lo que permitía procesamiento de señales en paralelo y distribuido en varias ramas de la red. Sin embargo, la primera red neuronal funcional había sido publicada por Ivakhnenko y Lapa en 1968 [42] como parte del método de agrupamiento para el manejo de datos (en inglés, Group method of data handling (GMDH)).

Las ANN resolvían el problema del XOR pero eran más difíciles de entrenar. Este problema se solucionaría más adelante usando propagación hacia atrás (en ingles, backpropagation). Un precursor de este método ya había sido formulado anteriormente por Henry J. Kelley [43] en 1960 pero la versión moderna no se formuló hasta 1970 por Seppo Linnainmaa [44]. Durante los siguientes años los avances estaban relacionados con la aplicación del método de backpropagation (e.g., Dreyfus, 1973 [45]; Werbo, 1975 [46] y 1982 [47]). No fue hasta 1986 cuando Rumelhart, Hinton y Williams [48] demostraron con un caso de procesamiento natural del lenguaje (NLP) que, con el método de backpropagation, las redes neuronales eran capaces de aprender representaciones interesantes en las capas ocultas. Poco después hubo avances (pooling y compartición de pesos) que hacían el entrenamiento más eficiente y que permitían extraer características locales.

En 1990 se publicó el primer trabajo sobre redes neuronales convolucionales [3] (en ingles, Convolutional Neural Networks (CNN)). Publicado por Yann LeCun et al. [3] entrenaron una CNN, cuya arquitectura se muestra en la figura 2.3 y demostraron que ésta agregaba características simples para conseguir cada vez características más complicadas y era capaz de usarlas para reconocer dígitos escritos a manos usando el conjunto de datos MNIST (números del 0-9 escritos a mano).

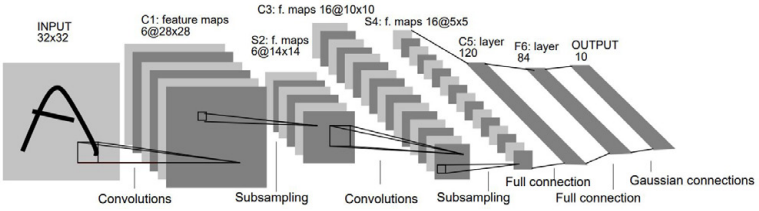


Figura 2.3: Arquitectura de la CNN propuesta por Yann LeCun et. al. [3]

Con la llegada de las CNNs surgió un nuevo problema: el desvanecimiento del gradiente [49] [50]. Este problema consistía en que las características que aprendían neuronas de las primeras capas no eran aprendidas por las capas más profundas ya que el gradiente se hacía cada vez más pequeño por lo que no les llegaba señal. Este problema fue solucionado unos años más tarde usando redes neuronales de tipo Long Short Term Memory

(LSTM)[51].

Durante finales de los años 90 y durante los primeros años de los 2000 se produjo el segundo invierno de la IA. Esto se debió en gran medida a la popularidad de otros algoritmos como las máquinas de vectores de soporte (en inglés, Support Vector Machines (SVM)) y Random Forest. Adicionalmente, el entrenamiento de redes neuronales tenía un coste elevado y era lento. En 2004, se presentó el uso de las GPU como solución viable a los problemas de entrenamiento de las redes[52], ya que permitía entrenamientos más rápidos al procesar más rápido las imágenes. A partir de ese momento se desarrollan numerosos avances (dropout[53], Rectified Linear Units(RELU)[54], Batch Normalization[55] y GAN[56] entre otros) para mejorar el rendimiento de las redes neuronales y solventar los nuevos problemas que iban surgiendo.

En 2009, un profesor de Stanford, Fei-Fei Li, lanzó ImageNet[57] , una base de datos gratuita con más de 14 millones de imágenes ya etiquetadas. Para 2011 la velocidad de las GPUs había aumentado significativamente, lo que facilitaba el entrenamiento de CNNs. AlexNet es un ejemplo de CNN que durante los años 2011-2012 ganó varias competiciones internacionales[58] usando RELU y dropout para mejorar el rendimiento. A medida que las redes neuronales han ido evolucionando su precisión ha ido en aumento, así como el número de parámetros con los que pueden trabajar. Esto se puede apreciar en la figura 2.4, donde se observa que existen redes neuronales que utilizan hasta 500 millones de parámetros. Sin embargo, también hay consecuencias negativas. Por un lado, debido al enorme número de parámetros es difícil usar dicha red en dispositivos con poca capacidad de cómputo. Por otro lado, aquellos dispositivos que sí tienen suficiente capacidad consumen más energía y esto tiene un impacto medioambiental.

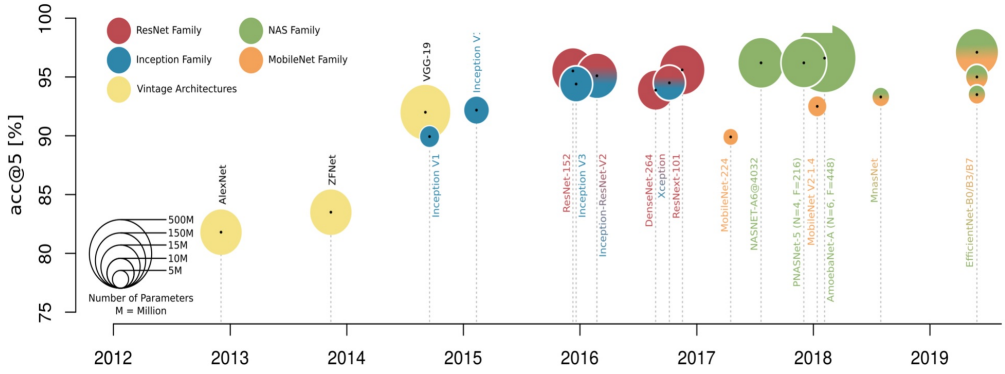


Figura 2.4: Evolución de la precisión y número de parámetros de las redes neuronales (2012-2019).[4]

Hasta este punto la mayoría de las estrategias de machine y deep learning se habían centrado en aprendizaje supervisado, es decir, el algoritmo aprende a mapear una entrada a una salida basándose en ejemplos de duplas entrada-salida. En 2012, Google

Brain publicó los resultados de un proyecto denominado "The Cat Experiment" [29], donde exploraban técnicas de aprendizaje no supervisado. En este tipo de aprendizaje la red neuronal es alimentada con datos sin etiquetar para que la propia red encuentra patrones recurrentes en ellos. Este experimento pretendía que la red aprendiera a identificar imágenes de gatos para lo cual se alimentó con 10 millones de imágenes procedentes de Youtube sin etiquetar. A pesar de mejorar en un 70 % los resultados de sus competidores, esta red sólo reconocía menos del 16 % de los objetos utilizados en el entrenamiento [29]. El aprendizaje no supervisado sigue siendo un objetivo a conseguir en el campo de deep learning. A pesar de estos desafíos, la introducción de la arquitectura de transformers [59], en 2017, ha permitido avances significativos en el deep learning. Los modelos de lenguaje grande (LLMs), como GPT-3 y GPT-4, han demostrado que es posible combinar técnicas de aprendizaje supervisado y no supervisado para alcanzar resultados sobresalientes en diversas tareas.

2.2. Visión Artificial

La visión artificial es un campo de la ciencia de computación que permite a los ordenadores ver, identificar y procesar de la misma manera que lo hace la visión humana. A pesar de que este campo se ha disparado recientemente (el gran éxito se produjo cuando AlexNet ganó ImageNet [58] en 2012) no es un campo de investigación nuevo, ya que los científicos de datos llevan desde antes de los años 60 buscando formas de programar un ordenador para que sea capaz de extraer información de las imágenes como lo hace un cerebro humano.

El primer avance significativo se produjo en 1957, cuando Russell Kirsch consiguió escanear y representar de manera digital la primera fotografía [60]. Tan sólo dos años después, en 1959, los neurofisiólogos David Hubel y Torsten Wiesel describieron las propiedades de la respuesta de las neuronas de la corteza visual en un gato. El experimento,descrito en su artículo “Receptive fields of single neurons in the cat’s striate cortex” [61], trataba de colocar electrodos en la corteza visual primaria de un gato para observar su actividad neuronal mientras le enseñaban diversas imágenes, tal y como muestra la figura 2.5. Descubrieron que éstas no se excitaban con la imagen en sí misma sino con el cambio de imagen, con el movimiento se creaba una sombra del borde de la diapositiva que contenía la imagen. Este experimento destacó que el procesamiento visual siempre empieza con estructuras simples como lo son los bordes.

El siguiente avance se produjo en 1963, cuando Larry Roberts publicó su tesis en el MIT (Massachusetts Institute of Technology). La idea tras la tesis era crear una representación 3D basada en imágenes de diferentes perspectivas 2D de un objeto, lo que se hacía transformando las imágenes en dibujos hechos con líneas tal y como muestra la figura 2.6, reduciendo el espacio visual a simples formas geométricas [6]. Roberts es considerado el padre de la visión artificial ya que el hecho de querer hacer una representación en 3D a

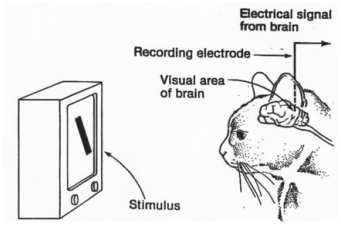


Figura 2.5: Esquema del experimento de Hubel y Wiesel.[5]

partir de imágenes en 2D diferencia la visión artificial de la visión de máquina (en inglés, Machine Vision), disciplina que se encarga de hacer que las máquinas sean capaces de 'ver' a través de la utilización de sensores.

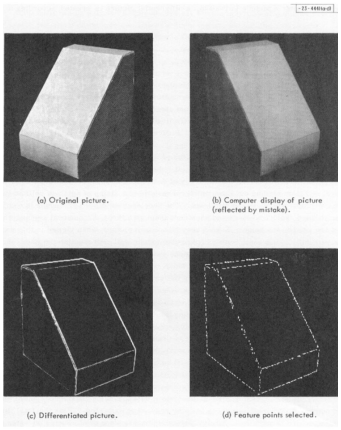


Figura 2.6: Reducción a simples formas geométricas según la tesis de Larry Roberts.[6]

Poco después, en 1966, Seymour Papert definió el proyecto "Summer Vision Project"[62] en el MIT , que pretendía hacer segmentación y reconocimiento de patrones en imágenes reales. Sin embargo,no se consiguieron resultados significativos debido a que el proyecto resultó ser demasiado complejo además de la falta de capacidad de computación de la época[63]. La investigación en este campo continuó siguiendo la dirección sugerida por Roberts. En 1971, David Huffman y Max Clowes, de manera independiente, publicaron un algoritmo para etiquetar líneas[63], que eran etiquetadas como cóncavas, convexas u ocluidas para más tarde ser usadas para distinguir las formas de los objetos.

En 1979 se produjo un logro importante de la mano de David Marr mediante la sugerencia de que la visión artificial debía ser aproximada desde un planteamiento ascendente. Según Marr la visión tenía una jerarquía[64] y empieza por características de bajo nivel (bordes, curvas, esquinas..) a partir de las cuales se construyen características de más alto nivel. Esto produjo que las investigaciones de los años 80 se centraran en la extracción y el procesamiento de características de bajo nivel.

Tan sólo un año después Kunihiko Fukushima, basándose en el trabajo de Hubel y Wiesel, modela una red neuronal diseñada para reconocer patrones en imágenes, denominada Neocognitron[7]. Esta red neuronal incluye muchas capas convolucionales, tal y como muestra la figura 2.7 y es considerada la primera red neuronal profunda. En la misma década se producen contribuciones al campo de la visión artificial basados en las matemáticas y la estadística, como el algoritmo para el cálculo de flujo de Lucas-Kanade[65] en 1981, el detector de bordes Canny[66] en 1986 o eigenface para reconocimiento facial en 1991[67].

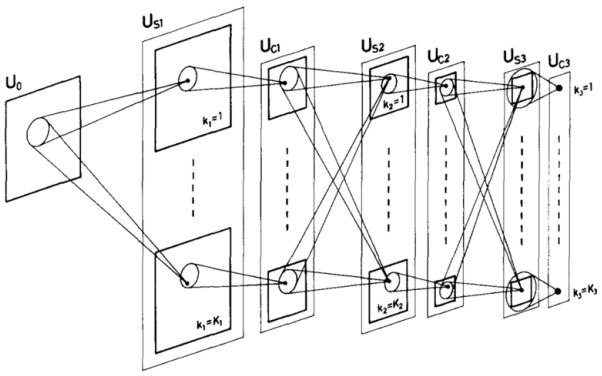


Figura 2.7: Capas interconectadas del Neocognitron de Fukushima.[7]

Ya en los 2000, concretamente en 2012, AlexNet gana el concurso anual de ImageNet[58] presentando la idea de que la profundidad de las redes neuronales es importante para conseguir resultados precisos. AlexNet tiene cinco capas convolucionales seguidas de tres capas conectadas completamente. Más tarde,la red neuronal de Microsoft, ResNet, consiguió mejores resultados de precisión utilizando 152 capas en su red[68].

Entre sus posibles aplicaciones se encuentra la clasificación de imágenes y la detección de objetos. La primera de ellas consiste en asignar una clase o categoría a una imagen (por ejemplo, perro o gato) mientras que la segunda consiste en detectar varias clases dentro de una misma imagen. Esto quiere decir que se puede detectar a un perro y a un gato dentro de la misma imagen y además dar sus posiciones dentro de la misma. La segunda de estas aplicaciones, detección de objetos, se explica en detalle en la siguiente sección por ser la utilizada para este proyecto.

2.2.1. Detección de objetos

La detección de objetos es una disciplina que trata de encontrar e identificar diferentes objetos en imágenes. Los avances generales en deep learning han permitido que esta aplicación del campo de la visión artificial avance hasta el punto de utilizarse para un número elevado de aplicaciones en el mundo real, como lo son los coches autónomos[34], videovigilancia o reconocimiento facial[35]. En los últimos años ha habido un aumento del

número de publicaciones relacionadas con la detección de objetos tal y como se muestra en la figura 2.8, por lo que es de esperar que el número de aplicaciones en el mundo real también aumente.

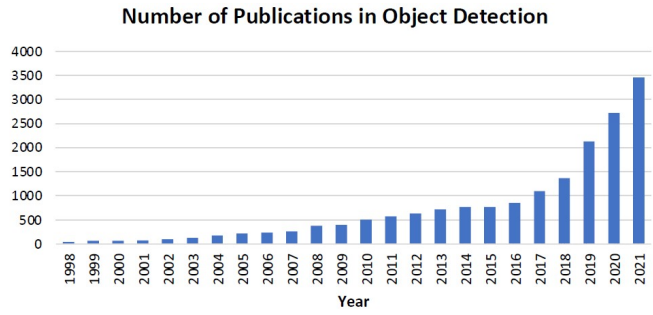


Figura 2.8: Aumento del número de publicaciones relacionadas con la detección de objetos (1998-2021).[8]

La detección de objetos se diferencia de la clasificación de imágenes en que la segunda conlleva la asignación de una etiqueta mientras que la primera implica dibujar una caja, denominada bounding box, que encuadre el objeto dentro de la imagen. Por consiguiente, la detección de objetos es una tarea más complicada que además combina también la parte de clasificación, ya que además de encuadrar el objeto de interés dentro de la imagen le asigna una etiqueta.

Entre los factores que han hecho posible la rápida evolución de técnicas de detección de objetos se encuentra el desarrollo de redes neuronales convolucionales y la potencia de computación. La mayoría de los detectores del estado del arte utilizan hoy en día redes neuronales profundas como backbone (parte de clasificación) y después una red de detección. Ambas se centran en extraer características de las imágenes o vídeo de entrada, pero la primera está centrada en la parte de clasificación y la segunda en la parte de localización del objeto.

De forma general el progreso en la detección de objetos se puede dividir en dos fases diferentes tal y como muestra la figura 2.9.

Para poder clasificar un objeto es necesario extraer sus características, que son las representaciones simplificadas de la imagen y contienen sólo la información más importante de ésta que permite distinguirla de otra imagen diferente.Encontramos una primera época en la cual, debido a la falta de recursos de computación o a la limitación de estos, las características de las imágenes eran extraídas de forma manual. Uno de los primeros detectores de la historia fue diseñado por Viola y Jones en 2001, con el que consiguieron detectar caras humanas[69]. El detector funcionaba usando una ventana móvil: la ventana se movía por todas las posiciones posibles dentro de la imagen para determinar si alguna de ellas contenía una cara humana. Este detector, que recibió el nombre de detector Viola-Jones en honor a sus creadores, ha sido mejorado con el paso de los años para

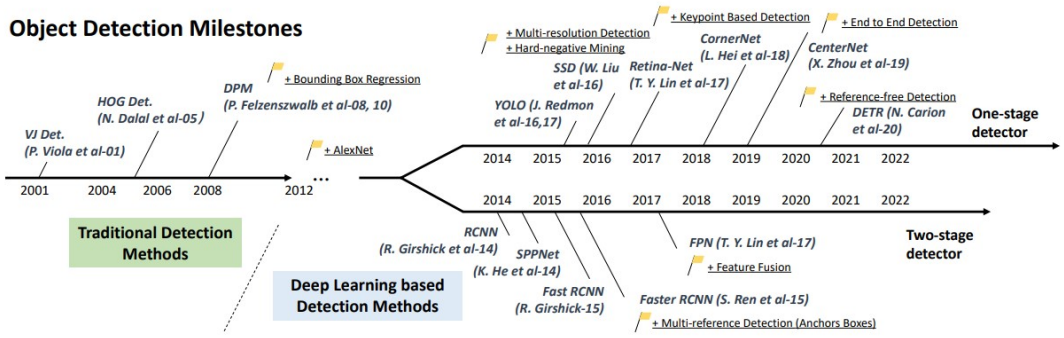


Figura 2.9: Línea temporal del progreso de la detección de objetos[8].

poder mejorar la velocidad de detección mediante la inclusión de técnicas como selección de características o detección en cascada[8].

El siguiente avance se produjo en 2005 con la propuesta de Dalal y Triggs de un nuevo descriptor de características, el histograma de gradientes orientados (en inglés, Histogram of Oriented Gradients (HOG))[9]. HOG se centra en la estructura o forma de un objeto y se diferencia de otros descriptores porque no sólo identifica si el píxel pertenece a un borde o no, sino que además es capaz de proporcionar la dirección, tal y como se aprecia en la figura 2.10. Esto es posible gracias a que extrae el gradiente y la orientación de este en los bordes. Comparado con otros detectores y dada la mejora en la extracción de características, HOG se ha usado como base para detección de objetos en múltiples ocasiones[70;71;8;72].

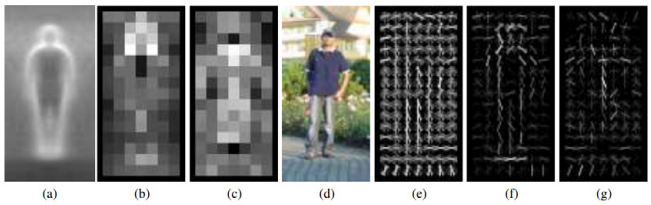


Figura 2.10: Ejemplo de los gradientes calculados con HOG en Histograms of Oriented Gradients for Human Detection[9].

En 2008 P.Felzenszwalg propuso el Deformable Part-based Model (DPM)[70] como una extensión natural de HOG que ganó la competición de detección de objetos tres años consecutivos (2007, 2008 y 2009). Más tarde R. Girshick añadió una serie de mejoras que permiten tratar con objetos del mundo real con más variaciones[71],[73]. Este detector sigue la filosofía de "divide y vencerás", por lo que trata la detección el entrenamiento como una proceso de aprendizaje en el que se descompone un objeto en partes elementales y la inferencia como una manera de ensamblar las diferentes partes de este[8]. A partir de 2010 los avances se ralentizan y el progreso consiste en mejorar sistemas que ya existen[8].

Sin embargo, a partir del 2012 nacen con éxito las CNNs[58], [74] que son redes capaces de aprender de manera robusta características de una imagen. A partir de este punto las características ya no son extraídas de forma manual, sino que se usan métodos de Deep learning. Dentro de esta segunda etapa encontramos dos tipos de técnicas de detección de objetos: detección en dos pasos y detección en un paso.

El mecanismo de uso de las CNNs es muy simple, se coge un clasificador (como VGGNet o Inception) y se convierte en un detector de objetos al deslizarlo por una imagen. Se usa una ventana deslizante y en cada una de ellas se ejecuta el clasificador para obtener una predicción de qué objeto se encuentra en esa zona. Al usar una ventana deslizante se obtienen cientos o miles de predicciones para una única imagen pero al final sólo permanecen aquellas que el clasificador ha dado más alta probabilidad. Este enfoque funciona pero a cambio es muy lento ya que hay que ejecutar el clasificador un número elevado de veces.

La detección en dos pasos se introdujo cuando R. Girshick et al. propusieron en 2014 las redes neuronales basadas en regiones (en ingles, Region Based CNN, RCNN)[10], que mejoraban en cierta medida el problema de las CNNs. Las RCNN funcionan de la siguiente manera:

1. El modelo extrae regiones de interés por búsqueda selectiva. Estas regiones de interés son posibles bounding boxes, es decir, son regiones donde es posible encontrar los objetos de interés.

2. Un clasificador procesa las regiones candidatas.

La búsqueda selectiva pasa la imagen por ventanas de diferente tamaño y para cada tamaño intenta agrupar píxeles adyacentes teniendo en cuenta textura, color o intensidad para poder identificar diferentes objetos. Esto se puede apreciar en la figura 2.11

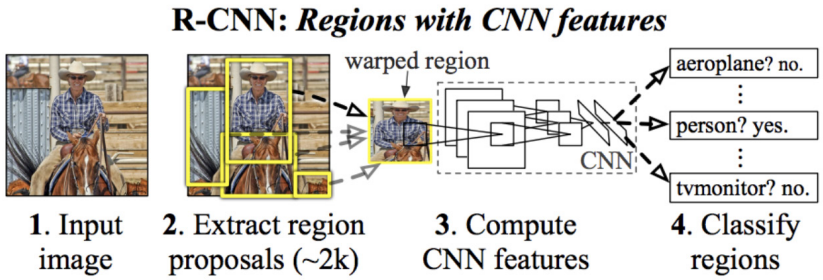


Figura 2.11: Funcionamiento de una RCNN[10].

La desventaja es la velocidad de detección ya que al tener un número elevado de regiones candidatas, que además se solapan, hacen de la detección un proceso lento

si bien el número es menor que usando sólo CNN porque ya no se hace en toda la imagen sino en regiones candidatas. Este problema se resolvió el mismo año con Spatial Pyramid Pooling Networks (SPPNet)[75]. Las CNNs previas requerían siempre que las imágenes de entrada tuvieran un tamaño fijo pero la introducción de una capa Spatial Pyramid Pooling (SPP) permitía extraer mapas de características de tamaño fijo aunque las imágenes de entrada fuera de distinto tamaño[8]. Estos mapas se calculan una única vez y de ellos se pueden extraer las regiones por lo que la velocidad de detección es más rápida[8]. Más tarde surgieron las Fast RCNN[76] y Faster RCNN[77].

El primer detector en un paso surgió en 2015 de la mano de R. Joseph. Denominado You Only Look Once (YOLO), proponía aplicar una única red a toda la imagen[8] tal y como muestra la figura 2.12. Ésta divide la imagen en regiones y predice bounding boxes y probabilidades para cada región de manera simultánea[11], lo cual aumenta en gran medida la velocidad de detección comparada con la técnica de dos pasos[8].Para ello divide la imagen en una cuadrícula de S x S celdas, donde S es un número entero. Cada una de estas celdas predice 5 bounding boxes diferentes y a la vez una clase para cada una de ellas. Además calcula un índice de confianza, es decir, calcula la probabilidad (0-1) de que la identificación del objeto es correcta. En la figura 2.12, en la imagen superior, se puede apreciar esto mismo ya que cuanto más alto el índice de confianza más gruesa la línea del rectángulo que encierra el objeto. El índice de confianza y la clase asignada al rectángulo se combinan para determinar la probabilidad de que dicho rectángulo contenga cierto objeto. En total hay S* S * 5 bounding boxes de las cuales un número elevado tendrán índices de confianza muy bajos que serán descartados, por lo que la predicción final sólo se compone de unas pocas bounding boxes; mostrada en la figura 2.12, concretamente en la imagen de la derecha .

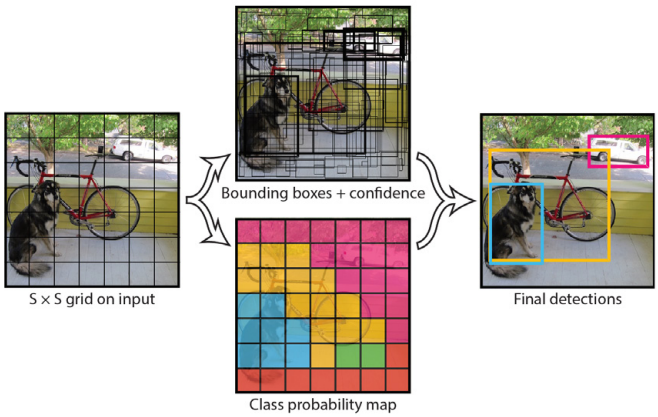


Figura 2.12: Sistema YOLO[11].

YOLO mejoró en gran medida la velocidad de detección[8] pero a cambio, comparado

con detectores en dos pasos, sufre de un descenso en precisión de la localización principalmente en objetos pequeños. En 2015 W. Liu et al. propusieron otro detector en un paso denominado Single Shot MultiBox Detector (SSD), cuya mayor contribución es la introducción de técnicas de multi-resolución y multi-referencia que aumentan la precisión de detección sobre todo en objetos pequeños[78]. Esto resolvía el problema de YOLO pero a cambio cuenta con una velocidad de detección más lenta[8].

En un principio, y a pesar de la rapidez y la simplicidad de los detectores de un paso, estos detectores producían resultados menos precisos comparados con los detectores en dos pasos. En 2017 T.-Y. Lin et al. propusieron RetinaNet[79], una red neuronal que usa una función de pérdida nueva, denominada focal loss, que resuelve el problema del desequilibrio de clases entre el primer plano y el fondo. Este problema era la causa de la menor precisión en los detectores de un paso y con la esta nueva función de pérdida consiguen una precisión comparable a la de los detectores de dos pasos[8].

En 2019 surgieron EfficientDet[12] y CenterNet[80], ambas redes siguieron el esquema de detección en un paso, con EfficientDet superando en precisión a muchas de las redes neuronales del momento como muestra la figura 2.13.

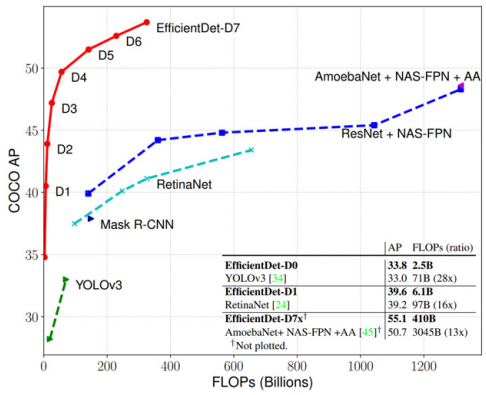


Figura 2.13: Precisión de EfficientDet vs. otras ANN del momento.[12].

A partir de la publicación de EfficientDet, el campo de la detección de objetos ha experimentado avances significativos, especialmente con la evolución de la familia de modelos YOLO y las arquitecturas basadas en transformers. Con el tiempo se han desarrollado más versiones de este primer detector hasta llegar a la versión 11, YOLO11[21]. Esta última, junto a la versión 5 (YOLOv5[15]), son explicadas en detalle en la sección 3.3 al ser las utilizadas en este proyecto. Por otro lado, en 2020 surgió DETR[81], un modelo basado en transformers que marcó un hito al cambiar el enfoque tradicional utilizado hasta el momento. Posteriormente, Deformable DETR[81] y Swin Transformer[82] abordaron desafíos como la eficiencia de entrenamiento y la detección de objetos de pequeño tamaño[81]. Los últimos avances han seguido utilizando transformers, como es el caso de DETRv2[83] en 2022 y RT-DETRv2[84] en 2024.

2.3. Deep Learning eficiente: Transfer Learning

Como se ha explicado en secciones anteriores, los avances tanto en hardware como en software han permitido que los modelos de deep learning sean cada vez más grandes al ser capaces de diseñar redes con más capas y calcular más parámetros, lo cual tiene consecuencias tanto positivas como negativas. Por un lado, como se ha visto en la figura 2.4, cuantos más parámetros tenga la red neuronal mejor precisión se consigue. Sin embargo, el número elevado de parámetros implica un tiempo mayor de computación, lo que a su vez implica hardware con gran capacidad de cómputo y consumo de energía. Así mismo, entrenar redes neuronales de gran tamaño conlleva la utilización de un conjunto de datos de gran tamaño, normalmente etiquetados, que puede ser difícil de conseguir por lo que el proceso es aún más lento. La técnica de la transferencia de aprendizaje (en inglés, transfer learning) pretende resolver estos problemas.

Transfer learning[30] es una técnica que se basa en usar una red neuronal que ha sido entrenada para una tarea concreta y reutilizarla en un conjunto de datos diferente. Las personas tenemos la capacidad intrínseca de transferir conocimiento de una tarea a otra, de ganar conocimiento sobre una determinada tarea y aplicarlo a otra similar. Por ejemplo, somos capaces de aprender a conducir un coche y utilizar esa base para aprender a conducir una moto. No aprendemos desde cero sino que aprendemos partiendo de una base conocida. Tradicionalmente los algoritmos de machine y deep learning han sido diseñados para trabajar aislados y resolver tareas concretas. Sin embargo, típicamente en cuanto cambian los datos se tiene que modificar el modelo. Con transfer learning se pretende utilizar el conocimiento adquirido en una tarea para resolver otra tarea similar. Esquemas de ambas situaciones se pueden encontrar en la figura 2.14.

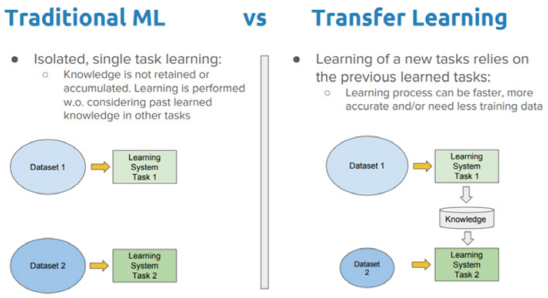


Figura 2.14: Machine learning tradicional vs. transferencia de aprendizaje[13]

En 2016, Andrew Ng,científico de datos asociado a Google Brain y Stanford entre otros, comentó en su charla 'Nuts and bolts of building AI applications using Deep Learning' en la conferencia NIPS (Neural Information Processing Systems) que, tras el éxito del aprendizaje supervisado, transfer learning sería siguiente el motor del éxito comercial de técnicas de machine learning.

Sin embargo, el concepto de la transferencia de aprendizaje no es nuevo. En 1976, Stevo Bozinovski y Ante Fulgosi publicaron un artículo sobre la posibilidad de emplear transfer learning en el entrenamiento de redes neuronales[85]. En él detallaban un modelo de transfer learning tanto desde una perspectiva matemática como geométrica. A partir de la charla "Learning to Learn: Knowledge Consolidation and Transfer in Inductive Systems" en NIPS en 1995 se empezaron a acuñar términos como Learning to Learn, Knowledge Consolidation o Inductive Transfer. Todos ellos perseguían el mismo objetivo: encontrar la manera de readaptar grandes redes neuronales a otras tareas sin necesidad de crear redes nuevas desde cero[13]. En 2016, Goodfellow et al en su libro "Deep learning" hicieron referencia a la técnica de transfer learning en un contexto de generalización :

"... situación dónde lo que se ha aprendido en un escenario se explota para mejorar la generalización en otro escenario" [27].

Las redes neuronales entrenadas en imágenes reales exhiben el mismo comportamiento ya mencionado en la sección 2.1 : en capas de bajo nivel aprenden características simples (bordes, formas...). Estas características no parecen ser específicas para un conjunto de datos concreto sino que parece que son generales y, por lo tanto, aplicables a diversos conjuntos de datos y tareas[86]. Las características, con el tiempo, se transforman de generales a específicas al llegar a la última capa de la red. Esto es con lo que juega transfer learning, busca reutilizar las características generales aprendidas por la red y reentrenar con el nuevo conjunto de datos únicamente aquellas capas que extraen características específicas.

Hay varias estrategias de transferencia de aprendizaje, mostradas en la figura 2.15, que se pueden aplicar según el dominio de los datos, la disponibilidad de los mismos o la tarea a realizar.

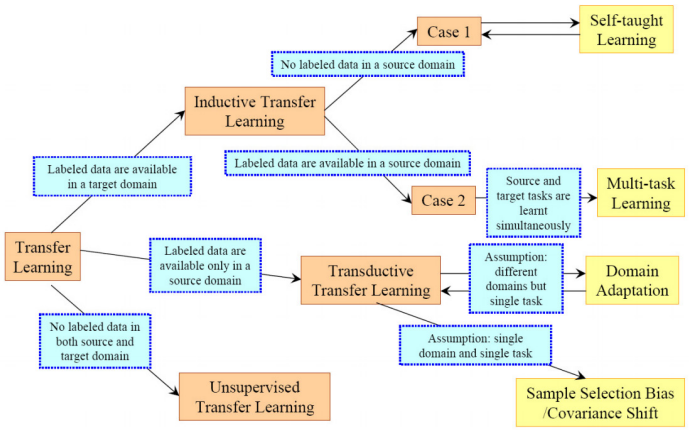


Figura 2.15: Diferentes estrategias de transferencia de aprendizaje[13]

Una vez escogida la estrategia que mejor convenga según el caso de uso surge la

pregunta de qué conocimiento se va a transferir. Éste se puede dividir en 4 casos [86]:

- Transferencia basada en instancias. Se usa cuando parte de los datos del conjunto de origen puede ser utilizadas para el nuevo conjunto de datos (conjunto objetivo) volviendo a ponderar los pesos.
- Transferencia de representación de características. Se usa cuando se pueden extraer buenas representaciones de las características del conjunto de origen, que posteriormente son codificadas en los mapas de características y utilizadas en el nuevo conjunto de datos.
- Transferencia de parámetros. Se usa cuando ambas tareas (la del origen y la del objetivo) son parecidas y por lo tanto comparten una serie de parámetros o distribuciones de hiperparámetros.
- Transferencia de conocimiento relacional. Asume que hay algún tipo de relación entre los datos de un conjunto y que dicha relación es similar entre el conjunto de origen y el objetivo.

Todas las estrategias que aparecen en la figura 2.15 pueden aproximarse a cualquier problema de machine learning mientras que en deep learning la estrategia utilizada es la del aprendizaje inductivo (en inglés, inductive learning). El objetivo es inferir un mapa a partir de una serie de muestras de entrenamiento [13]. Por ejemplo, en un caso de clasificación el algoritmo aprende un mapa entre características de entrada y etiquetas de salida. Dentro de esta estrategia hay 4 formas de hacer transfer learning, tal y como muestra la figura 2.16.

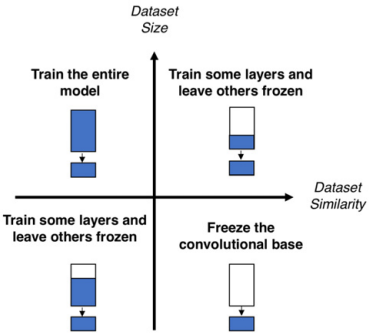


Figura 2.16: Formas de hacer transfer learning según el tamaño del conjunto de datos y la similitud de éste con el conjunto de datos original [14]. Los rectángulos de mayor tamaño representan la base convolucional de la red (extractora de características) mientras que los de menor tamaño representan el clasificador.

Dependiendo del tamaño del conjunto de datos nuevo y de la similitud de éste con el utilizado para entrenar la red neuronal se buscará usar primero una estrategia u otra de las

mostradas en la figura 2.16. Congelar capas hace referencia a congelar los pesos de dichas capas, es decir, al entrenar sobre el conjunto de datos nuevos dichos pesos no cambian. Cuanto más se parezcan los conjuntos de datos, más se parecerán las características extraídas por la red por lo que es de esperar que los pesos funcionen para ambos conjuntos. En este caso se congelarían capas de la red, desde unas pocas a todas las capas menos la parte del clasificador. Esta estrategia disminuye aún más el tiempo de computación ya que se reduce el número de parámetros a calcular. Por otro lado, cuanto menos se parezcan los conjuntos de datos menos se parecerán los pesos de la red, por lo que será necesario entrenar más capas de la red para que dichos pesos sean recalculados de manera que funcionen en el nuevo conjunto de datos.

Capítulo 3

Descripción del problema

Como se ha mencionado anteriormente, la seguridad en entornos de tipo industrial no sólo se ve afectada por la gestión de procesos sino también la gestión de las personas que trabajan en ellos. Los accidentes ocasionados por el factor humano son, en gran medida, debido a la violación de los protocolos de seguridad. Antes de los avances en computación la manera de asegurar el cumplimiento de las medidas de seguridad era analógica, una persona física debía estar pendiente de las cámaras de seguridad para comprobar, por ejemplo, que todos los trabajadores llevaban los equipos de protección individual. El avance de la computación, y de la inteligencia artificial en particular, ha permitido automatizar una gran cantidad de tareas y mejorar otras. Asimismo, la evolución de las técnicas de Deep Learning ha supuesto un aumento del rendimiento de los modelos así como de su eficacia y precisión. Esto afecta de manera positiva a las empresas que consiguen desarrollar soluciones basadas en software ya que éstas proporcionan ventajas competitivas sobre el resto de empresas. Sin embargo, y para que esto sea posible, también ha aumentado el número de parámetros a calcular en cada modelo, lo que a su vez requiere de ordenadores que tengan mayor capacidad de cómputo. Este aumento en la capacidad de computación supone un aumento en el tiempo necesario para entrenar modelos y, por lo tanto, un aumento en la energía total consumida. Desarrollar inteligencia artificial supone dos desafíos principales, siendo el primero de ellos técnico ya que desarrollar modelos de inteligencia artificial es una tarea compleja. El segundo reto consiste en desarrollar dichos modelos de manera que sean lo más eficientes posibles para reducir el consumo de energía. Dicha reducción no sólo supone una reducción en costes para la empresa sino que además supone reducir emisiones de CO_2.

Con este proyecto se busca encontrar una medida que pueda ayudar a mejorar las condiciones de seguridad en entornos industriales mientras se intentan abordar los desafíos mencionados. Para ello se utilizan técnicas de visión artificial con el objetivo de detectar diferentes objetos en imágenes y valorar la viabilidad del modelo para perfeccionar medidas de seguridad en un entorno industrial. Los objetos a detectar y su justificación

para este proyecto se detallan en los siguientes apartados. Asimismo se utiliza la técnica de transfer learning, ya que al utilizar una red neuronal que ha sido pre-entrenada se reduce el tiempo de computación. Esto implica una reducción en las emisiones de CO_2 comparado con si se hubiera modelado y entrenado una red neuronal nueva desde cero, ya que para entrenar redes neuronales se necesita una gran cantidad de datos y, por lo tanto, tiempo de entrenamiento. Además se espera que con esta técnica se llegue de manera más rápida a modelos con buen rendimiento ya que los pesos de la red neuronal ya están optimizados. El rendimiento dependerá en parte de la similitud que haya entre los objetos que se desean detectar en este proyecto y aquellos objetos que la red esté entrenada para detectar. Otros factores de los cuales dependerá el rendimiento del modelo serán los hiperparámetros del mismo.

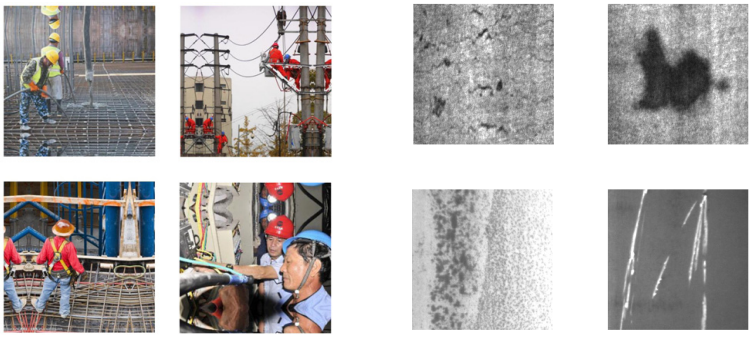
Este capítulo está dedicado a la descripción de los conjuntos de datos utilizados y el preprocesado necesario. De igual forma se detallan las redes neuronales utilizadas (YOLOv5 y YOLO11), además de la forma de entrenar el modelo y las métricas que han sido utilizadas para evaluar el mismo.

3.1. Conjunto de datos

Con el fin de probar la técnica de transfer learning y evaluar su posible implantación para mejorar medidas de seguridad se utilizan dos conjuntos de datos diferentes. Este proyecto usa aprendizaje supervisado, en el cual una red neuronal aprende a inferir un mapa entrada-salida a partir de ejemplos resueltos. Por ello es necesario partir de un conjunto de datos que contenga imágenes sobre las cuáles se quieren detectar los objetos y, a la vez, algo que indique dónde están los objetos que se quieren detectar dentro de dichas imágenes. Por lo tanto, ambos conjuntos se componen de los siguientes elementos:

- Imágenes
- Anotaciones

El primer conjunto de datos se compone en su mayoría de imágenes en distintos entornos de tipo industrial donde se necesita llevar casco de seguridad. En una menor parte contiene imágenes de personas en entornos no industriales donde no existe la necesidad de llevar elementos de protección personal. Estas imágenes han sido anotadas manualmente con la intención de detectar trabajadores y si estos llevan cascos de seguridad. El segundo contiene imágenes sobre seis tipos diferentes de defectos que se pueden dar en placas de acero laminado en caliente. La figura 3.1 muestra ejemplos de las imágenes que se pueden encontrar en ambos conjuntos.



(a) Conjunto #1 (b) Conjunto #2

Figura 3.1: Ejemplos de imágenes de los conjuntos de datos utilizados.

Por lo tanto, las categorías a detectar por el algoritmo para cada conjunto de datos son las siguientes:

- Conjunto 1 : persona, cabeza, casco
- Conjunto 2: rolled-in scale, patches, crazing, pitted surface, inclusion y scratches.

Ambos conjuntos de imágenes son públicos siendo el primero de ellos obtenido a través de Kaggle[87]. El segundo ha sido adquirido a través de la Northeastern University (NEU) y producido por Kechen Song y Yunhui Yan[88].

El objetivo de usar dos conjuntos de imágenes es poder evaluar el rendimiento del modelo para dos situaciones completamente diferentes. A priori, se esperan mejores resultados usando el primer conjunto de datos que el segundo. Esto se debe a que las clases del primer conjunto de imágenes son más similares a con las que se ha entrenado previamente la red neuronal utilizada en este proyecto. YOLOv5 ha sido entrenado con el conjunto de datos Microsoft Common Objects in Context (COCO)[89] versión de 2017 que contiene 80 clases diferentes. COCO 2017, entre otras muchas, contiene la clase "persona", que es una de las clases que contiene el primer conjunto de datos de este proyecto. Es de esperar que si esta clase está bien representada en el conjunto, es decir, aparece un número de veces suficiente, YOLOv5 y YOLO11 sean capaces de detectar dicha clase con una precisión alta ya que ambas redes ha sido previamente entrenadas para ello. Sin embargo, ninguna de las categorías presentes en COCO 2017 se asemejan a los defectos que se quieren detectar en el segundo conjunto por lo que se espera. El listado completo de las categorías contenidas en COCO 2017 (80 clases) y COCO original (91 clases) puede encontrarse en el apéndice A.

Con respecto a las anotaciones, son archivos que contienen información sobre los objetos de interés que se han encontrado en las imágenes, por lo que cada imagen tiene

su correspondiente archivo con los objetos etiquetados. Dichos archivos proporcionan información sobre el número de objetos de interés que hay en la imagen, el tipo de objeto que es cada uno de ellos e información sobre su posición relativa dentro de la imagen. Este proceso se denomina anotación y se debe llevar a cabo manualmente mediante la ayuda de un software de anotación, seleccionado de manera individual en cada imagen los objetos que se consideran de interés. Existen multitud de formatos en los que se pueden anotar imágenes y se distinguen en la manera de proporcionar la información, tanto en disposición como en contenido. El formato de las anotaciones para ambos conjuntos de datos es Pascal Visual Object Classes (VOC) [16]. Este formato está escrito en XML y está estructurado de la siguiente forma:

```
<annotation>
  <folder>Train</folder>
  <filename>01.png</filename>
  <path>/path/Train/01.png</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>224</width>
    <height>224</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>36</name>
    <pose>Frontal</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <occluded>0</occluded>
    <bndbox>
      <xmin>90</xmin>
      <xmax>190</xmax>
      <ymin>54</ymin>
      <ymax>70</ymax>
    </bndbox>
  </object>
</annotation>
```

En las primeras líneas contiene información relativa a la imagen tal y como nombre de la misma, carpeta donde está guardada o tamaño, entre otras. A partir de <object>contiene información relativa al objeto hasta que finaliza con </object>y esta estructura se puede repetir tantas veces como objetos de interés se encuentren en la imagen. Pascal VOC proporciona la posición del objeto dentro de la imagen mediante la posición en píxeles de los cuatro vértices del rectángulo (xmin, xmax, ymin y ymax). El parámetro <difficult>hace referencia a la dificultad con la que se encuentra ese objeto en la

imagen. Si el valor del parámetro es igual a 1 se dice que el objeto es difícil de apreciar en la imagen , es decir, o se ve muy pequeño, o está borroso, o se aprecia mal, etc. Este objeto tendrá unas coordenadas porque la persona que lo ha etiquetado ha sido capaz de encontrarlo pero estos objetos pueden afectar de manera negativa al modelo. Por ello, todos los objetos con valor 1 en este parámetro no serán tenidos en cuenta. Esto se tendrá en cuenta en el preprocesamiento de los datos, que se explica con detalle en la siguiente sección.

3.2. Análisis exploratorio y preprocesado

Esta sección está dedicada al análisis exploratorio de ambos conjuntos de datos así como el preprocesado realizado en ambos, como la conversión del formato de las anotaciones al apropiado para los modelos YOLO, entre otros.

En primer lugar se analizan para ambos conjuntos los tamaños de las imágenes, ya que de manera general se usan imágenes del mismo tamaño en el entrenamiento y se hace inferencia en imágenes del mismo tamaño que las del entrenamiento. El tamaño de la imagen influye en el tamaño de los objetos que contiene la imagen, cuanta menos resolución tenga la imagen más pequeños aparecerán los objetos. Esta es la razón por la cual se entrena y se hace inferencia en imágenes del mismo tamaño: la red aprende a detectar los objetos a un tamaño determinado por lo que, si se cambia el tamaño de detección a uno muy diferente al del entrenamiento, la red puede no ser capaz de detectar de forma adecuada los objetos.

Para el primer conjunto de datos las imágenes (5000 en total) oscilan en resolución, presentando los tamaños de la tabla 3.1. A pesar de contener imágenes de distinta resolución esta diferencia es mínima ya que tan sólo es de un píxel, por lo que se puede considerar que las imágenes son del mismo tamaño.

Ancho	**Alto**	**Número de imágenes**
415	415	23
415	416	325
416	415	2465
416	416	2197

Tabla 3.1: Recuento de número de imágenes por tamaño para el primer conjunto de datos.

Por el contrario, las imágenes del segundo conjunto (1800 imágenes) presentan todas la misma resolución: 200 x 200 píxeles. Esto implica que cada imagen de este segundo conjunto de datos contiene 4 veces menos píxeles que las del primer conjunto. Es de esperar que para el mismo número de imágenes la red tarde menos en procesar el segundo conjunto de datos que el primero.

Lo siguiente a evaluar es el número de objetos etiquetados que existen por clase en cada conjunto. Antes de llevar a cabo esta evaluación se separa cada conjunto de datos en tres particiones diferentes:

- Train : partición que contiene el 70 % de los datos originales y que es utilizada para entrenar la red.
- Validation : partición que contiene el 20 % de los datos originales y que es utilizada para validar la elección de hiperparámetros.
- Test : partición que contiene el 10 % de los datos originales y que es utilizada para hacer inferencia y probar el modelo.

La figura 3.2 representa el número total de objetos en las imágenes de cada partición. Asimismo se busca comprobar que las clases están balanceadas entre particiones, ya que de lo contrario la red podría aprender a detectar objetos de una clase que esté sobre representada en el conjunto de entrenamiento y no aprender el resto de clases. Tal y como se aprecia en la figura 3.2, las clases están balanceadas entre particiones pero, sin embargo, las clases no están balanceadas dentro de cada partición. Como se puede observar hay más representación de la clase "helmet"que de las otras dos clases. Más llamativo aún es la infra representación de la clase "persona": hay 24 veces menos instancias de esta clase que de la clase mayoritaria ("helmet") y 8 veces menos que la clase "head".

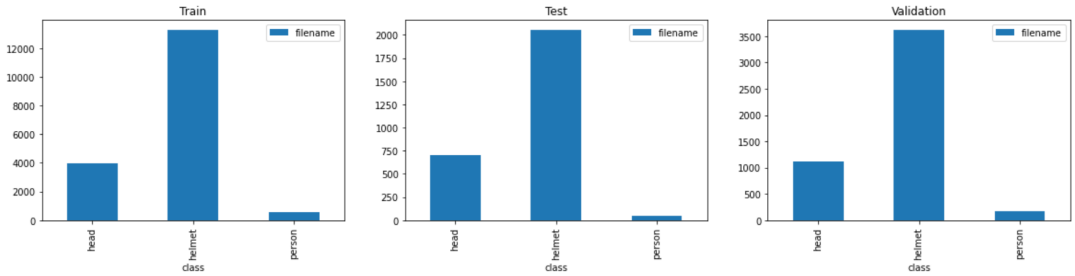


Figura 3.2: Número de objetos etiquetados por clase del primer conjunto de datos.

Esto podría causar problemas al entrenar ya que es posible que la red no aprenda a detectar de forma precisa la clase "persona"debido al número bajo de instancias de la misma que existen en la partición de entrenamiento. Las posibles soluciones a este problema se detallan en la sección 3.4 'Entrenamiento'.

De la misma manera se realiza este análisis en el segundo conjunto de datos. En este caso, el conjunto de datos estaba ya dividido en train y test por lo que, para obtener el conjunto de validación, se ha dividido el conjunto de entrenamiento. Como se puede observar en la figura 3.3 los conjuntos de entrenamiento y validación tienen la misma distribución mientras que la de test es ligeramente diferente, teniendo más instancias de

los 3 primeros defectos que los otros conjuntos. Sin embargo, las clases en general están más balanceadas que en el primer conjunto de datos.

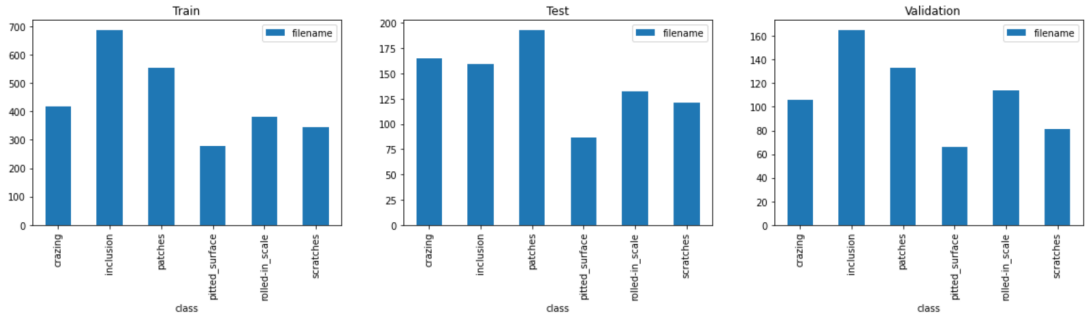


Figura 3.3: Número de objetos etiquetados por clase del segundo conjunto de datos.

En lo referente al preprocesado de los datos, destaca la conversión del formato de las anotaciones. Como se ha mencionado anteriormente ambos conjuntos presentan las anotaciones en formato Pascal VOC (archivo de tipo XML). Sin embargo el formato requerido por YOLOv5 y YOLO11 es parecido al formato de otras versiones de la familia YOLO: YOLO Darknet. A cada imagen le corresponde un único archivo en formato txt en el que cada línea corresponde a un objeto y deben cumplir lo siguiente:

- Tener formato <object-class><x><y><width><height>
- Las clases se representan con números empezando desde 0
- <x>e <y>hacen referencia a las posiciones que marcan el centro de la imagen.
- Las coordenadas de la bounding box deben estar normalizadas (valores entre 0 y 1).

La figura 3.4 presenta un ejemplo gráfico de las anotaciones en formato YOLO Darknet. La línea que corresponde a un objeto de la clase "persona"de dicha imagen (en este caso a Zinedine Zidane) sería la siguiente: 0 0.48 0.63 0.69 0.7

Teniendo en cuenta estas premisas se ha desarrollado un código que lee cada anotación en formato Pascal VOC y la transforma a un formato compatible con el entorno Darknet que maneja YOLO. Las anotaciones en formato Pascal VOC contienen más información que la que utiliza YOLO y, concretamente, contienen información sobre lo complicado que es encontrar un objeto en dicha imagen. Aquellas objetos que en formato Pascal VOC tuvieran por valor un 1 en el parámetro <difficult>no han sido tenidos en cuenta y por lo tanto no aparecen en las anotaciones con formato adecuado para la red. Como se ha mencionado antes, estos objetos son excluidos de la evaluación debido a que pueden afectar de manera negativa al modelo al no ser ejemplos claros de su clase.

Figura 3.4: Ejemplo gráfico de anotaciones compatibles con el entorno Darknet utilizado por la familia de redes YOLO[15].

La figura 3.5 muestra ejemplos de objetos, marcados con líneas discontinuas, marcados como difíciles.

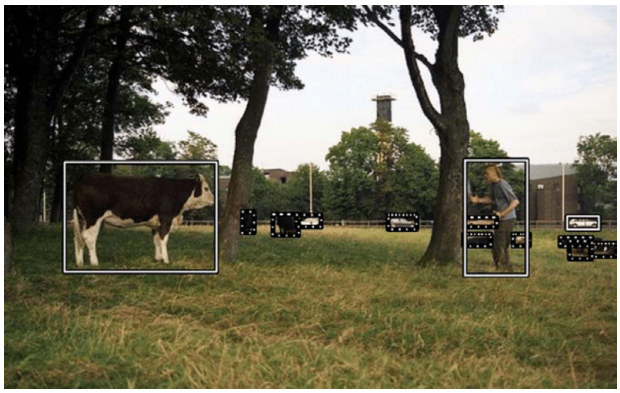


Figura 3.5: Ejemplo de objetos marcados como difíciles de encontrar (objetos encapsulados dentro de líneas discontinuas[16].

Por último, se configuran los archivos .yaml, necesarios para que la red pueda localizar y organizar correctamente los datos y parámetros necesarios para su funcionamiento. Estos archivos contienen información sobre el directorio que contiene las imágenes, número de clases y nombre de éstas. A continuación se muestra el ejemplo de este archivo para el primer conjunto de datos.

```
train: /..../train
val: /..../valid
nc: 3
names: ['helmet', 'person', 'head']
```

3.3. Redes preentrenadas : YOLOv5 y YOLO11

En la sección 2.2.1 'Detección de objetos' ya se ha visto que YOLO es una familia de redes neuronales que pertenecen a los detectores de objetos en un paso. La versión original de YOLO[11] fue desarrollada por Joseph Redmon en 2015 en un entorno nuevo denominado Darknet. Darknet es un entorno de desarrollo escrito en lenguajes de bajo nivel (como C) altamente flexible y con el cual se han podido desarrollar detectores de alta precisión en tiempo real como lo son la familia de redes YOLO. La primera versión detectaba a una velocidad de 45 fotogramas por segundo y cambió el paradigma de la detección de objetos. Con la segunda versión, YOLOv2, Redmon y Ali Farhadi introdujeron mejoras como la normalización de la capa de entrada de la imagen mediante la alteración de las funciones de activación o la posibilidad de utilizar imágenes de mayor resolución (de 224 x 224 a 448 x 448)[90]. Como la anterior, la tercera versión se construyó sobre los modelos anteriores con mejoras mediante la adición de conexiones a las capas del backbone o haciendo predicciones en 3 niveles de distinta granularidad para mejorar la precisión en la detección de objetos de pequeño tamaño[91]. En 2018 YOLOv3 era 3 veces más rápida que SSD y 4 veces más rápida que RetinaNet[92].

A partir de YOLOv3, Joseph Redmon abandonó el mundo de la visión artificial por cuestiones éticas[93] por lo que las versiones subsecuentes han sido desarrolladas por otras personas. En 2020, un nuevo grupo de desarrolladores, entre los que se encuentra Alexey Bochoknovskiy quién desarrolló YOLO para Windows en la primera versión[92], publicaron YOLOv4[94]. Esta nueva versión es dos veces más rápida que la versión contemporánea de EfficientDet a la vez que consigue una precisión muy similar[94].

YOLOv5[15] aparece tan sólo unos días más tarde que YOLOv4 de la mano de Glenn Jocher, quién desarrolló la extensión de YOLOv3 a Pytorch. Dicha extensión era muy utilizada para transferir los pesos desdes el entorno Darknet a Pytorch por aquellos desarrolladores que deseaban desplegar sus modelos y ponerlos en producción[95]. Una de sus ventajas fue la velocidad de detección. Comparada con las diferentes versiones del detector EfficientDet de Google, YOLOv5 mostraba una velocidad de detección bastante superior mientras alcanzaba una precisión comparable tal y como muestra la figura 3.6

YOLOv5 fue la primera red neuronal de la familia en estar escrita en el entorno de Pytorch en vez de Darknet. Esto supuso una gran mejora ya que facilita la transición de pesos al formato ONXX o CoreML de iOS. Asimismo, al estar escrito en PyTorch sólo requiere de la instalación de pocas librerías de Python.

Los modelos de la familia YOLO han seguido evolucionando buscando una mayor precisión y eficiencia. YOLOv6[96] y YOLOv7[97] mejoraron la optimización del entrenamiento y la detección de objetos pequeños. YOLOv8[98] y YOLOv9[99] adoptaron técnicas avanzadas de aprendizaje profundo, manteniendo un equilibrio entre velocidad y precisión. YOLOv10, introducido en 2024, eliminaba la necesidad de Non-Maximum Sup-

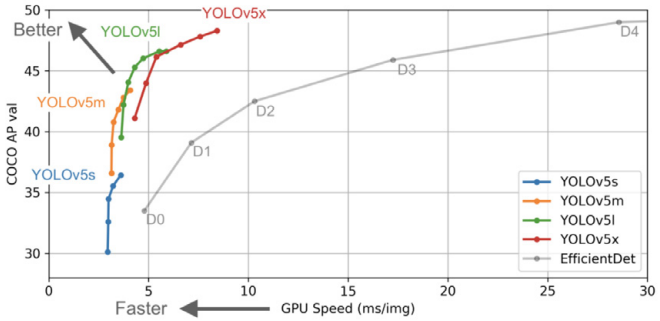


Figura 3.6: Comparación de YOLOv5 con EfficientDet[15].

pression (NMS). YOLOv11[23], el modelo más reciente y también introducido en 2024, mejora la precisión y velocidad de detección de objetos, utilizando un 22 % menos de parámetros que YOLOv8m[23].

En lo referente a la red, la figura 3.7 muestra la arquitectura de YOLOv5 mientras que la figura 3.8 muestra la referente a YOLO11.

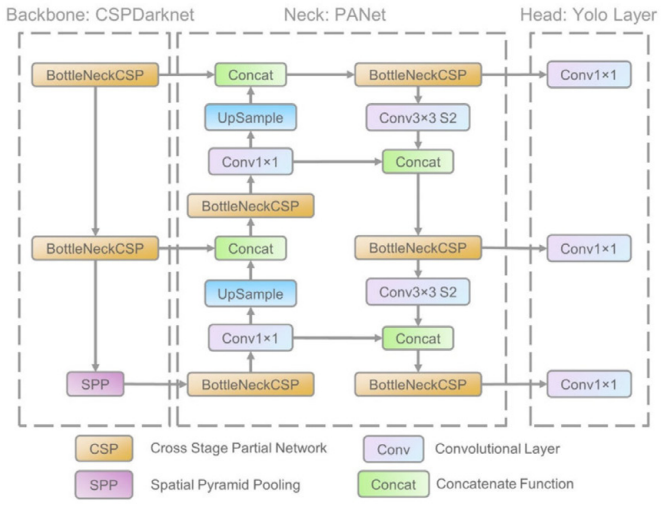


Figura 3.7: Arquitectura de YOLOv5[17].

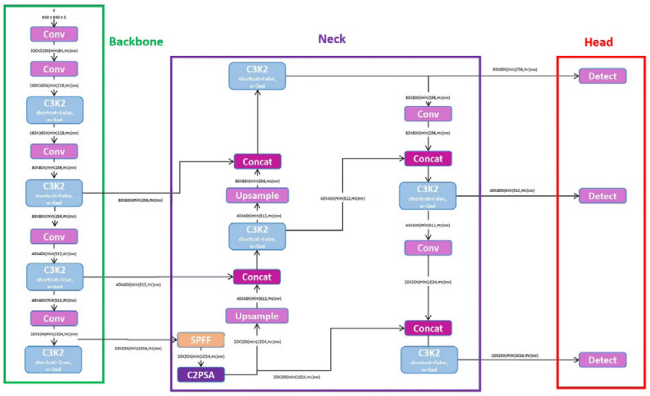


Figura 3.8: Arquitectura de YOLO11.

3.4. Entrenamiento

Se entiende por entrenamiento el proceso por el cual una red neuronal aprende mediante el procesamiento de ejemplos, cada uno de los cuales contiene una 'entrada' y un 'resultado' para que ésta infiera un mapa que conecte ambos. Este proceso es iterativo: los ejemplos son alimentados varias veces a la red de manera que ajusta los pesos de las neuronas cada vez. Dentro de este proceso hay una serie de parámetros que se pueden ajustar y de los cuáles depende en cierta medida la precisión final del modelo. Los parámetros para este proyecto se encuentran en la siguiente lista:

- Tipo de transfer learning según se ha visto en la sección 2.3.
- Optimizador de la función de pérdida. La función de pérdida es utilizada por la red para evaluar si una solución candidata es buena por lo que optimizar esta función es crucial para obtener buenos resultados.
- Épocas. Se definen como épocas el número de veces que se pasa el conjunto de datos entero por la red.
- Tamaño de Batch. Número de muestras que se propagan cada vez a través de la red neuronal.
- Pesos iniciales. Se refiere a los pesos iniciales de la neurona, que pueden ser iniciados de manera aleatoria o con algún conocimiento previo que se ajuste mejor a la situación.
- Otros hiperparámetros. Estos serán explicados en la siguiente sección.

Dadas las clases incluidas en el conjunto de datos MS COCO 2017, a priori, se espera que ambas redes produzcan mejores resultados con el primer conjunto de datos (cabeza, casco, persona) que con el segundo (defectos en acero). Esto se debe a que las clases del primer conjunto son más similares a las de MS COCO 2017 que al segundo. Sin embargo, ninguna de las clases de los conjuntos está contenida en las clases con las que se ha entrenado la red. Por ello se decide entrenar ambos conjuntos tanto entrenando el modelo entero como entrenando el modelo pero congelando capas de la red. Esta segunda opción implica que se realizarán menos cálculos, ya que los pesos de las capas congelados no se recalcularán, por lo que se espera que tenga un tiempo de computación menor y , por lo tanto, se reduzca la emisión de CO_2 comparada con la primera opción. La desventaja de congelar capas es que los pesos de dichas capas no serán optimizados para el conjunto de datos nuevos por lo que la precisión es menor utilizando esta técnica que entrenando todo el modelo.

Por otra parte YOLOv5 permite la implementación de dos optimizadores: Stochastic Gradient Descent (SGD) y Adaptive Moment Estimation (ADAM) . En este caso se han probado ambos para determinar cuál es el adecuado para el caso de uso. YOLO11 contiene varios optimizadores más pero se ha optado por la opción "auto", que selecciona automáticamente el más óptimo para la configuración.

En cuanto al tamaño del batch y número de épocas se han hecho diversos experimentos con diferentes valores para encontrar los óptimos. En lo relativo a los pesos, se han utilizado aquellos optimizados por YOLOv5 y por YOLO11. Asimismo también se ha hecho una prueba inicializando los pesos de manera aleatoria e intentando obtener la misma precisión que se obtiene utilizando pesos optimizados (ajustando épocas, tamaño de batch...). El objetivo es estimar y comparar de forma efectiva la reducción de CO_2 que se produce al utilizar los pesos optimizados en una red preentrenada en vez de entrenar y optimizar los pesos partiendo de valores aleatorios.

Por otra parte, cada modelo de la familia YOLO presenta modelos diferentes en tamaño pero con la misma arquitectura general. La diferencia entre los modelos recae en el número de parámetros internos (pesos y bias) lo que implica que el modelo con menos parámetros tiene menos neuronas por capa que el siguiente más grande tal y como se observa en el ejmplo de la figura 3.9.

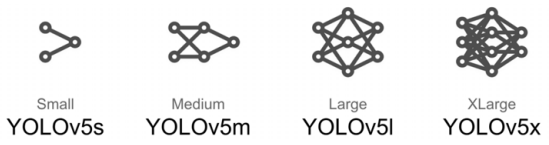


Figura 3.9: Modelos de YOLOv5[15]

La tabla 3.2 resume los cuatro modelos para YOLOv5, los cinco para YOLO11 y sus características. Como se puede observar, cuanto menor es el número de parámetros más rápida es la detección pero menor es su precisión (columna mAP).

Modelo	Tamaño (px)	mAPval 50-95	Velocidad CPU ONNX	Velocidad T4 TensorRT10	Parámetros (M)	FLOPs (B)
YOLOv5						
YOLOv5s	640	37,4	79,8 ± 0,9	2,2 ± 0,0	7,2	16,5
YOLOv5m	640	45,3	162,0 ± 1,5	3,8 ± 0,1	21,2	49,0
YOLOv5l	640	49,0	220,0 ± 1,0	6,1 ± 0,1	40,0	107.0
YOLOv5x	640	50,7	444,0 ± 5,0	10,4 ± 0,2	86,7	205,7
YOLO11						
YOLO11n	640	39,5	56,1 ± 0,8	1,5 ± 0,0	2,6	6,5
YOLO11s	640	47,0	90,0 ± 1,2	2,5 ± 0,0	9,4	21,5
YOLO11m	640	51,5	183,2 ± 2,0	4,7 ± 0,1	20,1	68,0
YOLO11l	640	53,4	238,6 ± 1,4	6,2 ± 0,1	25,3	86,9
YOLO11x	640	54,7	462,8 ± 6,7	11,3 ± 0,2	56,9	194,9

Tabla 3.2: Características de los diferentes tamaños de modelo para YOLOv5 y YOLO11[15][23].

Los resultados de los experimentos descritos aquí se encuentran en el capítulo 4.

3.4.1. Hiperparámetros: algoritmo genético

Los hiperparámetros son variables que determinan de qué manera va a ser entrenada la red neuronal. El ajuste de hiperparámetros hace referencia a encontrar la mejor combinación de estos que permita maximizar el rendimiento del modelo. Los hiperparámetros utilizados en los experimentos se pueden consultar en el apéndice B, ya que son los recomendados por defecto[15].

En general se pueden estimar de forma manual, es decir, fijando valores y experimentando para encontrar los valores óptimos (como se ha hecho para las épocas o el tamaño de batch) o bien de manera automática, utilizando algoritmos de optimización . Una posible opción es utilizar un algoritmo genético[100] para el ajuste.

Un algoritmo genético es una búsqueda heurística que se apoya en elementos inspirados por la biología del cuerpo humano, A través de un proceso iterativo se hace evolucionar a una población (conjunto de soluciones candidatas) mediante mutaciones, cruces entre individuos o selección. Esta selección determina qué individuos (cada una de las soluciones candidatas) son mejores según un criterio y por lo tanto sobreviven en la siguiente generación (iteración) o, por el contrario, cuáles son descartados al ser menos aptos. La evolución empieza con una población de individuos generados de manera aleatoria. En cada generación, se calcula la función de aptitud (en inglés, fitness) para determinar cómo de "buena.[es] dicha solución. Una proporción de los mejores individuos

sufrirán mutaciones o cruces para formar una nueva población. El proceso termina cuando se ha alcanzado el número de generaciones configurado o cuando el valor de la función de aptitud es suficientemente bueno.

Tras experimentar con diferentes valores de épocas, tamaño de batch, tipo de transfer learning y optimizador se han seleccionado los valores con mejor resultado para cada conjunto. Sobre ese escenario base se ha ejecutado el algoritmo genético con el fin de ajustar los hiperparámetros. Los parámetros del genético se detallan a continuación.

- 60 generaciones para el primer conjunto de datos y 300 generaciones para el segundo
- La función de aptitud, función a maximizar, es una combinación ponderada de métricas. mAP@0.5 contribuye un 10 % mientras que mAP@0.5:0.95 contribuye el resto. Estas métricas son explicadas en detalle en la sección 3.5
- Probabilidad de mutación del 90 %
- Probabilidad de cruce del 4 %

Dado que este proceso es intensivo en energía, sólo se ha realizado para el modelo YOLOv5 en ambos conjuntos. Los resultados de la ejecución de ambos genéticos se encuentran explicados en el capítulo 4.

3.5. Métricas de evaluación

En esta sección se explican las métricas utilizadas para evaluar el rendimiento de los diversos modelos entrenados. Además, se presenta la herramienta de visualización utilizada para monitorizar los modelos, automatizarlos y visualizar el rendimiento.

Con el fin de entender las métricas presentadas más adelante, se definen primero una serie de conceptos importantes: Intersección sobre la Unión (Intersection Over Union, IoU) [18] y los posibles resultados de un problema de clasificación.

La primera de ellas, la IoU, es una medida basada en el índice Jaccard que evalúa el solapamiento entre dos bounding boxes. Para ello compara la bounding box que se toma como verdadera, es decir, la que proviene de las anotaciones de las imágenes hechas manualmente, con la bounding box que predice el algoritmo. El valor de la intersección viene dado por el solapamiento de las distintas áreas dividido entre el área total de ambas, tal y como se muestra en la figura 3.10.

Por otro lado, en un problema de clasificación se pueden dar cuatro situaciones diferentes en cuanto a los resultados que devuelve el algoritmo. Éstas se ven reflejadas en la siguiente matriz de confusión (figura 3.11) y son explicadas a continuación.

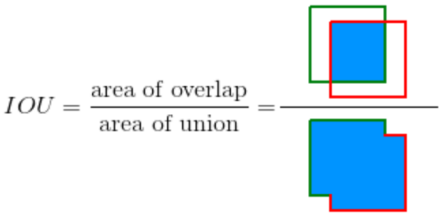


Figura 3.10: Intersection Over Union. [18]

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figura 3.11: Matriz de confusión. [19]

- **Verdadero Positivo** (True Positive, TP). Se considera TP a una detección correcta del objeto donde IOU >umbral.
- **Falso Positivo** (False Negative, FP). Se considera FP a una detección incorrecta, donde IOU <umbral
- **Falso Negativo** (False Negative, FN). Se considera FN cuando hay un objeto no detectado
- **Verdadero Negativo** (True Negative, TN).

Teniendo en cuenta estos cuatro tipos de resultados se pueden calcular dos métricas que se usan en la evaluación de modelos: precisión y recuperación (en inglés, recall). Éstas son explicadas en la siguiente sección.

3.5.1. Métricas

Las métricas utilizadas para evaluar y comparar los distintos modelos en este proyecto son las siguientes :

- Recuperación [101]
- Precisión [101]
- mean Average Precision (mAP) [101]: mAP@0.5 y mAP@0.5:0.95

- Utilización de GPU

Las ecuaciones 3.1 y 3.2 definen las métricas de precisión y recuperación, utilizadas para la evaluación de modelos de clasificación.

$$Precision = \frac{VerdaderosPositivos}{ResultadosReales} = \frac{VerdaderosPositivos}{VerdaderosPositivos + FalsosPositivos} \quad (3.1)$$

$$Recuperacion = \frac{VerdaderosPositivos}{ResultadosPrediccin} = \frac{VerdaderosPositivos}{VerdaderosPositivos + FalsosNegativos} \quad (3.2)$$

Como se puede observar, la precisión es una medida de la relevancia de los resultados, es decir, indica la proporción de identificaciones positivas que ha llevado a cabo el modelo. Por otro lado, la recuperación es una medida que indica el porcentaje de resultados relevantes que han sido clasificados correctamente por el el algoritmo, es decir, indica la proporción de positivos reales que han sido identificados correctamente. En modelos donde existe un umbral de confianza que se pueda configurar se puede hacer un intercambio entre la precisión y la recuperación. Por ejemplo, se puede configurar un umbral de confianza alto para que el modelo produzca resultados muy precisos a expensas de reducir la cobertura del modelo.

Ambas medidas se pueden expresar de manera conjunta en el mismo gráfico, de forma que aparece una nueva medida, el valor F1, que permite encontrar el valor óptimo del umbral de confianza que produce los valores más altos de precisión y recuperación. La figura 3.12 muestra la gráfica precisión-recuperación (izquierda). El área bajo la curva (en inglés, Area Under the Curve (AUC)), representada en la gráfica intermedia, es una medida de la eficacia de un sistema. Sin embargo, si en vez de integrar el área bajo la curva se calcula una media ponderada de la precisión a cada umbral usando el valor de la recuperación como peso, se obtiene la precisión ponderada (en inglés, Average Precision (AP)).

En problemas de detección de objetos el umbral mencionado viene dado por el valor de la intersección sobre la unión. Este valor es configurable según lo que considere el desarrollador del proyecto. Para cada una de clases que contenga el conjunto de datos, se calcula la curva precisión-recuperación para dicho umbral y su precisión ponderada (AP). Por último, se hace la media de las AP entre clases, calculando lo que se conoce como media de la precisión ponderadas (en inglés, mean Average Precision (mAP)).

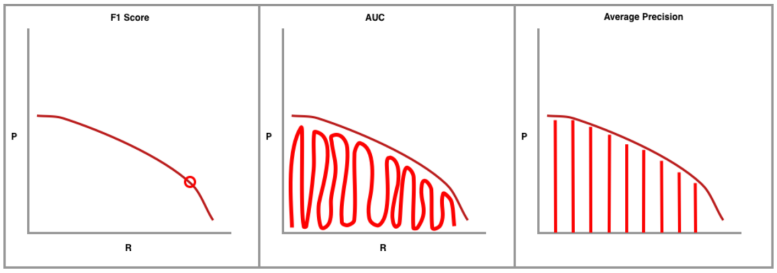


Figura 3.12: Gráficas del valor F1, AUC y AP[20].

Como práctica habitual se utiliza mAP@0.5, lo que indica que es la media ponderada con un IoU del 50 %. Asimismo, se suelen usar mAP@0.5:0.95, que indica que se han utilizado varios umbrales (empezando en 0.5 hasta llegar a 0.95 con una paso de 0.05). Para este caso se hace una curva precisión-recuperación para cada umbral, se calcula AP para cada umbral por clase, se calcula la media para cada clase tal y como muestra la figura 3.13. Por último se calcula la media entre clases para llegar a un mAP general.

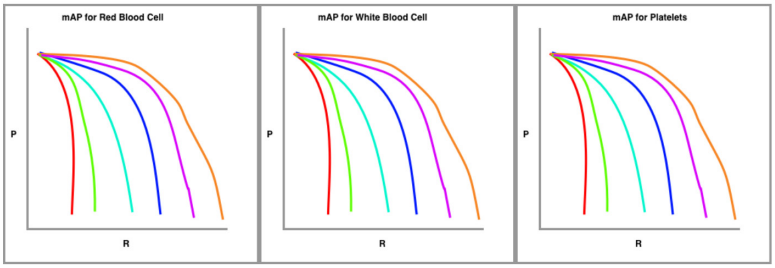


Figura 3.13: mAP para distintos IoUs (cada línea representa un umbral diferente) y distintas clases[20].

Hasta este punto las métricas explicadas están relacionadas con el rendimiento del modelo. Sin embargo, una parte importante del proyecto recae en la búsqueda de técnicas que permitan reducir la huella de carbono de los algoritmos de inteligencia artificial. Por ello es crucial monitorizar en casos de uso reales el uso del sistema que hace un modelo con el fin de optimizar recursos, evaluar eficiencia energética o el desgaste del sistema y valorar el impacto medioambiental del modelo. Por consiguiente, en cada uno de los experimentos se ha llevado a cabo la monitorización del uso de la GPU que incluye, entre otros, porcentaje de uso en el tiempo, temperatura o potencia utilizada (W).

3.5.2. Weights & Biases

A día de hoy, el uso de técnicas de Machine y Deep Learning se está convirtiendo en una herramienta importante que impulsa el valor de negocio de las empresas. A cambio, tiene una serie de problemas operacionales. En su artículo "Machine Learning: The High

Interest Credit Card of Technical Debt', Google describe los problemas de poner en producción modelos de Machine Learning de manera indiscriminada [102] . Es fundamental poder monitorizar los experimentos y los modelos para poder evaluar su funcionamiento y las condiciones en que lo hacen.

Con el fin de monitorizar y comparar de manera sistemática los resultados de cada modelo se ha utilizado una herramienta de visualización, Weights & Biases [103]. Esta herramienta es fácilmente integrable en cualquier script de Python y permite obtener métricas en tiempo real, logs de la terminal y estadísticas del sistema utilizado para ejecutar los experimentos, entre otros. Todo ello se puede visualizar de manera gráfica en un panel de control centralizado que además permite la comparación entre experimentos y la creación de informes interactivos.

Capítulo 4

Resultados

En este capítulo se presentan los resultados de los experimentos descritos en el capítulo anterior. Para la ejecución se ha utilizado una máquina virtual de Microsoft Azure cuyo sistema presenta las siguientes características:

- Sistema operativo: Ubuntu 24.04
- 113 GB de memoria RAM
- CPU de 6 núcleos
- Tarjeta gráfica: NVIDIA Tesla V100-PCIE-16GB

La versión del cliente de Weights & Biases utilizado para monitorizar los resultados es la versión 0.19.2. Por otra parte, la versión de Python utilizada es la 3.12.3.

Es necesario mencionar que en el primer conjunto la clase "persona" está infrarepresentada. Como mínimo debería haber el mismo número de instancias de dicha clase que de la clase "cabeza" y sin embargo hay 8 veces menos. Esto se debe a un incorrecto proceso de etiquetado de esta clase. Una posible solución hubiera sido volver a etiquetar el conjunto de datos o aumentar los datos mediante rotación, traslación de imágenes, entre otros posibles, en un proceso conocido como data augmentation. Sin embargo, y por falta de tiempo, se procedió con el conjunto de datos actual. Cuando se obtuvo un valor bajo en las métricas para dicha clase, se volvió a ejecutar el modelo utilizando una técnica que asigna más peso a las clases con peor mAP en la siguiente época, con el fin de que las imágenes que contengan más objetos con una mAP baja tengan más probabilidad de ser seleccionadas durante el entrenamiento. No obstante no se obtuvo una mejora significativa en los resultados y el valor de la mAP de la clase 'persona' seguía siendo significativamente más bajo que para el resto de clases, tal y como muestra la figura 4.1.

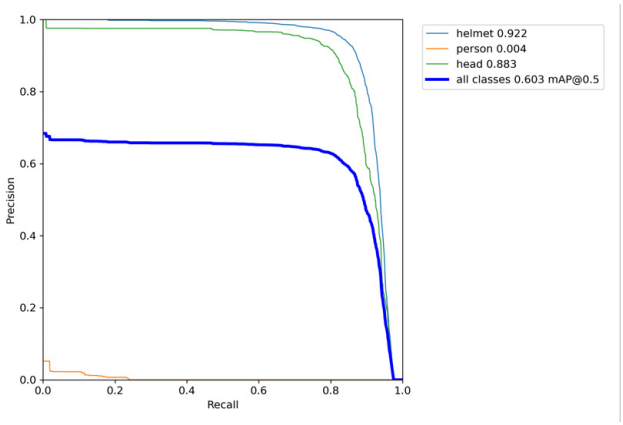


Figura 4.1: Curva precisión-recuperación para el primer conjunto de datos . Modelo: YOLOv5s.

En consecuencia, y teniendo en cuenta que la clase de más interés de este conjunto es la clase "casco", se decidió eliminar esta clase en los sucesivos experimentos con el objetivo de conseguir el mejor resultado posible y evitando que la clase "persona" afecte negativamente al modelo. Por lo tanto, todos los experimentos descritos en las subsiguientes secciones sólo cuentan con dos clases para el conjunto número 1: cabeza y casco de seguridad.

En las secciones subsiguientes se procede a evaluar las métricas descritas anteriormente. Para no sobrecargar la exposición de los resultados, no se han presentado todas las métricas para el conjunto total de experimentos sino que siempre se han presentado los resultados de la mAP@0.5:0.95. En caso de que existan resultados destacables tanto positivos como negativos para alguna de las otras métricas se ha procedido a comentarlos. Se han realizado informes interactivos con Weights & Biases que se pueden consultar escaneando los códigos QR de la figura 4.2.

SCAN ME

(a) Conjunto #1

SCAN ME

(b) Conjunto #2

Figura 4.2: Códigos QR de acceso a los informes interactivos completos.

Por último se evalúan los modelos desde la perspectiva de uso de los recursos del

sistema.

4.1. Evaluación de métricas

Según Glenn Rocher, el desarrollador principal de YOLOv5, en la mayoría de los casos se pueden obtener resultados decentes sin que haga falta modificar los modelos iniciales, suponiendo que el conjunto de datos sea lo suficientemente grande y esté bien etiquetado[15]. Sin embargo, y teniendo en cuenta que uno de los objetivos principales es el de minimizar la huella de carbono, se han ejecutado los experimentos de dos maneras: entrenando el modelo entero sobre el nuevo conjunto de datos y entrenando el modelo pero congelando los pesos de algunas capas. Dado que la arquitectura de los modelos es la misma y sólo cambia la profundidad de las capas, se ha congelado la parte de atrás (backbone) que corresponde a la parte del modelo extractora de características (ver figura 3.7). La figura 4.3 muestra el resultado de la mAP para un umbral del 0.5:0.95, tanto para cuando se ha entrenado el modelo entero como cuando se ha entrenado congelando capas. En ambos casos se ha ejecutado el modelo durante 100 épocas y con un tamaño de batch de 32. Por simplicidad sólo se muestran en la figura 4.3 los resultados del modelo más pequeño y del mediano (s y m respectivamente) para ambos conjuntos de datos con YOLOv5.

	Conjunto de datos					
	Conjunto 1			**Conjunto 2**		
Modelo	**mAP@0.5**	**mAP@0.5:0.95**	**Tiempo (min)**	**mAP@0.5**	**mAP@0.5:0.95**	**Tiempo (min)**
YOLOv5s	0,9191	0,5507	41,20	0,2914	0,1435	7,43
YOLOv5s_frozen	0,9063	0,5289	41,40	0,2582	0,1164	7,19
YOLOv5m	0,9208	0,5437	46,38	0,2989	0,1519	10,37
YOLOv5m_frozen	0,9083	0,5247	44,13	0,2673	0,133	10,32
YOLOv5l	0,92	0,5466	58,70	0,2807	0,1486	13,37
YOLOv5l_frozen	0,9123	0,5308	47,80	0,295	0,1483	13,18
YOLOv5x	NA	NA	NA	0,2862	0,1557	21,53
YOLOv5x_frozen	0,9092	0,5324	58,37	0,2733	0,1358	21,53
YOLO11n	0,8889	0,5535	30,87	0,2840	0,1638	9,05
YOLO11n_frozen	0,8762	0,5535	30,97	0,2793	0,1638	9,02
YOLO11s	0,9128	0,5585	32,59	0,2698	0,1647	9,61
YOLO11s_frozen	0,9091	0,5492	33,78	0,2950	0,1655	9,75
YOLO11m	0,9237	0,5707	49,50	0,3037	0,1701	11,90
YOLO11m_frozen	0,9237	0,5795	49,78	0,3037	0,1662	11,88
YOLOv7l	0,9130	0,5720	62,45	0,3300	0,2880	15,88
YOLOv7l_frozen	0,8988	0,5734	63,35	0,3178	0,1731	15,630
YOLO11x	0,9161	0,5763	90,50	0,2863	0,1562	20,13
YOLO11x_frozen	0,9012	0,55420	90,96	0,2894	0,1691	20,02

Tabla 4.1: Resumen del rendimiento de los modelos de YOLOv5 y YOLO11 (congelando capas y sin congelar) en ambos conjuntos de datos.

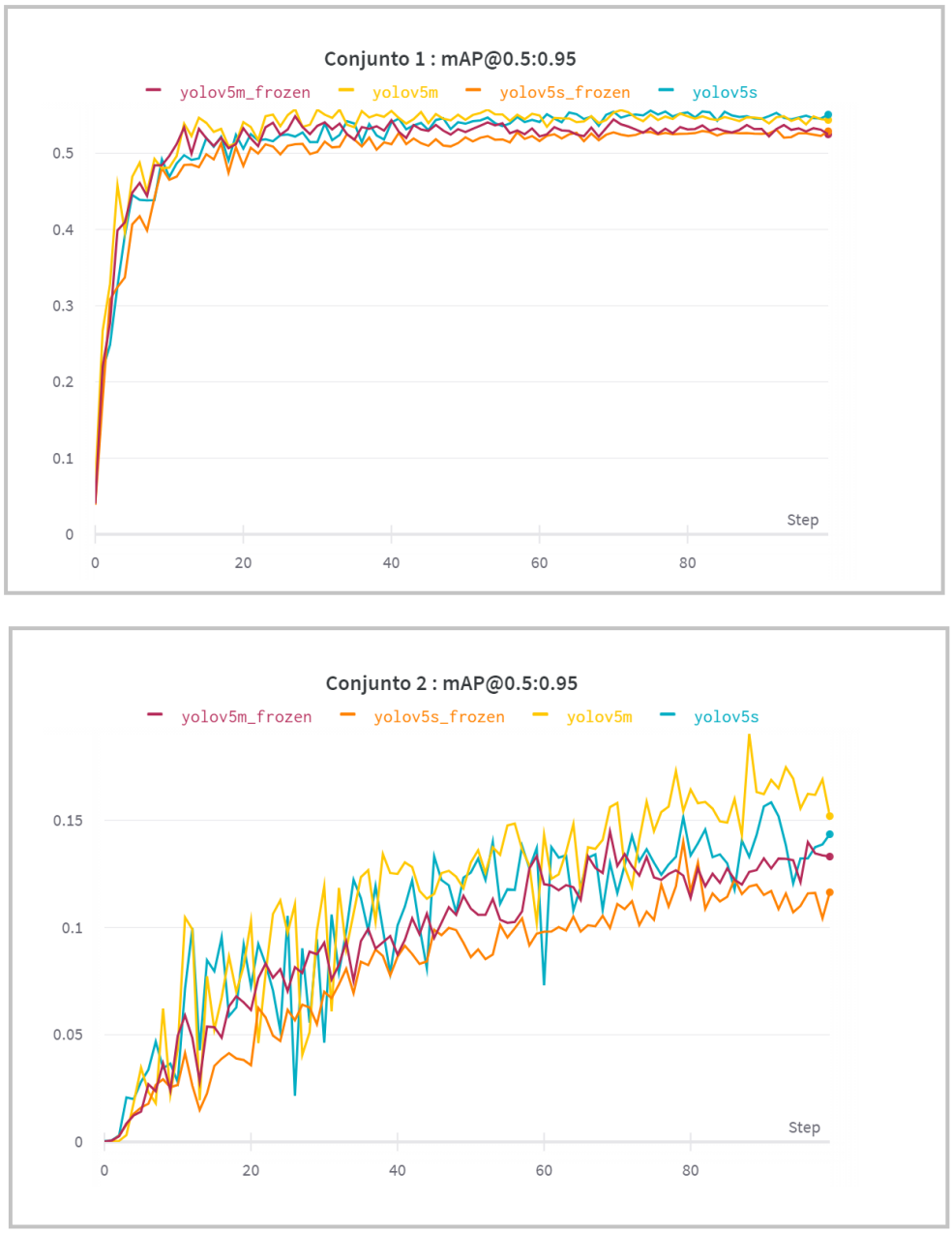


Figura 4.3: Resultados de mAP@0.5:0.95 para YOLOv5s y YOLOv5m congelando capas (frozen) y sin congelar para ambos conjuntos de datos.

En el caso del primer conjunto y tal y como se aprecia en la figura 4.3, a partir de la época 60 el modelo fluctúa en valores parecidos de mAP y el valor para la época final no es el mayor conseguido en la ejecución. YOLO archiva los pesos de la última época y de la que ha producido los mejores resultados. Por consiguiente, a la hora de comparar

modelos y para evaluar cuál es el mejor en este caso hay que fijarse en la gráfica, la cual muestra que el modelo con más parámetros (m en la gráfica) posee una mAP mayor que el del modelo s (que tiene un número de parámetros inferior) y las versiones de los mismos con capas congeladas muestran una mAP menor. Esto se ha comprobado en el resto de modelos (l y x para YOLOv5 y n, l, x para YOLO11) y siguen la misma pauta. Asimismo, se puede observar que, para el mismo tamaño entre modelos diferentes, YOLOv5 y YOLO11 alcanzan resultados similares en mAP. Si bien éstos muestran una ligera mejora utilizando YOLO11, los tiempos de ejecución son más elevados también.

Se pueden extraer las mismas conclusiones con respecto al segundo conjunto de datos, si bien se puede observar que el número de épocas no es suficiente para este conjunto y presenta margen de mejora.

Teniendo en cuenta los resultados, a partir de este punto los experimentos se han realizado congelando el backbone en todos los modelos. Esto se debe a que, si bien los resultados son mejores cuando no se congelan capas éstos son muy parecidos y el tiempo de ejecución es menor.

4.1.1. Épocas

Según se ha visto anteriormente, para el segundo conjunto de datos el número de épocas configurado (100) es insuficiente ya que se observa que el modelo puede seguir aprendiendo. Asimismo no sorprende su bajo valor de mAP, ya que las clases de este segundo conjunto (defectos en acero laminado en caliente) no tienen ninguna similitud con las clases con las que han sido entrenados los modelos de YOLO. Por lo tanto, se han realizado experimentos aumentando el número de épocas con el fin de observar cuánto margen de mejora tienen los diferentes modelos. Los resultados que se pueden observar en la figuras 4.4 y 4.5 indican que a partir de 300 épocas el aprendizaje del modelo comienza a estancarse utilizando YOLOv5. Con YOLO11 el estancamiento es menor, si bien se observa que el modelo aprende poco en cada época. Se ha realizado una prueba con 500 épocas para los modelos n ,s y m y la pauta no cambia, el modelo sigue aprendiendo pero a un ritmo muy bajo.

Con estos resultados se puede concluir que para el segundo conjunto de datos todos los modelos se beneficiarían de un entrenamiento más largo, si bien no parece que vaya a tener un impacto significativo en las métricas.

Con respecto a los resultados del primer conjunto de datos, tal y como se aprecia en la figura 4.3 el modelo parece haber alcanzado una meseta, es decir, una zona donde fluctúa cerca de su nivel máximo en cada iteración. Es posible que si se entrenara con un número elevado de épocas el modelo siguiera aprendiendo, pero la diferencia con respecto a las métricas obtenidas para 60-100 épocas en principio sería pequeña frente a un gran aumento en el tiempo de entrenamiento. Como consecuencia se ha decidido no

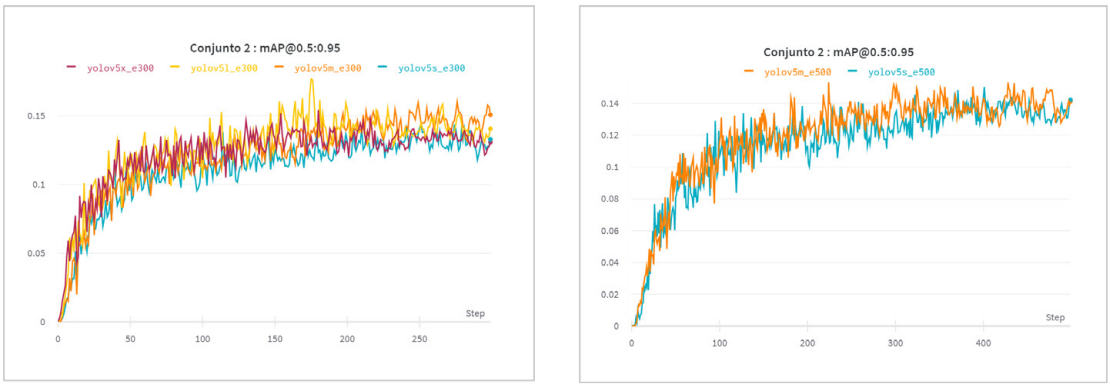


Figura 4.4: Resultados de mAP@0.5:0.95 para los diferentes modelos de YOLOv5 con 300 y 500 épocas utilizando el segundo conjunto.

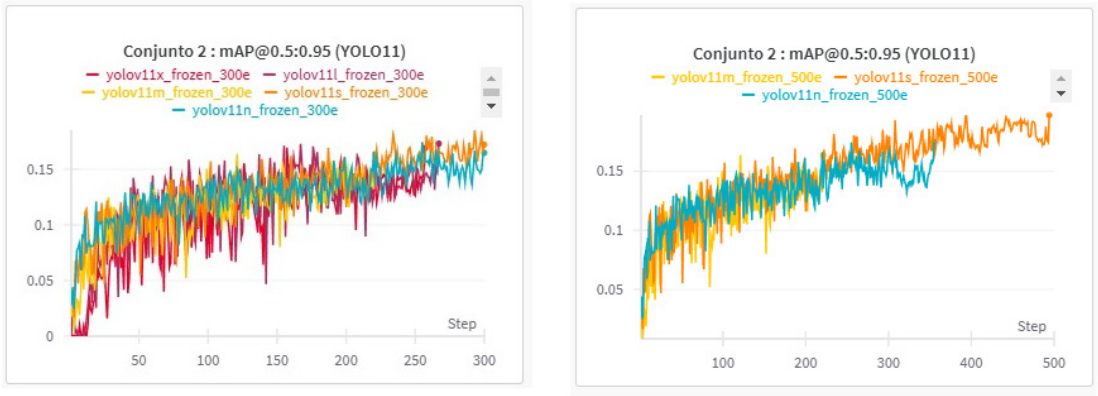


Figura 4.5: Resultados de mAP@0.5:0.95 para los diferentes modelos de YOLO11 con 300 y 500 épocas utilizando el segundo conjunto.

experimentar con más épocas para dicho conjunto.

4.1.2. Tamaño de Batch

El tamaño de batch utilizado hasta este punto ha sido de 32. En principio utilizar un tamaño de batch pequeño implica que la muestra contiene más ruido (con un menor número de muestras la probabilidad de que dos sean parecidas decrece) lo que ofrece un efecto de regularización y un error de generalización menor[104]. Asimismo un tamaño pequeño permite que al usar una GPU para entrenar el modelo todos los datos de ese batch quepan en la memoria de la tarjeta gráfica. Por otra parte, existe un intercambio entre tamaño de batch y tiempo de computación, cuanto mayor sea el tamaño de batch más rápido entrena el modelo. Como consecuencia se han llevado a cabo una serie de experimentos cambiando el tamaño de batch para determinar si existe una reducción en

el valor de las métricas cuando se aumenta el tamaño.

La tabla 4.2 muestra los resultados obtenidos para los modelos *s* y *m* (y *n* en el caso de YOLO11) para los distintos tamaños de batch utilizados.

Modelo	Tamaño de batch	mAP@0.5:0.95	Modelo	Tamaño de batch	mAP@0.5:0.95
YOLOv5s	16	0,5277	YOLOv5s	32	0,1164
	32	0,5161		64	0,1262
	64	0,5189		96	0,114
YOLOv5m	16	0,5269	YOLOv5m	32	0,133
	32	0,5221		64	0,1231
	64	0,5241		96	0,1189
YOLO11n	16	0,5707	YOLO11n	32	0,1638
	32	0,5785		64	0,1645
	64	0,5717		96	0,1693
YOLO11s	16	0,5853	YOLO11s	32	0,1655
	32	0,5835		64	0,1582
	64	0,5839		96	0,1713
YOLO11m	16	0,5799	YOLO11m	32	0,167
	32	0,5850		64	0,1643
	64	0,5874		96	0,1623

Tabla 4.2: Resultados obtenidos para los modelos YOLOv5 (s y m), YOLOs, YOLOm, y YOLO para los distintos tamaños de batch utilizados para el Conjunto #1 (izq.) y para el Conjunto #2 (dcha.)

El rango de tamaños de batch es diferente para ambos conjuntos como se puede observar en la tabla 4.2. Esto se debe a que el conjunto #1 contiene 4 veces más píxeles que el #2 por lo que es más intensivo en computación y la tarjeta gráfica no soporta un batch mayor de 64 para este primer conjunto. Aun así, como se puede observar en la tabla 4.2, los cambios de tamaño de batch no afectan a la mAP final.

Esto es consecuencia de que el código del entrenamiento de los modelos ha sido diseñado de manera que escala la función de pérdida y el weight decay con el tamaño de batch. Cuando el batch es menor de 64 se acumula la pérdida antes de optimizarla mientras que si es mayor se optimiza después de cada batch[15]. Esto es otra ventaja de la familia YOLO, ya que de esta manera los resultados se vuelven reproducibles sin necesidad de utilizar el mismo hardware. Por otra parte, esto implica que se puede reducir el tiempo de computación sin perder rendimiento del modelo configurando el tamaño de batch lo más grande posible que permita la tarjeta gráfica.

4.1.3. Clases ponderadas

Uno de los problemas en conjuntos cuyas clases no están balanceadas es que la mAP general del modelo se ve afectada de manera negativa cuando hay una clase con una mAP muy baja. Una de las posibles soluciones es hacer data augmentation previo a la ejecución del modelo. Sin embargo, YOLOv5 ya realiza tres tipos de data augmentation por defecto: escalado, ajuste de color y mosaico. Esta última combina cuatro imágenes en una sola donde cada una de las imágenes es colocada en una tesela de tamaño aleatorio. Otra posible solución es tomar muestras del conjunto de entrenamiento ponderadas por la mAP inversa de la época anterior (en vez de tomar muestras de manera uniforme como se hace normalmente). Como consecuencia, las imágenes que contengan objetos con una mAP baja tendrán más probabilidades de ser seleccionadas en la siguiente época, por lo que el modelo se centrará en aprender a detectar aquellos objetos cuya mAP sea baja en un principio. Esta opción ha sido probada con los siguientes resultados, mostrados en la figura 4.6.

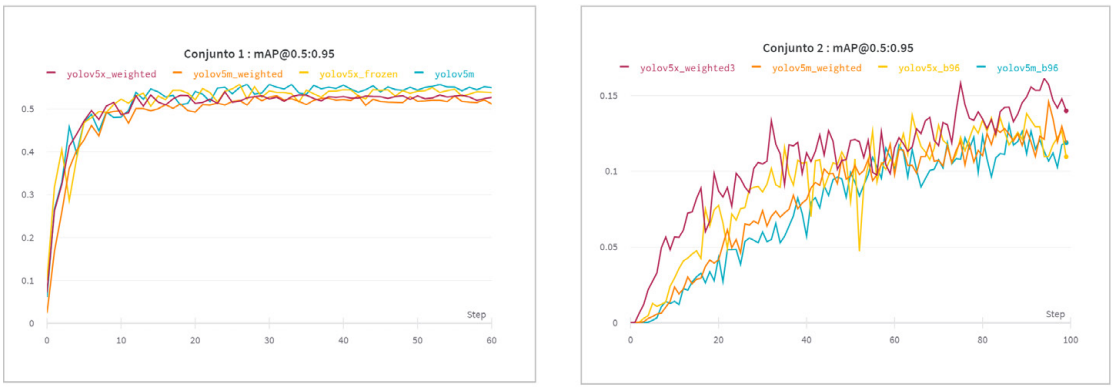


Figura 4.6: Resultados de mAP@0.5:0.95 para los modelos YOLOvm y YOLOv5x utilizando clases ponderadas.

Para el caso del primer conjunto de datos, según lo observado en la figura 4.6, los modelos que se han entrenado con ponderación de clases no sólo no han conseguido mejorar la mAP sino que además obtienen resultados más bajos que su homólogo sin ponderación. Es posible que en este caso los resultados se deban al hecho de que los valores de mAP parecen estar casi en el máximo posible, por lo que la ponderación fluctúa en torno al valor máximo como lo hacen el resto de modelos.

No obstante, para el segundo conjunto de datos sí que se puede observar en la figura 4.6 un aumento de la mAP@0.5:0.95 para ambos modelos; en este caso m y x pero extensible también a los modelos s y l. Los resultados son además coherentes con lo que se esperaba, el modelo mediano (YOLOv5m), en general, no supera la mAP del modelo extra largo (YOLOv5x) lo cual era esperable debido a que este último modelo presenta 3 veces más parámetros que el mediano. La razón por la cual esta técnica funciona para el segundo conjunto de datos pero no para el primero podría ser el margen de mejora que

presenta el segundo conjunto sin importar el modelo que se utilice. La tabla 4.3 resume las métricas y el tiempo de ejecución de los modelos para el segundo conjunto.

	Clases ponderadas	mAP@0.5:0.95	Precisión	Recuperación	Tiempo de ejecución (minutos)
YOLOv5s	Sí	0,0949	0,257	0,5718	22,43
	No	0,114	0,2444	0,6019	4.28
YOLOv5m	Sí	0,1189	0,2165	0,7697	10,57
	No	0,1189	0,2468	0,632	5,41
YOLOv5l	Sí	0,1292	0,2522	0,6732	14,43
	No	0,1329	0,2845	0,521	7,38
YOLOv5x	Sí	0,1399	0,2448	0,7045	22,3
	No	0,01296	0,2498	0,5958	13,15

Tabla 4.3: Resultados de la ejecución de los cuatro modelos de YOLOv5 con y sin ponderación de clases

Con los datos de la tabla se puede calcular la diferencia porcentual para la mAP@0.5:0.95. Para un aumento del 2,8 % generalizado para todos los modelos el tiempo de ejecución se dobla, excepto para el modelo pequeño que se multiplica por más de 5 veces. Debido a que el tiempo de ejecución sufre un aumento significativo mientras que los resultados de las métricas no se ven tan afectados se ha decidido que, de momento, los modelos continúan sin ejecutar ponderación de clases. Por esta misma razón, este experimento no se ha realizado para la familia de modelos YOLO11.

4.1.4. Optimizador

Uno de los últimos hiperparámetros en ser modificados ha sido el optimizador. YOLOv5 permite el uso por defecto de dos optimizadores: SGD y ADAM. Se ha hecho una prueba utilizando ambos para determinar cuál es el mejor para el caso de uso. La figura 4.7 muestra claramente que el mejor de los dos optimizadores es SGD, al menos para el caso de uso de ambos conjuntos.

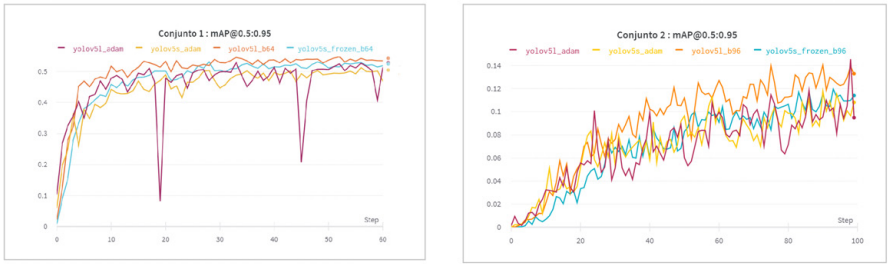


Figura 4.7: Resultados de mAP@0.5:0.95 para los modelos YOLOvs y YOLOv5l utilizando distintos optimizadores de la función de pérdida.

4.1.5. Algoritmo genético

Los experimentos detallados anteriormente han sido evaluados y se ha elegido para cada conjunto aquellos parámetros que producen un mejor resultado. Dado que un algoritmo genético es un proceso intensivo en uso de GPU, y que dependiendo del conjunto y de los parámetros del mismo puede tardar días en ejecutarse , se ha procedido a ejecutarlo sobre el modelo más pequeño de YOLOv5: YOLOv5s. Para dicho modelo, la combinación de parámetros que ha generado mejores métricas para ambos conjuntos de datos está detallada en la tabla 4.4.

	Tamaño de batch	Optimizador	Épocas	Clases ponderadas
Conjunto #1	64	SGD	60	No
Conjunto #2	96	SGD	300	No

Tabla 4.4: Combinación de parámetros con mejores resultados utilizando YOLOv5

Los parámetros de la tabla 4.4 describen las situaciones que se pretenden evolucionar por lo que sirven como base para ejecutar el algoritmo genético, uno por cada conjunto de datos. En cuanto a las generaciones del algoritmo, se considera que son necesarias un mínimo de 300 generaciones para obtener buenos resultados, especialmente cuando los resultados iniciales no son óptimos. Dentro de este último caso se encuadra el modelo para el conjunto de datos de defectos en acero laminado en caliente, por lo que se configuran 300 generaciones ya que adicionalmente por el tamaño y la resolución del mismo este modelo tiene un tiempo de ejecución corto. Por otra parte, el primer conjunto de datos ya presenta unos resultados adecuados y su tiempo de ejecución es bastante superior al del modelo del segundo conjunto incluso con menos épocas, por lo que se configuran 60 generaciones.

Se puede observar la evolución de los hiperparámetros durante las generaciones para el primer conjunto de datos y para el segundo, en las figuras 4.8 y 4.9, respectivamente .

Ambas figuras, 4.8 y 4.9, contienen un gráfico por cada hiperparámetro donde el valor del hiperparámetro se encuentra en el eje x y el valor de la función de aptitud en el eje y. En cuanto a la gama cromática, el color amarillo representa mayores concentraciones. Las líneas verticales que se aprecian en los hiperparámetros shear, perspective o flipud, entre otros, indican que estos hiperparámetros no se han evolucionado, cuestión configurable antes de ejecutar el genético. En este caso se han congelado y, por lo tanto dejado fuera de la evolución, aquellos parámetros que desde el repositorio de YOLOv5 consideran que son susceptibles de congelación si no se tiene información previa sobre los hiperparámetros en general. El apéndice B contiene una lista de los valores por defecto de los hiperparámetros así como las descripciones de los mismos. Comparando ambas figuras, se ha podido concluir que el algoritmo genético para el segundo conjunto de datos ha evolucionado el escenario base mejor que para el primer conjunto. Si se escoge cualquiera de las gráficas de la figura 4.8 que no corresponda a un hiperparámetro congelado, se puede observar

que el valor de la función de aptitud fluctúa muy poco correspondiendo además a un valor bajo. Esto puede deberse a que sólo se ha ejecutado el genético durante 60 generaciones.

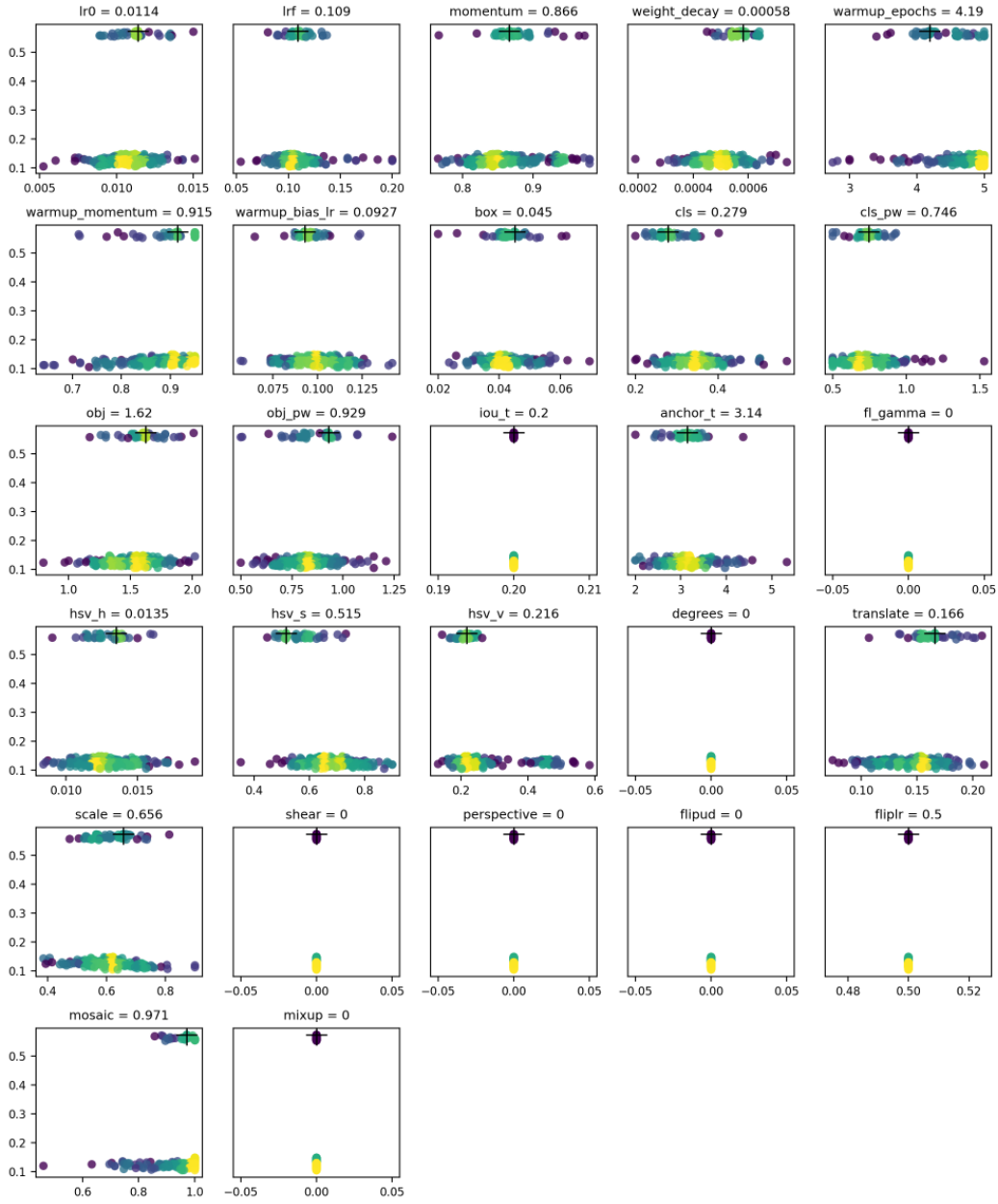


Figura 4.8: Evolución de los distintos hiperparámetros durante las generaciones en el primer conjunto de datos.

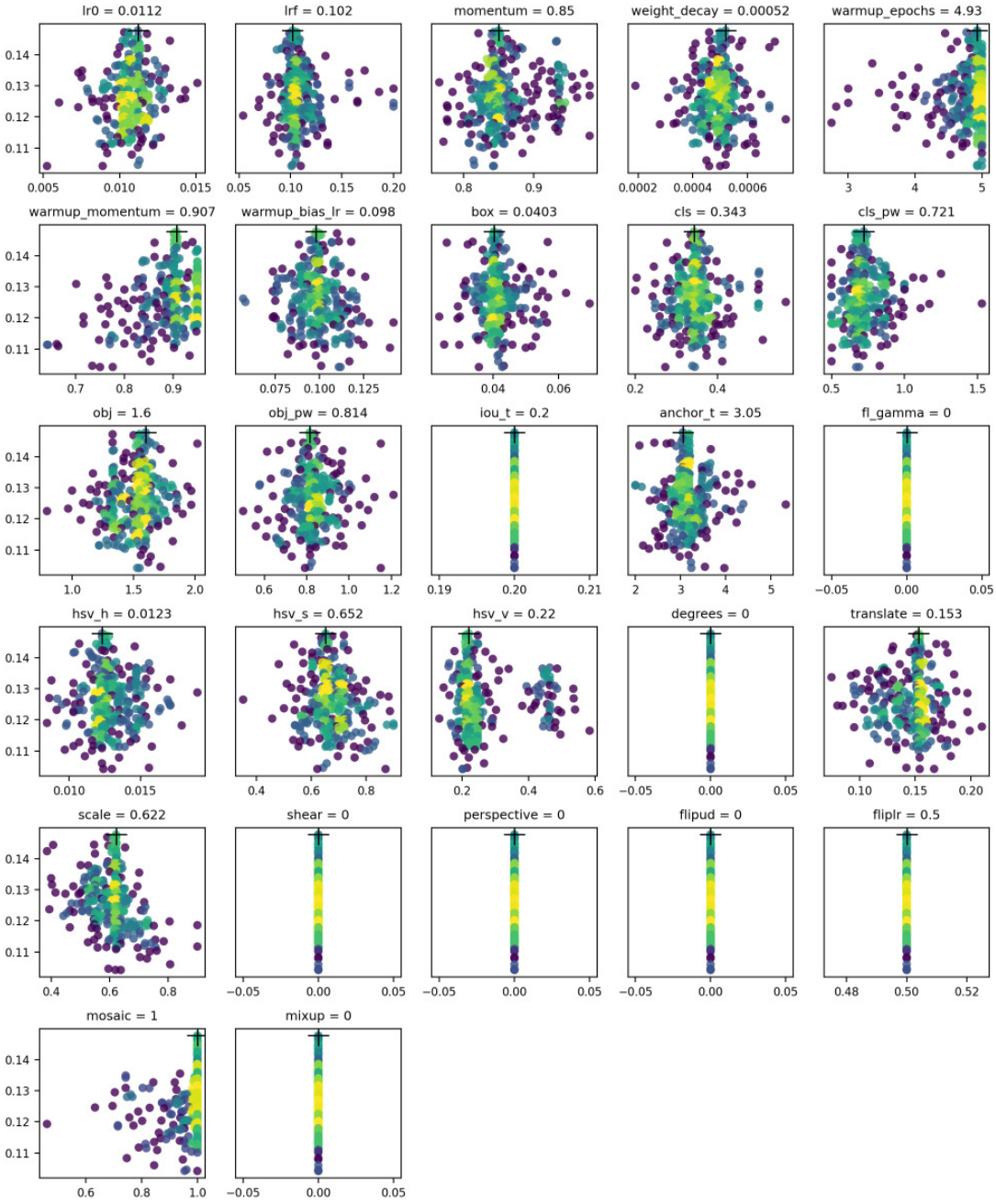


Figura 4.9: Evolución de los distintos hiperparámetros durante las generaciones en el segundo conjunto de datos.

Las tablas 4.5 y 4.6 muestran los resultados obtenidos previos a la ejecución del genético así como los posteriores. Como se puede observar, el genético ha conseguido

mejorar los resultados de precisión, recuperación y mAP, si bien la mejora es pequeña.

	Precisión	Recuperación	mAP@0.5	mAP@0.5:0.95
Pre genético	0,9182	0,8328	0,9055	0,5161
Post genético	0,907	0,847	0,909	0,537

Tabla 4.5: Resultados pre y post genético para el primer conjunto de datos.

	Precisión	Recuperación	mAP@0.5	mAP@0.5:0.95
Pre genético	0,2444	0,6019	0,2542	0,114
Post genético	0,262	0,617	0,272	0,134

Tabla 4.6: Resultados pre y post genético para el segundo conjunto de datos.

Teniendo en cuenta el tiempo de ejecución de ambos algoritmos, el consumo de energía y los resultados obtenidos para este caso de uso concreto no parece que merezca la pena la ejecución del genético. Al mismo tiempo es posible que el número de generaciones configuradas no sean suficientes por lo que los resultados se verían afectados al no haber llegado al máximo posible por falta de iteraciones. Por otro lado, también es posible que la probabilidad de mutación, que no ha sido modificada según el estándar del modelo, sea demasiado alta para el caso de uso y por lo tanto los mejores individuos pierdan aquello que les hace ser buenos candidatos con la mutación. En todos los casos, el uso de un algoritmo genético sigue siendo una opción adecuada para la búsqueda y optimización de los hiperparámetros.

Llegado a este punto se han obtenido los mejores modelos posibles para ambos conjuntos de datos y para el modelo pequeño se ha realizado una búsqueda para optimizar los valores de los hiperparámetros con el propósito de mejorar el modelo base. Para estos últimos dos modelos (modelo YOLOv5s con hiperparámetros optimizados) se realiza inferencia en el conjunto de imágenes de test. El objetivo es calcular la mAP por clase en un conjunto de datos diferente al que se ha utilizado para el entrenamiento y determinar si el modelo es capaz hacer predicciones de manera correcta. La tabla 4.7 muestra los valores de mAP@0.5 para el primer conjunto de datos desglosado por clases. Tal y como se aprecia en dicha tabla, no existe sobreaprendizaje para el modelo pequeño optimizado en el primer conjunto de datos.

Clase	mAP@0.5 train	mAP@0.5 test
Cabeza	0,888	0,861
Casco	0,926	0,937

Tabla 4.7: mAP@0.5 para el primer conjunto de datos desglosado por clases tanto para el conjunto de train como para el de test.

El mismo proceso se ha repetido para el segundo conjunto de datos, cuyos resultados se muestran en la tabla 4.8. Este modelo tampoco presenta sobreaprendizaje, lo que significa que los hiperparámetros configurados son correctos para este caso de uso.

Clase	mAP@0.5 train	mAP@0.5 test
Crazing	0,206	0,198
Inclusion	0,164	0,144
Patches	0,277	0,23
Pitte surface	0,238	0,233
Rolled-in scale	0,062	0,07
Scratches	0,579	0,528

Tabla 4.8: mAP@0.5 para el segundo conjunto de datos desglosado por clases tanto para el conjunto de train como para el de test.

4.2. Uso de recursos del sistema y emisiones de CO_2

Esta sección está dedicada a la evaluación del uso que hacen los modelos de los recursos del sistema. En concreto se ha buscado monitorizar y evaluar el uso que hace de la tarjeta gráfica para calcular las emisiones de CO_2. A día de hoy los modelos que ejecutan profesionales de machine y deep learning típicamente usan hasta ocho tarjetas gráficas pero no son grandes productores o emisores de CO_2 [105]. Sin embargo, las grandes empresas como Google o Facebook entrenan modelos que requieren de un número mucho más elevado de GPUs además de acumular varios modelos. Además no sólo hay que tener en cuenta el entrenamiento de éstos sino la inferencia que se realiza con ellos. Por ejemplo, algunos prototipos de modelos para conducción autónoma utilizan para hacer inferencia hasta 2500 Vatios, por lo que si se utilizaran en todos los coches del mundo el impacto sería significativo [105].

Por otra parte, la monitorización de los modelos es una herramienta necesaria para mejorar la eficiencia energética de los mismos, teniendo un impacto directo sobre el coste en la empresa. De la misma manera, si al ejecutar modelos se guardan sus resultados y los parámetros que se han utilizado se evita una gran cantidad de computación redundante.

En este proyecto, de todos los experimentos que se han realizado se pueden extraer las siguientes conclusiones con respecto al uso de la GPU, tanto para YOLOv5 como para YOLO11.

- A mayor número de capas congeladas, más rápido el entrenamiento y menor uso de la GPU

- A mayor número de parámetros, mayor tiempo de ejecución y, por lo tanto, más uso de la GPU.
- El tamaño de batch afecta no sólo al tiempo de ejecución sino que también afecta a la potencia utilizada por la GPU y a la memoria que la misma requiere para entrenar.
- Asignar más peso a las clases con peor mAP en la siguiente época, con el fin de mejorar dicha métrica para dicha clase, es un proceso intensivo en GPU.
- YOLO11 es más intensivo en el uso de energía que YOLOv5.

El primer experimento realizado buscaba comparar un modelo entrenado por completo sobre el nuevo conjunto de datos contra un modelo parcialmente entrenado sobre el nuevo conjunto (modelo con capas congeladas). Si bien entrenar con capas congeladas repercute de manera negativa tanto sobre la mAP como sobre la precisión y la recuperación, lo hace de manera positiva sobre el uso de la GPU. En la figura 4.10 se puede apreciar que tanto la memoria requerida en la GPU para ejecutar el modelo como el uso de la misma decrecen al utilizar modelos con capas congeladas. Para no sobrecargar las gráficas éstas sólo muestran los resultados de dos modelos para el primer conjunto de datos (m y l) pero los resultados son extensibles al resto de modelos, si bien es cierto que cuanto más pequeño el modelo menor la diferencia. Cuantos más módulos se congelen, menos memoria se requiere y menor uso se hace de la GPU. Esto indica que los modelos con conjuntos de datos más grandes o con imágenes de alta resolución pueden encontrar una ventaja en la congelación de capas con el fin de entrenar de manera más rápida.

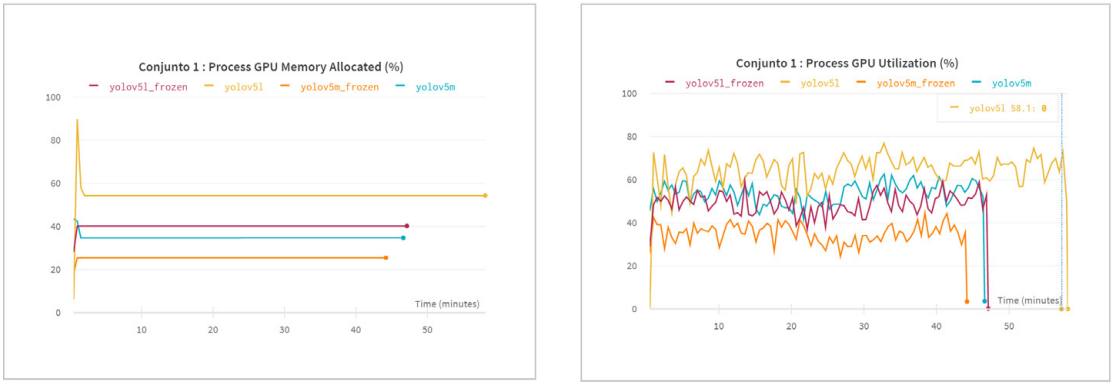


Figura 4.10: Porcentaje de memoria utilizado por la GPU (izquierda) y porcentaje de uso de la GPU (derecha) en modelos entrenados al completo y con capas congeladas. Tamaño de batch = 32, épocas = 100

La tabla 4.9 muestra las diferencias porcentuales entre los resultados obtenidos para YOLOv5 y YOLO11 en todos sus tamaños. En general, YOLO11 muestra mejoras en

las métricas (tanto para ambos umbrales de la mAP considerados como para precisión y sensibilidad) a costa de tiempos de ejecución y uso de GPU.

Tamaño	Conjunto	mAP@0.5 (%)	mAP@0.5:0.95 (%)	Tiempo (%)
Entrenamiento completo				
s	1	-0,69	1,42	-20,88
	2	-7,40	14,76	29,34
m	1	0,32	4,96	6,73
	2	1,61	12,00	14,75
l	1	0,00	4,29	6,38
	2	-4,84	-0,20	18,79
x	1	NA	NA	NA
	2	1,12	-12,66	-6,51
Entrenamiento con capas congeladas				
s_frozen	1	-2,97	3,98	-18,37
	2	-12,45	40,88	29,09
m_frozen	1	1,70	10,44	12,81
	2	13,63	24,89	15,12
l_frozen	1	0,69	8,13	-18,58
	2	-6,44	0,00	1,44
x_frozen	1	-0,88	4,10	55,88
	2	5,90	-11,83	-6,49

Tabla 4.9: Diferencias porcentuales en métricas mAP y tiempo entre YOLO11 y YOLOv5, tanto para entrenamiento completo como con capas congeladas.

Para el conjunto 2, la media del incremento es de 3,5-13,5 % en la mAP@0.5:0.95 dependiendo del tamaño y de si se entrena el modelo entero o con capas congeladas. En cuanto al tiempo de ejecución, éste se incrementa en un 8-9,8 % más que YOLOv5. Para el conjunto 1, la media oscila entre 3,5-6,7 % para la mAP@0.5:0.95 y el tiempo entre un 2,4 % menor y un 8 % mayor según el modelo. Sin embargo, tal y como muestra la figura 4.11, el uso de la GPU utilizando YOLO11 es, en algunos casos, un 50 % mayor para el conjunto 1.

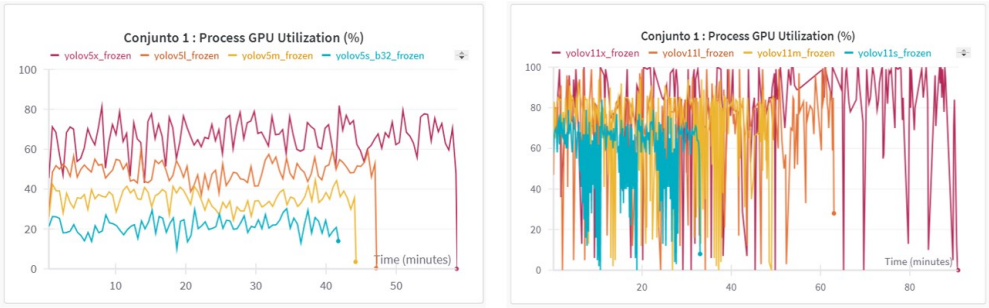


Figura 4.11: Porcentaje de uso de la GPU utilizando YOLOv5(izquierda) y YOLO11 (derecha). Conjunto 1, 100 épocas y con 32como tamaño de batch.

Asimismo, la tabla 4.10 muestra el consumo de energía de la GPU para el entrenamiento del Conjunto 1. Se puede concluir que el entrenamiento de modelos de YOLO11 hace un uso más intensivo de la energía que YOLOv5 para el mismo tamaño. Por ejemplo, para el modelo *m*, YOLO11 consume un 21 % más que el homólogo de YOLOv5 y, en concreto, esto se traduce en un 5 % de mejora en la mAP@0.5:0.95.

	YOLOv5 (kJ)	YOLO11(kJ)
s	192,17	257,69
s (frozen layers)	153,90	196,81
m	346,50	419,36
m (frozen layers)	224,31	381,21
l	481,77	556,86
l (frozen layers)	389,92	535,62
x	NA	943,58
x (frozen layers)	531,06	856,98

Tabla 4.10: Comparación de energía en kilojulios (kJ) entre YOLOv5 y YOLO11 para los diferentes tamaños.

El segundo experimento buscaba comparar el tamaño de batch con le objetivo de maximizar el uso de la gpu con el mñinimo descenso posible en métricas. La siguiente gráfica, en la figura 4.12, muestra el porcentaje de memoria utilizado por la GPU y el uso de la misma para el segundo conjunto de datos y para diferente tamaño de batch utilizando YOLOv5. Como se puede observar, un mayor tamaño de batch precisa de un uso menor de la memoria de la tarjeta, así como menos tiempo de ejecución y, por lo tanto, menor uso de la GPU.

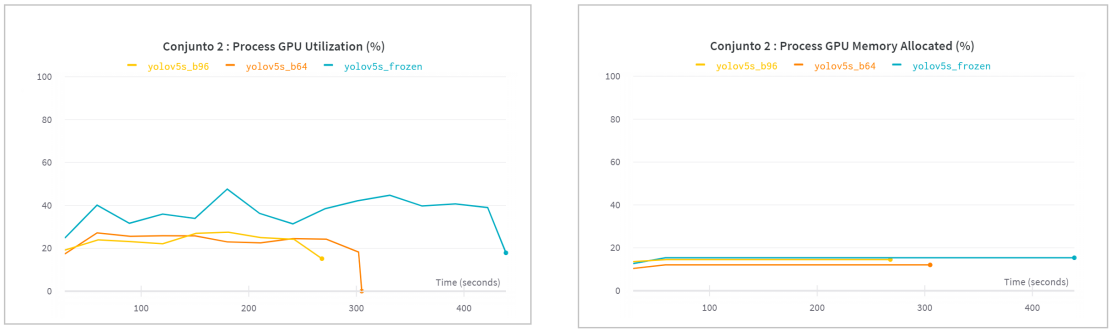


Figura 4.12: Porcentaje de memoria utilizado por la GPU (izquierda) y porcentaje de uso de la GPU utilizando distintos tamaños de batch en el modelo mediano (YOLOv5m) .

En concreto, un incremento de tres veces el tamaño de batch de inicio (32) supone una reducción a la mitad del tiempo de ejecución y de la memoria requerida mientras que sólo supone una reducción del 2 % en mAP y un 0,01 % en precisión y recuperación, tal y como se puede calcular a partir de la tabla 4.11.

	Tamaño de batch	mAP@0.5:0.95	Precisión
YOLOv5s	32	0,1164	0,2914
YOLOv5s	64	0,1262	0,2523
YOLOv5s	96	0,114	0,2444

Tabla 4.11: Resultados del modelo YOLOv5s para distintos tamaños de batch.

El segundo conjunto de datos es el más pequeño de los dos, con tan sólo cerca de 1000 imágenes para el entrenamiento y con una baja resolución (200 x 200 píxeles). Esto se considera un conjunto de imágenes de tamaño pequeño, por lo que el hecho de reducir a la mitad el tiempo de ejecución cobra más importancia en modelos donde se utilice una mayor cantidad de imágenes.

Estas conclusiones son extensibles a los distintos tamaños del modelo, tanto para YOLOv5 como YOLO11. La figura 4.13 muestra la evolución del uso de la GPU en ambos conjuntos y para el modelo *s* de ambos. En cuanto a las diferencias entre YOLOv5 y YOLO11, la mAP es cerca de un 8 % mayor para los modelos de la familia YOLO11, sin embargo, el uso de la GPU es, de nuevo, superior, como se puede observar en la figura 4.13).

Figura 4.13: Resultados de uso de la GPU para el modelo YOLOv5s y YOLO11s utilizando diferente tamaño de batch.

Dados los tiempos de computación de cada experimento y conociendo las especificaciones técnicas de la GPU utilizada se puede hacer una estimación de las emisiones de CO_2 producidas. Las estimaciones se han llevado a cabo mediante el uso de una calculadora denominada 'Machine Learning Impact Calculator' presentada en el artículo "Quantifying the Carbon Emissions of Machine Learning"[106]. Para los ejemplos de tamaño de batch presentados en esta sección las emisiones de CO_2 son las reflejadas en la tabla 4.12.La columna de compensación de CO_2 hace referencia a una reducción de CO_2 u otros gases de efecto invernadero, típicamente a través de apoyo económico a proyectos que buscan reducir la emisión de estos gases. Microsoft Azure, al menos al nivel de este proyecto, compensa lo mismo que emite con el uso de sus máquinas virtuales. Estos datos de emisión de carbono son valores pequeños comparados, por ejemplo, con emisiones producidas por otros modelos de mayor tamaño. Por ejemplo, en su artículo 'Energy and Policy Considerations for Deep Learning in NLP'[22] un grupo de investigadores liderados por Emma Strubell entrenaron un modelo de deep learning de tamaño medio y estimaron las emisiones de entrenar, no sólo la versión final, sino de los experimentos previos hasta llegar a dicha versión. Durante seis meses entrenaron 4.789 versiones diferentes que requirieron del equivalente a casi 10.000 días de computación de GPU, lo que estimaron que suponía unas emisiones cercanas a 36.000 Kg CO_2 eq. Aunque las emisiones sean mucho más pequeñas para el caso descrito, en la tabla 4.12 entre el primer caso de uso y el último ya hay una reducción del 66 % en la emisión. Si esto se pudiera extender a casos

de uso más intensivos en energía se reduciría la huella de carbono considerablemente.

	Tamaño de Batch	Emisión de CO2 (kg CO2 eq.)	Compensación de CO2 (kg CO2 eq.)
YOLOv5s / YOLO11s	32	0,03	0,03
	64	0,02	0,02
	96	0,01	0,01

Tabla 4.12: Emisiones de CO_2 para diferentes tamaños de batch en los modelos YOLOv5s/YOLO11s utilizando el segundo conjunto de datos.

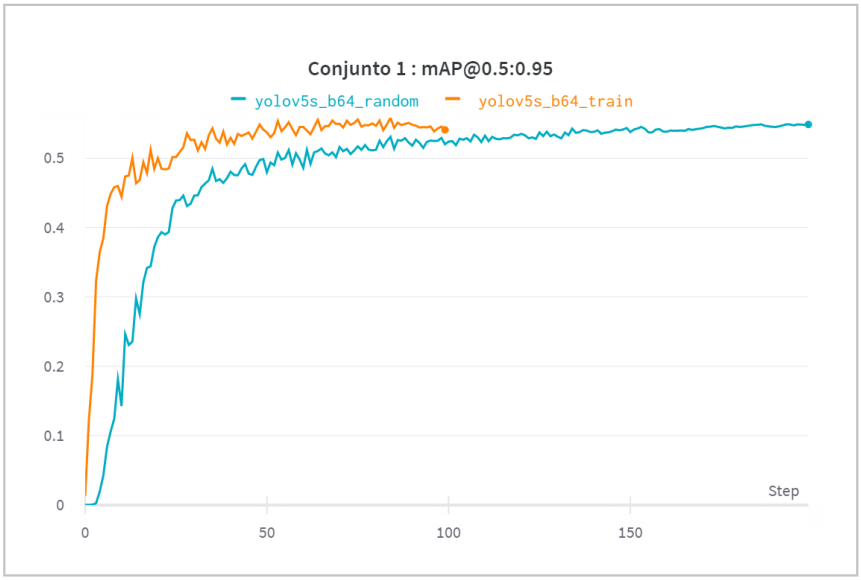
4.2.1. Emisiones según el punto de partida

Hasta este punto se han comparado las emisiones de CO_2 que resultan de utilizar la red neuronal con los pesos optimizados, lo cual proporciona una estimación de la reducción en emisiones en la medida en la que se pueda optimizar la red. Sin embargo, también se busca calcular la reducción de emisiones que se produce al utilizar una red neuronal preentrenada contra las producidas si se tuviera que crear dicha red desde cero.

Una primera aproximación a este problema es mediante la comparación de un modelo donde se utilizan los pesos optimizados de YOLOv5 contra un modelo donde la inicialización de los pesos sea aleatoria. De esta forma se compara el uso de una red preentrenada contra el uso de una arquitectura ya diseñada pero que no ha sido entrenada. Para ello se entrenan ambos conjuntos de datos con los pesos optimizados, con pesos aleatorios y sin congelar capas en ninguno de los casos. Se busca encontrar los parámetros que, inicializando de manera aleatoria los pesos de la red, produzcan los mismos resultados que cuando se hace uso de los pesos optimizados para MS COCO 2017 en YOLOv5.

La figura 4.14 muestra el resultado de este experimento para el modelo YOLOv5s en ambos conjuntos de datos. Como se puede observar en las gráficas, para el primer conjunto de datos el modelo con la inicialización aleatoria de pesos necesita algo más de 200 épocas para conseguir igualar el mejor resultado del modelo con pesos optimizados. Dicho resultado sucede en torno a la época 50, lo que supone una diferencia de más de 150 épocas y de cuatro veces más de tiempo de ejecución.

La figura 4.15 muestra un ejemplo de las etiquetas originales versus las predichas por el modelo pequeño de YOLOv5 utilizando los pesos optimizados.



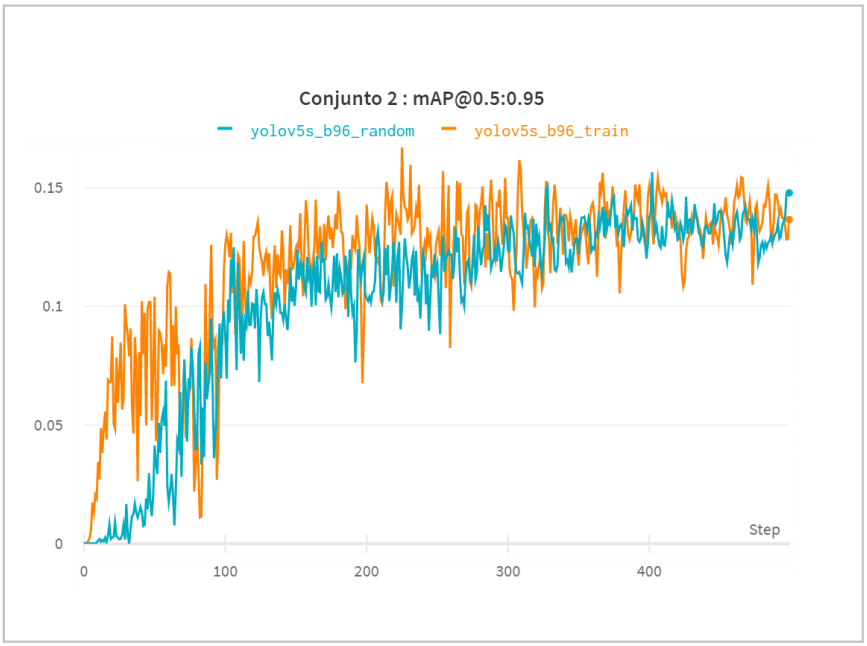


Figura 4.14: Evolución en el tiempo de YOLOv5s entrenado sin congelar capas, inicializado primero con pesos aleatorios y después con los pesos optimizados.

En términos de emisiones de CO_2 esto queda reflejado en la tabla 4.13. De la tabla se puede obtener la diferencia porcentual que existe entre las emisiones de un modelo con el otro. Tomando como base el modelo inicializado con los pesos optimizados, las emisiones para el modelo sin pesos optimizados equivalen a casi cuatro veces las del primer modelo.

Figura 4.15: Ejemplo de detección en un batch del conjunto de validación. Etiquetas originales (izq.) versus etiquetas predichas por el modelo (dcha.).

A la vista de esta comparación parece claro que, dado el volumen de modelos que se diseñan y entrenan al día de forma global, la reducción de emisiones de CO_2 entre un modelo con pesos optimizados y uno entrenado desde cero es significativa.

	Parámetros	mAP@0.5:0.95	Emisión de CO2 (kg CO2 eq.)	Compensación de CO2 (kg CO2 eq.)
YOLOv5s	Pesos optimizados Batch size = 64 Épocas =50	0,5542	0,06	0,06
YOLOv5s	Pesos aleatorios Batch size = 64 Épocas = 200	0,5486	0,23	0,23

Tabla 4.13: Emisiones de CO_2 estimadas para el primer conjunto de datos utilizando los modelos descritos en la tabla.

No ocurre lo mismo para el segundo conjunto de datos ya que si observamos la segunda gráfica de la figura 4.14 se puede observar que ambos casos son muy similares. La inicialización con pesos aleatorios empieza a aprender más despacio que con los pesos optimizados y en ningún momento llega a igualar en mAP@0.5:0.95 a esta última para el mismo número de épocas. Estos resultados eran de esperar, debido a que el rendimiento pobre de este conjunto de datos se debe en parte a las pocas imágenes del conjunto así como el bajo número de instancias de cada clase. Sin embargo, la inicialización aleatoria de pesos sí alcanza en el resto de métricas (recuperación, precisión y mAP@0.5) al modelo que utiliza los pesos óptimos, tal y como se aprecia en la figura 4.16.

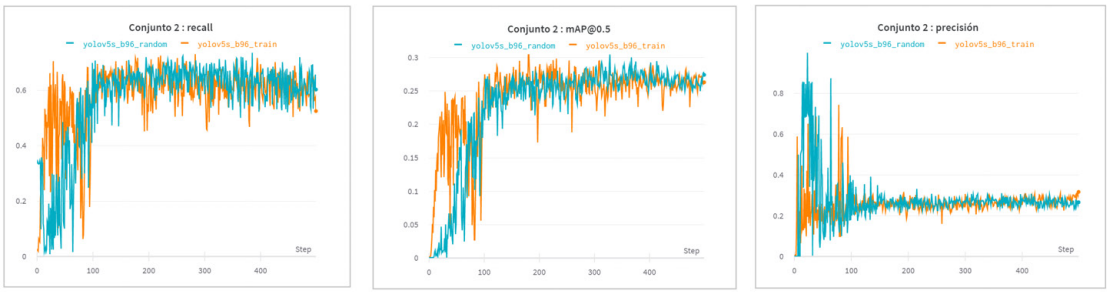


Figura 4.16: Evolución en el tiempo de diferentes métricas de YOLOv5s entrenado sin congelar capas, inicializado primero con pesos aleatorios y después con los pesos optimizados.El conjunto de datos utilizado es el segundo.

Observando la figura 4.16 se puede ver que ambos modelos producen un valor de recuperación más alto que de precisión, lo que indica que producen muchos resultados, de los cuales pocos son falsos negativos pero la mayoría son incorrectos comparados con los originales que se toman como verdaderos. La figura 4.17 muestra un ejemplo de las etiquetas originales versus las predichas por el modelo yolov5s_b96_train, es decir, el modelo que usa los pesos optimizados en YOLOv5. Al analizar dicha figura se observa que para la clase 'crazing' no hay falsos negativos y la precisión oscila entre el 0.3-0.9.

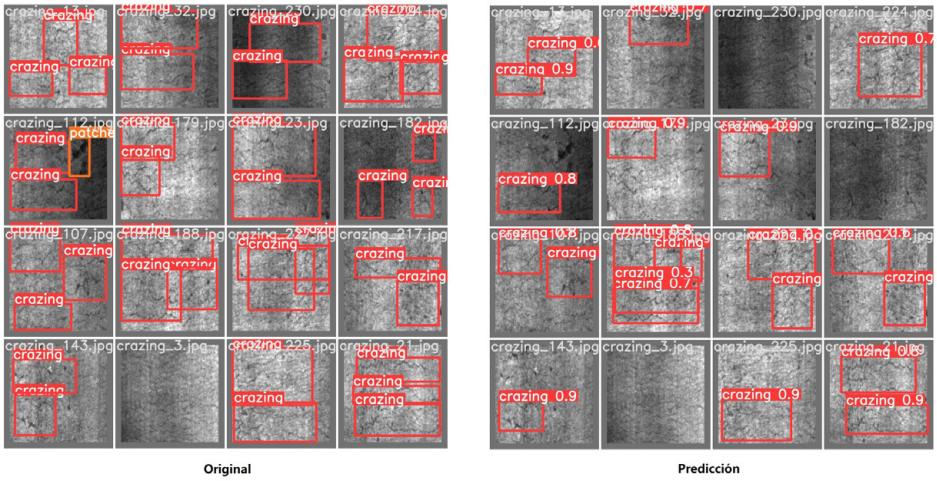


Figura 4.17: Ejemplo de detección en un batch del conjunto de validación. Etiquetas originales (izq.) versus etiquetas predichas por el modelo (dcha.).

Teniendo en cuenta que la mAP general es muy baja esto debe significar que el resto de clases de este conjunto de datos tienen una mAP individual más baja que para este clase, lo que se confirma en la figura 4.18, donde se aprecia que para las clases 'crazing' y 'scratches' el valor de la mAP es significativamente mayor que para el resto. Dados los resultados, y como se ha mencionado en secciones anteriores, este experimento no se ha llevado a cabo utilizando más modelos ya que se entiende que es generalizable al resto.

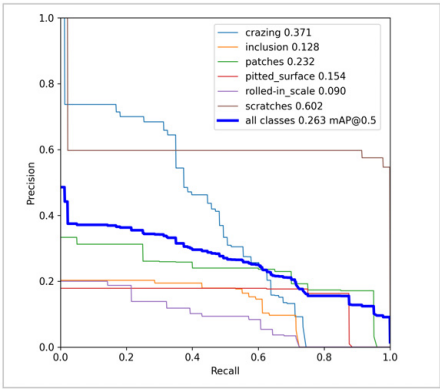


Figura 4.18: Curva precisión-recuperación para el segundo conjunto de datos, utilizando como modelo YOLOv5s sin congelar, con pesos optimizados, tamaño de batch de 32 y 500 épocas

Por ahora las emisiones calculadas hacen referencia a la diferencia entre hacer uso de la red con los pesos optimizados o inicializando de manera aleatoria. Sin embargo, la forma más adecuada de evaluar la reducción de CO_2 es mediante la comparación del uso de la red frente a la creación de la misma desde cero. A falta de datos oficiales para YOLOv5 y YOLO11, y dado que no se ha dispuesto de información acerca de las horas de entrenamiento para ninguno, se ha hecho una estimación del tiempo de computación de la misma utilizando el artículo 'Energy and Policy Considerations for Deep Learning in NLP'[22] como base. En dicho artículo se hace un estudio del tiempo de ejecución y emisiones para tres casos diferentes: entrenar un modelo, ajustar hiperparámetros del mismo y todos los entrenamientos necesarios para conseguir el modelo adecuado. Si bien es verdad que este caso de uso está realizado en modelos de procesamiento de lenguaje natural (en inglés, Natural Language Processing (NLP)) se ha tomado como guía para estimar el tiempo que ha llevado encontrar la arquitectura adecuada en YOLOv5. La tabla 4.14 muestra el tiempo de ejecución del modelo utilizado en el artículo, Linguistically-Informed Self-Attention[107], en las tres situaciones descritas.

Situación	**Tiempo (h)**
Entrenar el modelo	120
Ajustar hiperparámetros	2880
Entrenamientos necesarios para conseguir el modelo adecuado	239.942

Tabla 4.14: Tiempo de ejecución del modelo Linguistically-Informed Self-Attention en tres situaciones diferentes. Datos obtenidos del artículo 'Energy and Policy Considerations for Deep Learning in NLP'[22].

Como se puede observar en la tabla 4.14, existe una diferencia de 3 órdenes de magnitud entre el tiempo de entrenar un modelo y el de los múltiples entrenamientos para crear el modelo funcional. Esta tabla se ha usado como base para estimar los tiempos y las emisiones de YOLOv5 y YOLO11.

Los datos correspondientes al entrenamiento del modelo han sido obtenidos de la documentación de Ultralytics[15], dónde se puede consultar el tiempo de ejecución utilizando COCO128, un subconjunto de imágenes de MS COCO 2017. Dado que el entrenamiento hasta llegar al modelo correcto se ha realizado sobre MS COCO 2017, cuya partición de entrenamiento contiene en torno a 118.000 imágenes y no 128, se ha inferido el tiempo de entrenamiento de las 118.000 imágenes a partir del tiempo del subconjunto. Este tiempo corresponde al entrenamiento del modelo una vez se ha conseguido la arquitectura adecuada, por lo que se estima que el tiempo total de los múltiples entrenamientos hasta llegar a la arquitectura final es 3 órdenes de magnitud mayor. Las tablas 4.15 y 4.16 muestran dichas estimaciones para los diferentes modelos de YOLOv5 y YOLO11 así como las emisiones de CO2 estimadas a partir de dichos tiempos.

Modelo	Situación	Tiempo estimado (h)	Emisión de CO2 (kg CO2 eq.)
YOLOv5s	Entrenar el modelo	75,00	12,82
	Entrenamientos necesarios para conseguir el modelo adecuado	75.000,00	12.825,00
YOLOv5m	Entrenar el modelo	150,00	25,65
	Entrenamientos necesarios para conseguir el modelo adecuado	150.000,00	25.650,00
YOLOv5l	Entrenar el modelo	225,00	38,47
	Entrenamientos necesarios para conseguir el modelo adecuado	225.000,00	38.475,00
YOLOv5x	Entrenar el modelo	300,00	51,30
	Entrenamientos necesarios para conseguir el modelo adecuado	300.000,00	51.300,00

Tabla 4.15: Tiempo de ejecución y emisiones de CO_2 en la utilización de los modelos de YOLOv5 y su estimación para el entrenamiento.

Modelo	Situación	Tiempo estimado (h)	Emisión de CO2 (kg CO2 eq.)
YOLO11n	Entrenar el modelo	50,00	8,55
	Entrenamientos necesarios para conseguir el modelo adecuado	50.000,00	8.550,00
YOLO11s	Entrenar el modelo	100,00	17,10
	Entrenamientos necesarios para conseguir el modelo adecuado	100.000,00	17.100,00
YOLO11m	Entrenar el modelo	150,00	25,65
	Entrenamientos necesarios para conseguir el modelo adecuado	150.000	25.650,00
YOLO11l	Entrenar el modelo	225,00	38,47
	Entrenamientos necesarios para conseguir el modelo adecuado	225.000,00	38.470,00
YOLO11x	Entrenar el modelo	270,00	46,17
	Entrenamientos necesarios para conseguir el modelo adecuado	270.000,00	46.170,00

Tabla 4.16: Tiempo de ejecución y emisiones de CO_2 en la utilización de los modelos de YOLO11 y su estimación para el entrenamiento.

Cabe recalcar que las tablas 4.15 y 4.16 sólo son una estimación llevada a cabo en este proyecto. No obstante, un modelo más antiguo, concretamente YOLOv3, necesita más de 240 horas para entrenar el modelo sobre MS COCO 2017[108], lo cual podría indicar que la estimación no es del todo incorrecta para YOLOv5.

Con los datos de la tabla se puede calcular la reducción de CO_2 que supone el haber utilizado la red en vez de crearla y entrenarla desde cero hasta obtener los mismos modelos. Las horas de entrenamiento estimadas se han llevado a cabo teniendo en cuenta el conjunto de datos de entrenamiento original, MS COCO 2017. Si se hubiera hecho sobre alguno de los conjuntos de datos utilizados en este proyecto el resultado no hubiera

sido coherente, ya que la red a la que se hubiera podido llegar podría ser diferente a YOLOv5 y a YOLO11.

La estimación de todos los entrenamientos hasta alcanzar la arquitectura final supone, para el modelo pequeño de YOLOv5, 12.825,00 kg CO_2 eq. y teniendo en cuenta que las emisiones al haberlo utilizado sobre el primer conjunto de datos son de 0,06 , descritas en la tabla 4.13, la búsqueda de la red hubiera supuesto 213.750 veces las emisiones de hacer transfer learning. Para YOLO11, modelo *s*, el resultado estimado es de 285.000 veces.

Capítulo 5

Conclusiones

Por último, este capítulo presenta las conclusiones extraídas de la realización del proyecto así como las posibles siguientes fases del mismo.

5.1. Conclusiones

El objetivo principal de este proyecto ha sido el de buscar nuevas formas de beneficiarse de los últimos avances tecnológicos para mejorar la seguridad de los trabajadores en entornos industriales y, a la vez, buscar técnicas cuya huella de carbono tenga un impacto reducido. Además, se ha evaluado el rendimiento de un modelo de última generación (YOLO11) en comparación con uno ya establecido (YOLOv5), con el fin de determinar si el aumento en el uso de recursos y tiempo es justificable en términos de los beneficios obtenidos. Por otra parte se ha buscado poder ofrecer una solución basada en inteligencia artificial que, de ser implementada, proporcionara una ventaja competitiva a la empresa, redujera costes y minimizara el time to market.

Después del análisis efectuado, se puede concluir que, utilizando la técnica de transfer learning, se han cumplido con éxito todos los objetivos para al menos uno de los conjuntos de datos utilizados, concretamente para el primer conjunto de datos.

Mediante experimentación se han encontrado los parámetros que maximizan el rendimiento de los diferentes modelos de YOLO utilizados a la vez que se han monitorizado los tiempos de ejecución para encontrar aquellos parámetros que, produciendo el mismo rendimiento, tengan un impacto reducido en el medio ambiente. Se ha comprobado que, cuando se entrena el modelo entero sobre el nuevo conjunto de datos, los tiempos de ejecución y las emisiones de CO_2 son mayores que cuando se hace sobre un modelo con capas congeladas. El rendimiento es menor en el segundo caso, no obstante, dado que la

reducción es sólo del 1,4 % se ha procedido a utilizar modelos con capas congeladas como base para el resto de experimentos.

Una de las ventajas que se ha encontrado que proporciona YOLO es que la red está programada de manera que el tamaño de batch no afecte al rendimiento del modelo, lo que permite utilizar el mayor tamaño de batch que admita la memoria de la tarjeta gráfica. Se ha encontrado que multiplicar por tres el tamaño de batch reduce en un 66 % las emisiones de CO_2 manteniendo prácticamente el mismo rendimiento, lo que implica que optimizando la red se puede conseguir una solución de inteligencia artificial en menos tiempo y reduciendo la huella de carbono. Asimismo, esta reducción del tiempo de entrenamiento permite que, de ser una solución viable, se reduzca el tiempo que tarda en implementarse y pueda proporcionar una ventaja competitiva a la empresa. Entre otros experimentos de optimización se ha utilizado un algoritmo genético que, si bien ha mejorado el rendimiento del modelo, se ha encontrado que requiere de un uso intensivo de GPU, por lo que su utilización dependerá del caso de uso.

En relación al segundo conjunto se ha observado que, probablemente debido a la gran diferencia que existe entre las clases de este conjunto y las clases del conjunto de entrenamiento, el rendimiento de los diversos modelos no es bueno y las técnicas de optimización utilizadas no consiguen mejorarlo significativamente. Aun así se ha comprobado que las conclusiones extraídas en el primer conjunto sobre la reducción de emisiones de CO_2 se mantienen para este segundo.

Asimismo, se he hecho una comparación entre el modelo más reciente, YOLO11, y YOLOv5, un modelo ya establecido en la detección de objetos. Este análisis ha evaluado si el mayor tiempo de entrenamiento en algunos modelos de YOLO11 y el aumento de recursos están justificados por un incremento en rendimiento. Se ha observado que, de media, YOLO11 presenta una mejora en mAP de 3,5-6,7 % mientras el tiempo puede llegar a eser un 8 % mayor y un uso más intensivo de la GPU(50 % más que el homólogo de YOLOv5). Además, en algunos casos, el consumo de energía ha sido un 21 % mayor para una mejora del 5 %. Por lo tanto, la elección de una arquitectura más moderna dependerá de la necesidad de negocio de obtener métricas más exactas a cambio de un gasto mayor.

Por último, se ha calculado tanto el tiempo como las emisiones que se hubieran dado de no utilizar la técnica de transfer learning y haber creado la red desde cero. Este análisis implica que al hacer uso de una red preentrenada las emisiones han sido 213.750 veces menores en el caso de YOLOv5 y 285.000 para YOLO11 , lo que confirma que el uso de esta técnica permite cumplir los objetivos propuestos para este proyecto. Con ello se ha confirmado que se puede hacer uso de los últimos avances tecnológicos, proporcionando ventajas competitivas en forma de aplicación de software de IA pero acercándonos al compromiso con las 0 emisiones que persigue REPSOL.

5.2. Trabajos futuros

En primer lugar, existen varios modelos de YOLO, además de los utilizados en este proyecto, tal y como muestra la figura 5.1. Como siguiente paso, se podría analizar la relación entre eficiencia y emisiones comparando YOLO11, en este caso el modelo más reciente, y otros modelos más antiguos pero más recientes que YOLOv5. De esta manera, se podría determinar si las conclusiones extraídas en este proyecto son válidas y generalizables a otros modelos de la familia YOLO.

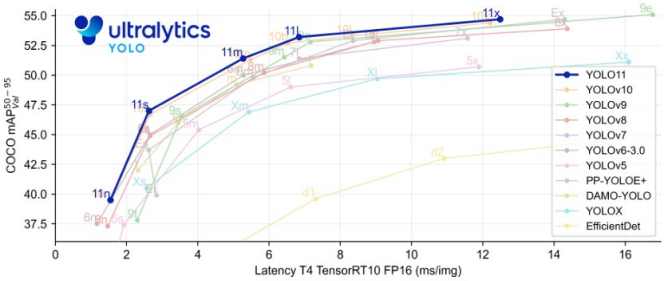


Figura 5.1: Comparación entre EfficientDet y los nuevos modelos de YOLOv5 en términos de velocidad y AP[21].

En cuanto al segundo conjunto,imágenes de defectos en acero laminado en caliente, los resultados no han sido favorables por las razones explicadas más arriba. Este conjunto fue elegido, en un principio, para efectuar una comparación del rendimiento de la red en dos conjuntos de datos que corresponden a situaciones de tipo industrial pero ambas muy diferentes. Si se quisiera seguir por esta ruta más allá de la comparación, habría que plantearse si sería necesario la adición de más capas para mejorar el rendimiento o si, por el contrario, la red no es la idónea para este conjunto de datos y sería beneficioso cambiar de red.

De forma general las redes neuronales de uso público están entrenadas sobre conjuntos de imágenes como Microsoft COCO, ImageNet o Pascal VOC, entre otros. Dichos conjuntos contienen imágenes de objetos cotidianos que se alejan de los objetos que se pueden encontrar en entornos de tipo industrial. Como consecuencia es posible que empresas que necesiten soluciones para este tipo de entornos tengan que crear desde cero tanto su propio conjunto de datos como la red neuronal. No obstante, esto es una tarea que consume una gran cantidad de tiempo, que supone un alto coste económico y que produce un impacto medioambiental que a día de hoy no está monitorizado. Una posible solución sería la creación de consorcios o asociaciones entre empresas de sectores diferentes, o del mismo sector pero sin ser competición directa, para el desarrollo de inteligencia artificial enfocada al sector de interés. En 2021 se creó IndesIA, el primer consorcio de inteligencia artificial de la industria en España, del cual forma parte REPSOL, que pretende impulsar el uso de los datos y de la IA en las empresas industriales españolas[109]. Este tipo de iniciativas podrían impulsar la creación de redes neuronales

adaptadas a la industria que produjeran buenos resultados a la vez que se reparten los costes mencionados.

Apéndice A

Clases MS COCO

ID	OBJECT (PAPER)	OBJECT (2017 REL.)	SUPER CATEGORY
1	person	person	person
2	bicycle	bicycle	vehicle
3	car	car	vehicle
4	motorcycle	motorcycle	vehicle
5	airplane	airplane	vehicle
6	bus	bus	vehicle
7	train	train	vehicle
8	truck	truck	vehicle
9	boat	boat	vehicle
10	traffic light	traffic light	outdoor
11	fire hydrant	fire hydrant	outdoor
12	street sign	-	outdoor
13	stop sign	stop sign	outdoor
14	parking meter	parking meter	outdoor
15	bench	bench	outdoor
16	bird	bird	animal
17	cat	cat	animal
18	dog	dog	animal
19	horse	horse	animal
20	sheep	sheep	animal
21	cow	cow	animal
22	elephant	elephant	animal
23	bear	bear	animal
24	zebra	zebra	animal
25	giraffe	giraffe	animal
26	hat	-	accessory
27	backpack	backpack	accessory

28	umbrella	umbrella	accessory
29	shoe	-	accessory
30	eye glasses	-	accessory
31	handbag	handbag	accessory
32	tie	tie	accessory
33	suitcase	suitcase	accessory
34	frisbee	frisbee	sports
35	skis	skis	sports
36	snowboard	snowboard	sports
37	sports ball	sports ball	sports
38	kite	kite	sports
39	baseball bat	baseball bat	sports
40	baseball glove	baseball glove	sports
41	skateboard	skateboard	sports
42	surfboard	surfboard	sports
43	tennis racket	tennis racket	sports
44	bottle	bottle	kitchen
45	plate	-	kitchen
46	wine glass	wine glass	kitchen
47	cup	cup	kitchen
48	fork	fork	kitchen
49	knife	knife	kitchen
50	spoon	spoon	kitchen
51	bowl	bowl	kitchen
52	banana	banana	food
53	apple	apple	food
54	sandwich	sandwich	food
55	orange	orange	food
56	broccoli	broccoli	food
57	carrot	carrot	food
58	hot dog	hot dog	food
59	pizza	pizza	food
60	donut	donut	food
61	cake	cake	food
62	chair	chair	furniture
63	couch	couch	furniture
64	potted plant	potted plant	furniture
65	bed	bed	furniture
66	mirror	-	furniture
67	dining table	dining table	furniture
68	window	-	furniture
69	desk	-	furniture
70	toilet	toilet	furniture

71	door	-	furniture
72	tv	tv	electronic
73	laptop	laptop	electronic
74	mouse	mouse	electronic
75	remote	remote	electronic
76	keyboard	keyboard	electronic
77	cell phone	cell phone	electronic
78	microwave	microwave	appliance
79	oven	oven	appliance
80	toaster	toaster	appliance
81	sink	sink	appliance
82	refrigerator	refrigerator	appliance
83	blender	-	appliance
84	book	book	indoor
85	clock	clock	indoor
86	vase	vase	indoor
87	scissors	scissors	indoor
88	teddy bear	teddy bear	indoor
89	hair drier	hair drier	indoor
90	toothbrush	toothbrush	indoor
91	hair brush	-	indoor

Apéndice B

Hiperparámetros por defecto

Hiperparámetro	Valor	Descripción
lr0	0.01	initial learning rate (SGD=1E-2, Adam=1E-3)
lrf	0.2	final OneCycleLR learning rate (lr0 * lrf)
momentum	0.937	SGD momentum/Adam beta1
weight_decay	0.0005	optimizer weight decay 5e-4
warmup_epochs	3.0	warmup epochs (fractions ok)
warmup_momentum	0.8	warmup initial momentum
warmup_bias_lr	0.1	warmup initial bias lr
box	0.05	box loss gain
cls	0.5	cls loss gain
cls_pw	1.0	cls BCELoss positive_weight
object	1.0	obj loss gain (scale with pixels)
object_pw	1.0	obj BCELoss positive_weight
iou_t	0.20	IoU training threshold
anchor_t	4.0	anchor-multiple threshold
fl_gamma	0.0	focal loss gamma (efficientDet default gamma=1.5)
hsv_h	0.0015	image HSV-Hue augmentation (fraction)
hsv_s	0.7	image HSV-Saturation augmentation (fraction)
hsv_v	0.4	image HSV-Value augmentation (fraction)
degrees	0.0	image rotation (+/- deg)
translate	0.1	image translation (+/- fraction)
scale	0.5	image scale (+/- gain)
shear	0.0	image shear (+/- deg)
perspective	0.0	image perspective (+/- fraction), range 0-0.001
flipud	0.0	image flip up-down (probability)
fliplr	0.5	image flip left-right (probability)
mosaic	1.0	image mosaic (probability)
mixup	0.0	image mixup (probability)

Bibliografía

[1] F. Rosenblatt, "Marki perceptron manual," *Cornell Aeronautical Laboratory.*, 1957.

[2] B. Widrow, "Adaptive 'adaline' neuron using chemical "memistors"," *Stanford University. Stanford Electronics Laboratories. Solid State Electronics Laboratory.*, 1960.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Object recognition with gradient-based learning," *Proceedings of the IEEE*, vol. 86, 1998.

[4] T. Hoeser and C. Kuenzer, "Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends," *Remote Sens*, vol. 12, 2020.

[5] R. Demush, "A brief history of computer vision (and convolutional neural networks)." February 2019.

[6] L. Roberts, *Machine Perception Of Three-Dimensional Solids*. PhD thesis, January 1963.

[7] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, January 1988.

[8] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," 2023.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," IEEE, 2005.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," pp. 580–587, 2014.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," vol. 2016-December, 2016.

[12] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," IEEE, June 2020.

[13] D. Sarkar, "A comprehensive hands-on guide to transfer learning with real-world applications in deep learning." November 2018.

[14] P. Marcellino, "Transfer learning from pre-trained models." October 2018.

[15] G. Jocher, "Ultralytics yolov5," 2020.

[16] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[17] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," *Forests*, vol. 12, p. 217, February 2021.

[18] R. Padilla, S. L. Netto, E. da Silva, and E. A. B., "A survey on performance metrics for object-detection algorithms," pp. 237–242, 2020.

[19] C. Dilmegani, "Machine learning accuracy: True vs. false positive/negative." Available: `https://research.aimultiple.com/machine-learning-accuracy`. Accessed on: 22-04-2021 [Online].

[20] J. Solawetz, "What is mean average precision (map) in object detection?." Accessed on: 15-04-2021 [Online], May 2020.

[21] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024.

[22] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," 2020.

[23] Ultralytics, "Yolov11 documentation," 2024. Accessed: 2025-01-7.

[24] M. de Trabajo y Economía social, "Estadística de accidentes de trabajo. 2023," tech. rep., Instituto Nacional de Seguridad y Salud en el Trabajo, 2023.

[25] REPSOL, "Las personas son el foco de nuestro compromiso con la seguridad," 2021. Accessed on: 30-02-2021 [Online].

[26] S. Deekshith, J. Varma, S. Navi, and M. R. Ahmed, "Diving deep into deep learning: History, evolution, types and applications," *The International Journal on Media Management*, vol. Volume-9, pp. 2278–3075, January 2020.

[27] Y. Bengio, I. Goodfellow, and A. Courville, *Deep Learning*. 2016.

[28] H. Shekhar, S. Seal, S. Kedia, and A. Guha, *Survey on Applications of Machine Learning in the Field of Computer Vision*, pp. 667–678. January 2020.

[29] J. Dean and A. Ng, "Using large-scale brain simulations for machine learning and a.i.." June 2012.

[30] S. Bozinovski, "Reminder of the first paper on transfer learning in neural networks, 1976," *Informatica (Slovenia)*, vol. 44, 2020.

[31] G. Dhillon, "Think every business should become a software business? think again," February 2018.

[32] M. Holmes, "Microsoft ceo: "every company is now a software company"," February 2019.

[33] G. H. et al., "Deep neural networks for acoustic modeling in speech recognition," 2021.

[34] M. B. et al., "End to end learning for self-driving cars," 2016.

[35] S. Balaban, "Deep learning and face recognition: the state of the art," *Biometric and Surveillance Technology for Human and Activity Identification XII*, May 2015.

[36] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," *Integration*, vol. 69, August 2019.

[37] D. Cazzato, C. Cimarelli, J. L. Sanchez-Lopez, H. Voos, and M. Leo, "A survey of computer vision methods for 2d object detection from unmanned aerial vehicles," *Journal of Imaging*, vol. 6, no. 8, 2020.

[38] A. M. et al., "Survey on artificial intelligence," *International Journal of Computer Sciences and Engineering*, vol. 7, pp. 1778–1790, May 2019.

[39] W. S. McCulloch and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull Math Biol. 1990*, 1943.

[40] F. Rosenblatt, "The perceptron, a perceiving and recognizing automaton (PROJECT PARA)," *Cornell Aeronautical Laboratory.*, 1957.

[41] M. Minsky and S. Papert, "Perceptrons: an introduction to computational geometry," *MA: MIT Press, Cambridge*, 1969.

[42] A. Ivakhnenko and V. G. Lapa, "Cybernetics and forecasting," *Nature*, vol. 219, 1968.

[43] H. J. Kelley, "Gradient theory of optimal flight paths," *ARS Journal*, vol. 30, pp. 947–954, 1960.

[44] S. Linnainmaa, *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors.* PhD thesis, 1970.

[45] S. Dreyfus, "The numerical solution of variational problems," *Journal of Mathematical Analysis and Applications*, vol. 5(1), pp. 30–45, 1962.

[46] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, January 2015.

[47] P. J. Werbos, *Applications of advances in nonlinear sensitivity analysis. In R. Drenick, F. Kozin, (eds): System Modeling and Optimization: Proc. IFIP.* Springer, 1982.

[48] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[49] S. Hochreiter, *Untersuchungen zu dynamischen neuronalen Netzen (Diplom thesis).* PhD thesis, 1991.

[50] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks,*, vol. 5, pp. 157–166, 1994.

[51] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–1780, 1997.

[52] K. S. Oh and K. Jung, "Gpu implementation of neural networks," *Pattern Recognition*, vol. 37, pp. 1311–1314, 2004.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[54] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," pp. 807º814–undefined, 2010.

[55] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," pp. 448–456, 2015.

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and Y. Bengio, "Generative adversarial nets," pp. 2672–2680, 2014.

[57] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[58] "Ilsvrc2012 results." https://image-net.org/challenges/LSVRC/2012/results.html, 2012. Accessed: 31-05-2021.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[60] N. N. I. of Standards and Technology), "Fiftieth anniversary of first digital image marked," *News, NIST, May 24. Accessed 2021-05-31.*, 2007.

[61] D. Hubel and T. Wiesel, "Receptive fields of single neurons in the cat's striate cortex," 1959.

[62] S. Papert, "The summer vision project." July 1966.

[63] Z. Zdziarski, "The early history of computer vision." https://zbigatron.com/the-early-history-of-computer-vision/, July 2018. Accessed: 31-05-2021.

[64] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman and Company, 1982.

[65] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," vol. 81, April 1981.

[66] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, November 1986.

[67] M. Slavkovic and D. Jevtic, "Face recognition using eigenface approach," *Serbian Journal of Electrical Engineering*, vol. 9, 2012.

[68] A. Linn, "Microsoft researchers win imagenet computer vision challenge." https://blogs.microsoft.com/ai/microsoft-researchers-win-imagenet-computer-vision-challenge/, 2015. Accessed on: 31-05-2021 [Online].

[69] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, pp. 1–3, June 2001.

[70] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," 2008.

[71] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," pp. 2241–2248, 2010.

[72] T. Malisiewicz, A. Gupta, and A. Efros, "Ensemble of exemplar-svms for object detection and beyond," pp. 89–96, 2011.

[73] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, pp. 1627–1645, 2010.

[74] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," pp. 1097–1105, 2012.

[75] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," pp. 346–361, Springer, 2014.

[76] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, p. 1440–1448, 2015.

[77] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[78] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Springer*, pp. 21–37, 2016.

[79] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, 2020.

[80] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019.

[81] N. Carion, F. Massa, G. Synnaeve, N. Usunier, N. Usunier, and S. Zagoruyko, "End-to-end object detection with transformers," May 2020.

[82] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," August 2021.

[83] X. Chen, F. Wei, G. Zeng, and J. Wang, "Conditional detr v2: Efficient detection transformer with box queries," 2022.

[84] W. Lv, Y. Zhao, Q. Chang, K. Huang, G. Wang, and Y. Liu, "Rt-detrv2: Improved baseline with bag-of-freebies for real-time detection transformer," 2024.

[85] S. Bozinovski and A. Fulgosi, "The influence of pattern similarity and transfer learning upon the training of a base perceptron b2," pp. 3–121, 1976.

[86] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," vol. 4, 2014.

[87] Larxel, "Safety helmet detection." Available: `https://www.kaggle.com/andrewmvd/hard-hat-detection`. Accessed on: 05-02-2021 [Online].

[88] H. Dong, K. Song, Y. He, J. Xu, Y. Yan, and Q. Meng, "Pga-net: Pyramid feature fusion and global context attention network for automated surface defect detection," *IEEE Transactions on Industrial Informatics*, vol. 16, 2020.

[89] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[90] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," January 2017.

[91] J. Redmon and A. Farhadi, "Yolo v3.0: An incremental improvement," *Tech report*, 2018.

[92] M. Maithani, "Guide to yolov5 for real-time object detection." December 2020.

[93] M. Maithani, "Yolo creator joseph redmon stopped cv research due to ethical concerns." February 2020.

[94] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[95] J. Solawetz, "Yolov5 new version - improvements and evaluation." Accessed on: 09-03-2021 [Online], June 2020.

[96] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[97] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[98] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time flying object detection with yolov8," *arXiv preprint arXiv:2305.09972*, 2023.

[99] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," *arXiv preprint arXiv:2405.14458*, 2024.

[100] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[101] M. Zhu, "Recall, precision and average precision," *University of Waterloo*, September 2004.

[102] W. . Biases, "The weights & biases guide to building a tehc stack for ml expertise." 2020.

[103] L. Biewald, S. Lewis, and C. V. Pelt, "Weights & biases." Available: `https://wandb.ai/site`.

[104] K. Brownlee, "How to control the stability of training neural networks with the batch size," *Machine Learning Mastery*, January 2019.

[105] L. Biewald, "Deep learning and carbon emissions." Available: `https://towardsdatascience.com/deep-learning-and-carbon-emissions-79723d5bc86e`. Accessed on: 21-06-2021 [Online].

[106] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," *arXiv preprint arXiv:1910.09700*, 2019.

[107] E. Strubell, P. Verga, D. Andor, D. Weiss, and A. McCallum, "Linguistically-informed self-attention for semantic role labeling," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 5027–5038, Association for Computational Linguistics, October 2018.

[108] J. Redmon, "darknet." Available: `https://github.com/pjreddie/darknet`. Accessed on: 20-06-2021 [Online].

[109] REPSOL, "Seis grandes empresas crean el primer consorcio de inteligencia artificial de la industria en españa." June 2021.