Sistema de detección y clasificación de residuos en plantas de selección de la fracción resto

Detección de cartón y plástico

Zaragoza, enero de 2023

Patricia Lázaro Tello

Tutor: Raúl Parada Medina
Profesor: Jordi Casas Roma

Trabajo de Fin de Máster, Área 2
Máster Universitario de Ciencia de Datos, 2022/2023

Ficha del trabajo final

Título del trabajo:	Sistema de detección y clasificación de residuos en plantas de selección de la fracción resto: Detección de cartón y plástico
Nombre del autor:	Patricia Lázaro Tello
Nombre del colaborador/a docente:	Raúl Parada Medina
Nombre del PRA:	Jordi Casas Roma
Fecha de entrega (mm/aaaa):	10/2022
Titulación o programa:	Máster Universitario de Ciencia de Datos
Área del Trabajo Final:	Área 2
Idioma del trabajo:	Español
Palabras clave:	*municipal solid waste* *waste sorting* *computer vision* *deep learning* *object detection* *image recognition*

Dedicatoria

A mi padres, por todo su amor y apoyo incondicional en cada paso que doy.

A mi pareja, por acompañarme en este camino, los que vinieron y los que quedan por recorrer.

A mi hermana, por creer en mí cuando ni yo lo hacía.

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor Raúl Parada Medina por su apoyo y por estar siempre disponible para resolver cualquier duda.

También me gustaría agradecer a la Universitat Oberta de Catalunya y el equipo docente por su atención y la calidad de la enseñanza de la titulación.

Por último, agradecer a Alexandra Elbakyan, fundadora de Sci-Hub; este trabajo no hubiera sido posible sin el acceso a este repositorio universal y gratuito de conocimiento.

Resumen

En las últimas décadas, el cambio climático se ha visto agravado por la industrialización y quema de combustibles fósiles. Para frenarlo y revertirlo en la medida de lo posible, han surgido una serie de iniciativas entre las que se encuentra el reciclaje, que busca reducir la quema de combustibles fósiles y la explotación de los recursos naturales, dándole una segunda vida a los residuos que anteriormente terminaban en un vertedero o en el océano.

Dentro de este contexto, los datos de España en materia de reciclaje y tratamiento de residuos son consistentemente inferiores a los de la Unión Europea; de ahí surge la motivación de este proyecto, cuyo anhelo es que una mejora en el proceso de categorización de residuos suponga un impulso al esfuerzo de reciclaje y recuperación de energía comunitario.

Para la consecución del objetivo, se plantea el desarrollo de un sistema de visión por computador instalado sobre la cinta transportadora de la planta de selección dedicada a la fracción resto que lleve a cabo tareas de recategorización de residuos mal clasificados.

Respecto al módulo de detección y clasificación, se contempla el uso de diferentes técnicas de *Deep Learning*; concretamente se compara el rendimiento de uno de los modelos de detección de objetos de una pasada más populares, SSD (Wei Liu y col. 2015) y el modelo más popular de dos etapas, *Faster* R-CNN (Shaoqing Ren y col. 2015). También se plantea la creación de modelos híbridos de DL+ML.

Palabras clave: *municipal solid waste*, *waste sorting*, *computer vision*, *deep learning*, *object detection*, *image recognition*

Abstract

In the last decades, climate change has been negatively influenced by industrialization and burning of fossil fuels. In order to stop it, and hopefully revert it, a wide array of initiatives has emerged, recycling being one of them. Recycling seeks to reduce the burning of fossil fuels and natural resources exploitation, giving a second life to solid waste, which used to end up in landfills and oceans.

In this context, Spain's figures in recycling and solid waste treatment are consistently below the European Union's; the motivation for this project stems from the hope that an improvement in categorization of solid waste will have a positive effect in the recycling and energy recovery communitary effort.

In order to meet the objective, a computer vision system installed over the conveyor belt of disposal facilities to recategorize wrongly classified solid waste is proposed.

Regarding the detection and classification module, different Deep Learning techniques are considered; specifically, the performance of one of the most well-known one-stage object detectors, SSD (Wei Liu y col. 2015) and the most popular two-stage detector, Faster R-CNN (Shaoqing Ren y col. 2015) is compared. Additionally, this thesis proposes a hybrid DL+ML model.

Keywords: *municipal solid waste*, *waste sorting*, *computer vision*, *deep learning*, *object detection*, *image recognition*

Índice

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto y justificación del Trabajo

El **cambio climático** (Naciones Unidas 2022a) se refiere a la variación de las temperaturas y patrones del clima a lo largo del tiempo en una escala global. A *priori*, esta definición no distingue entre las causas naturales (como las edades de hielo) y no-naturales, o más concretamente, las actividades humanas.

En las últimas décadas, se han acuñado las actividades humanas como principal causa del cambio climático, que se ha visto acelerado exponencialmente desde el siglo XIX con la industrialización y la quema de combustibles fósiles. Actualmente, las principales causas del aumento de temperatura (Naciones Unidas 2022b) son el uso de combustibles fósiles (para generar energía, en la industria, en el transporte...), la tala de bosques y un consumo excesivo.

En este contexto, el papel del **reciclaje** es uno de reducción de la dependencia en combustibles fósiles, reutilización y reducción de recursos. Los residuos se pueden utilizar para crear productos nuevos o convertirlos en energía (biomasa) mediante su combustión.

Mediante la **segregación de los residuos** por tipo de material se posibilita el resto del proceso de reciclaje, es decir, su reutilización como materia prima o la recuperación de energía. Sin esta fase de separación inicial, la reutilización y recuperación son imposibles y desemboca en el traslado de los residuos a vertederos.

Los datos de España en materia de reciclaje y tratamiento de residuos son consistentemente inferiores a los de la Unión Europea (ver figura 1.1); si bien su evolución fue rápida hasta los 2.000, en los últimos 20 años no se ha conseguido impulsar el tratamiento de residuos por encima del 50 %. Esto supone que la mitad de los residuos generados acaban en vertederos (ver figura 1.2), contribuyendo activamente (generan metano) y de forma pasiva (se impide la reutilización y recuperación de energía de los residuos) a la aceleración del cambio climático.

Frente a la Unión Europea (ver figura 1.3), la evolución de España en materia de reciclaje y recuperación de energía es trabajosa y lenta, señalando especialmente la sección de recuperación de energía y uso de biomasa, que apenas crece respecto al resto de la Unión.

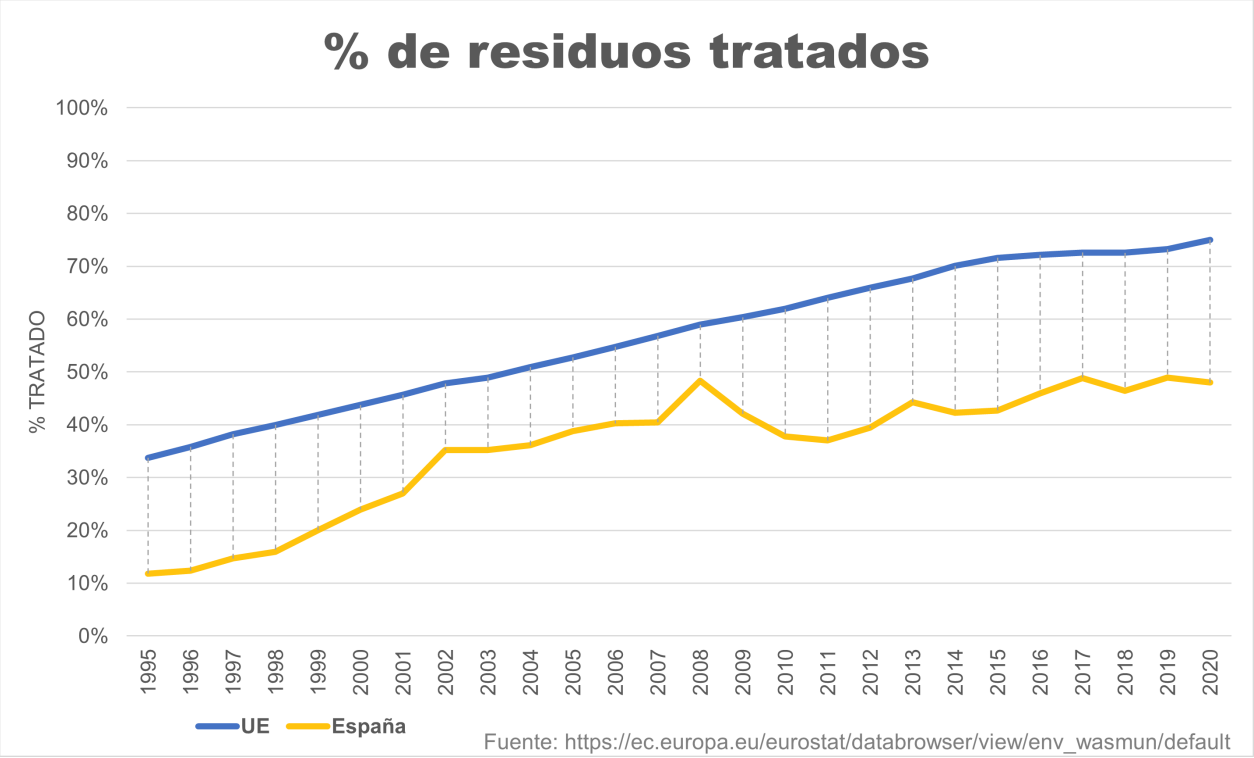


Figura 1.1: España trata alrededor de un 25 % menos de residuos que la media europea
Fuente: elaboración propia a partir de los datos de (Eurostat 2020)

De esta situación nace la necesidad de la autora de este proyecto de creación de un sistema de detección de residuos en las plantas de selección con la esperanza de que una mejora en el proceso de categorización de residuos suponga un impulso a las cifras de reciclaje y recuperación de energía y contribuya a frenar el cambio climático.

1.2. Objetivos del Trabajo

El marco en el que se incluye este trabajo tiene como finalidad mejorar la segregación de residuos. Para ello, el presente proyecto tiene como objetivo desarrollar un sistema de **detección y categorización de residuos** en el contexto de plantas de selección.

Concretamente, el trabajo plantea un sistema de visión por computador instalado sobre la cinta transportadora de la planta de selección dedicada a la fracción resto (Ministerio para la transición ecológica y el reto demográfico 2022) que realice una primera categorización más general para trasladar los residuos a las plantas de selección apropiadas.

Cabe destacar que aunque la denominada fracción resto hace referencia a los residuos que se obtienen tras separar papel, plástico, vidrio, metal y materia orgánica, también es el destino de las anteriores categorías si no existe un sistema de recogida separada. El ejemplo más común es el de la materia orgánica: hay zonas donde no se separan los residuos orgánicos del resto (el llamado “todo en uno”) y por tanto las plantas de selección de la fracción resto encuentran basura mal clasificada.

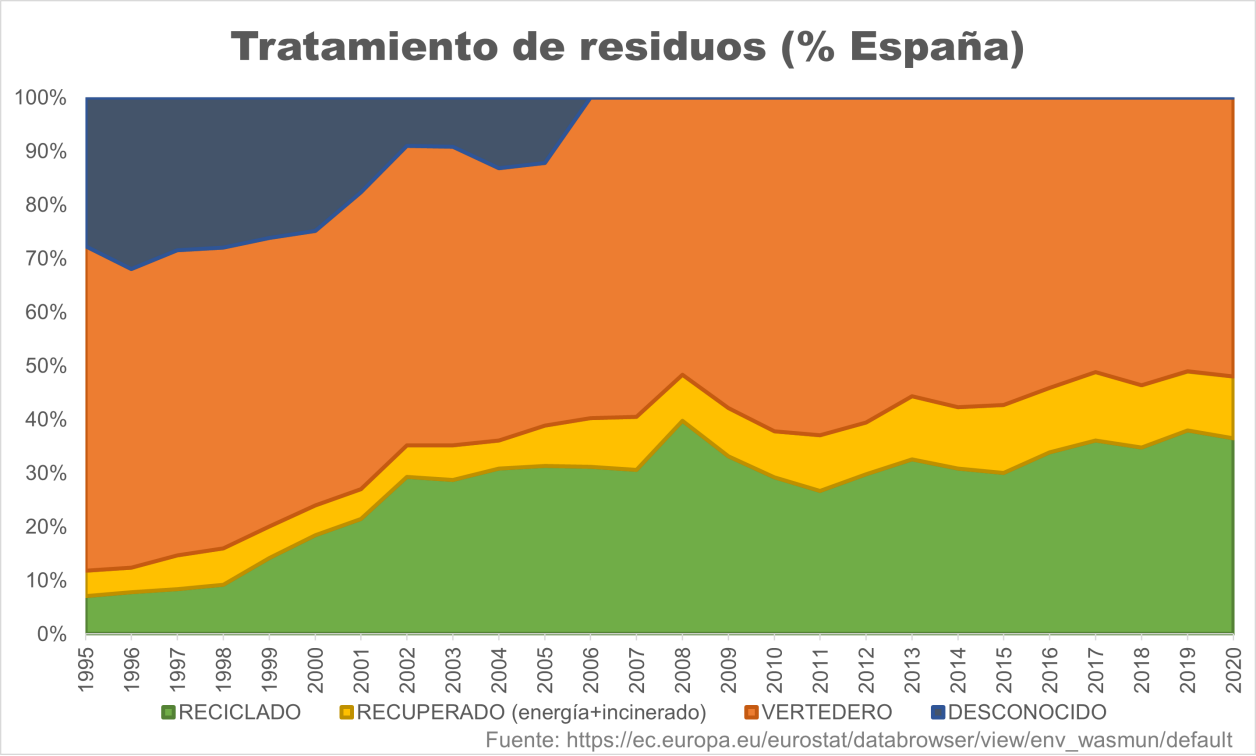


Figura 1.2: La mitad de la basura de España termina en vertederos
Fuente: elaboración propia a partir de los datos de (Eurostat 2020)

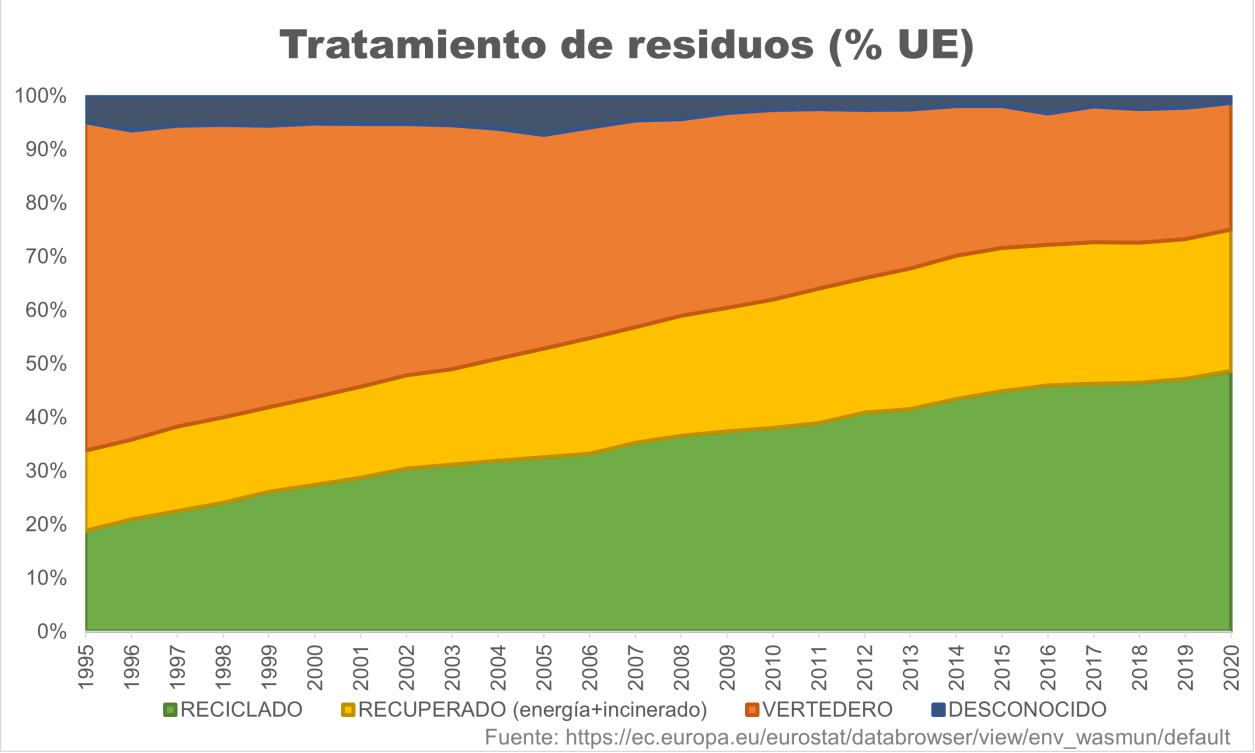


Figura 1.3: La Unión Europea recupera la energía del 30 % de sus residuos
Fuente: elaboración propia a partir de los datos de (Eurostat 2020)

Así, el proyecto propuesto abre la posibilidad de volver a categorizar residuos mal clasificados, permitiendo su reutilización, reciclaje y recuperación, y disminuyendo la cantidad de basura enviada a vertederos. Idealmente, el sistema de visión por computador planteado se vería acompañado de un sistema robótico para la retirada y recategorización efectiva de los residuos.

Finalmente, el proyecto se ha centrado en torno a la **detección y recategorización de papel, cartón y plástico** en exclusiva dada la ausencia de datos de calidad del resto de categorías. Se ha primado la calidad sobre la cantidad siguiendo la filosofía de *Garbage In, Garbage Out*, es decir, que los datos de baja calidad no aportan nada al modelo. El razonamiento detrás de esta decisión se abordará en mayor profundidad en el capítulo Selección del conjunto de datos.

1.3. Impacto en sostenibilidad, ético-social y diversidad

El presente trabajo tiene una relación directa con la **sostenibilidad**, buscando impactar de forma positiva en las tasas de reciclaje y recuperación de residuos. Una mejor clasificación de los residuos contribuye a potenciar la energía de biomasa, reducir los residuos vertidos y disminuir la cantidad de recursos naturales explotados. También promueve una producción y consumo más responsables como posible consecuencia. Por todo lo anterior, el resultado del trabajo fomenta comunidades más sostenibles, con menos emisiones de carbono y menor uso de recursos.

Indirectamente, se relaciona de forma positiva con la dimensión de **diversidad y derechos humanos**: una mejora de la eficiencia en la segregación de residuos permite a los procesos de reciclaje y recuperación de energía funcionar a más capacidad, fomentando la creación de puestos de trabajo en las empresas directamente relacionadas con dichos procesos.

El impacto en el **comportamiento ético y la responsabilidad social** es positivo, aunque muy inferior al impacto del trabajo en las dos dimensiones anteriores. El impulso a la industria local explicado en la dimensión de diversidad y derechos humanos contribuye a aumentar la independencia del país en materia de recursos, reduciendo así las desigualdades entre países.

Por último cabe remarcar que, teniendo el resultado del trabajo un impacto positivo en sostenibilidad, materia ético-social y diversidad, la realización del mismo incorpora una huella ecológica negativa: el desarrollo de un modelo de detección y clasificación de residuos conlleva una fase de optimización y búsqueda de hiperparámetros que hace uso intensivo en los recursos de procesamiento y como consecuencia produce un consumo de energía significativo.

1.4. Enfoque y método seguido

La metodología que guía el proceso de desarrollo de este trabajo es **CRISP-DM** (Pete Chapman y col. 2000), que es el estándar de facto para el desarrollo de proyectos de minería de datos. Esta metodología ofrece flexibilidad para navegar por sus 6 fases, permitiendo y fomentando las iteraciones para descubrir conocimiento escondido en los datos. Las fases son las que siguen:

1. Comprensión del negocio
2. Comprensión de los datos
3. Preparación de los datos
4. Fase de modelado
5. Evaluación
6. Implementación

Figura 1.4: El resultado del trabajo impacta de forma positiva en 9 de los 17 ODS (ver Glosario)
Fuente: Naciones Unidas 2022c

No obstante, la metodología CRISP-DM también tiene algunas desventajas que afectan a este proyecto, como es la asunción de que se disponen de datos de calidad. A este respecto, dado que se ha de realizar previamente una tarea de descubrimiento y obtención de datos, se ha optado por considerar esta tarea como un proyecto aparte.

1.5. Planificación del trabajo

La planificación del trabajo (ver figuras 1.6 y 1.5) ha sido desarrollada utilizando un diagrama de Gantt estructurado en cuatro grandes bloques, que siguen una estructura similar a la planteada en los criterios de evaluación. Cada bloque a su vez se subdivide en tareas, y estas se desgranan en subtareas cuando corresponda. El cronograma muestra también el trabajo previo realizado para la puesta en marcha del proyecto.

La planificación, incluyendo el trabajo previo desarrollado, supone un total de casi ocho meses de trabajo. A continuación se describen los bloques, tareas y subtareas definidos en la planificación:

- **Definición del proyecto**: comprende el trabajo englobado en la PEC 1 (Definición y planificación) y la PEC2 (Estado del arte), así como el trabajo previo de investigación de líneas de trabajo.

  En este bloque se han incluido dos tareas correspondientes al Estado del Arte (investigación y revisión) debido a que una parte de la investigación del Estado del Arte se realizó como trabajo previo como requisito indispensable para terminar de definir la línea de trabajo que seguiría el proyecto.

TAREA	PROGRESO	INICIO	FIN
Definición del proyecto	80%	14-6-22	20-10-22
Investigación de líneas de trabajo	100%	14-6-22	28-6-22
Investigación del estado del arte	100%	29-6-22	11-7-22
Revisión del estado del arte	0%	10-10-22	20-10-22
Definición de la propuesta final	100%	15-9-22	29-9-22
Planificación	100%	29-9-22	9-10-22
Obtención de datos	70%	12-7-22	16-10-22
Ronda de contactos	100%	12-7-22	28-7-22
Exploración de *datasets* iniciales	100%	12-7-22	26-7-22
EXP Anotación semisupervisada	100%	27-7-22	7-9-22
Exploración de *datasets* adicionales	50%	22-9-22	16-10-22
Diseño e implementación	0%	21-10-22	24-12-22
Exploración del *dataset* elegido	0%	21-10-22	3-11-22
Limpieza y transformación	0%	21-10-22	26-10-22
Integración y formato	0%	27-10-22	3-11-22
Selección de técnicas de modelado	0%	4-11-22	8-11-22
Diseño de la evaluación	0%	9-11-22	13-11-22
Construcción de modelos	0%	14-11-22	4-12-22
Evaluación de modelos	0%	5-12-22	20-12-22
Documentación	0%	15-12-22	24-12-22
Memoria, preparación y defensa	0%	26-12-22	3-2-23
Redacción de la memoria	0%	26-12-22	9-1-23
Preparación de la defensa	0%	10-1-23	23-1-23
Defensa pública	0%	23-1-23	3-2-23
DÍAS TOTALES INVERTIDOS			**234**

Figura 1.5: Bloques y tareas de la planificación

- **Obtención de datos**: comprende parte del trabajo englobado en la PEC 3 (Diseño e implementación). La mayor parte del esfuerzo se ha realizado como trabajo previo ya que, como se apuntaba en la sección 1.4, se ha decidido considerar la obtención de datos de calidad como un proyecto en sí mismo.

  A su vez, cabe destacar que el conjunto de datos supone una **limitación importante** al alcance del proyecto, dado que la existencia o ausencia de datos de calidad de las diversas etiquetas que se desea clasificar determina la amplitud que abarca el proyecto.

- **Diseño e implementación**: comprende parte del trabajo englobado en la PEC 3 (Diseño e implementación). En esta fase se realiza la exploración, transformación, limpieza y e integración del conjunto de datos seleccionado; se estudian las técnicas de modelado y se escogen las más prometedoras de acuerdo a las métricas de evaluación previamente diseñadas; se construyen y evalúan los modelos correspondientes y se documenta el trabajo realizado.

- **Memoria, preparación y defensa**: comprende la parte final de la asignatura y se refiere al trabajo de documentación y redacción de esta memoria, la preparación y la defensa del proyecto ante el jurado.

1.6. Breve sumario de productos obtenidos

Como resultado del trabajo, desde la fase de definición del marco del proyecto hasta la conclusión del trabajo, pasando por su desarrollo, se obtienen los siguientes productos.

- Un ***framework* y API de entrenamiento para modelos de detección de residuos urbanos**.
- Estado del arte de los artículos científicos relacionados con los sistemas de detección y clasificación de residuos basados en visión por computador y específicamente los que utilizan técnicas de *Deep Learning*.
- Una revisión exhaustiva de los conjuntos de datos más populares en las tareas de clasificación y detección de residuos urbanos o *Municipal Solid Waste*.
- Una comparativa de los modelos *Single Shot Detector* (Wei Liu y col. 2015) y *Faster* R-CNN (Shaoqing Ren y col. 2015) de detección de residuos en las categorías de papel (cartón incluido) y plástico.

1.7. Breve descripción de los capítulos de la memoria

En este primer capítulo de introducción, se ofrece el contexto y justificación del problema; se definen los objetivos y metodología a seguir; se describe el impacto en materia de sostenibilidad, comportamiento ético y diversidad; y se ofrece un cronograma de la planificación del proyecto.

En el capítulo 2 se establece el estado del arte (*State of the Art* o *SotA*) y se describe el marco en el que se realiza el proyecto desde una visión más general (los avances más recientes en *Deep Learning* aplicado a visión por computador) hasta los últimos proyectos que se han enfrentado a

problemáticas similares a las propuestas en este documento.

El análisis y selección de los conjuntos de datos se encuentra en el capítulo 3, en el que se estudia la idoneidad de cada uno de los *datasets* candidatos y se eligen aquellos sobre los que se trabajará; la selección del conjunto de datos final determinará el alcance del proyecto, centrándolo en la segregación de papel y plástico.

En el capítulo 4 se explica en profundidad los modelos seleccionados, las métricas de evaluación aplicadas y el proceso de entrenamiento, así como una recopilación de problemas encontrados y las decisiones y soluciones que se han aplicado para solventarlos o circundarlos.

Los resultados de los experimentos propuestos quedan recogidos en el capítulo 5, que analiza la bondad del ajuste de los modelos a través de las métricas planteadas en el capítulo anterior, presenta la huella de carbono del entrenamiento y el tiempo que ha requerido dicho entrenamiento.

Por último, en el capítulo 6 se extraen las conclusiones del proyecto y se proponen nuevas líneas de investigación para continuar el trabajo. En este capítulo también se realiza un análisis crítico de los impactos previstos, el seguimiento del calendario propuesto y la consecución de los objetivos planteados inicialmente.

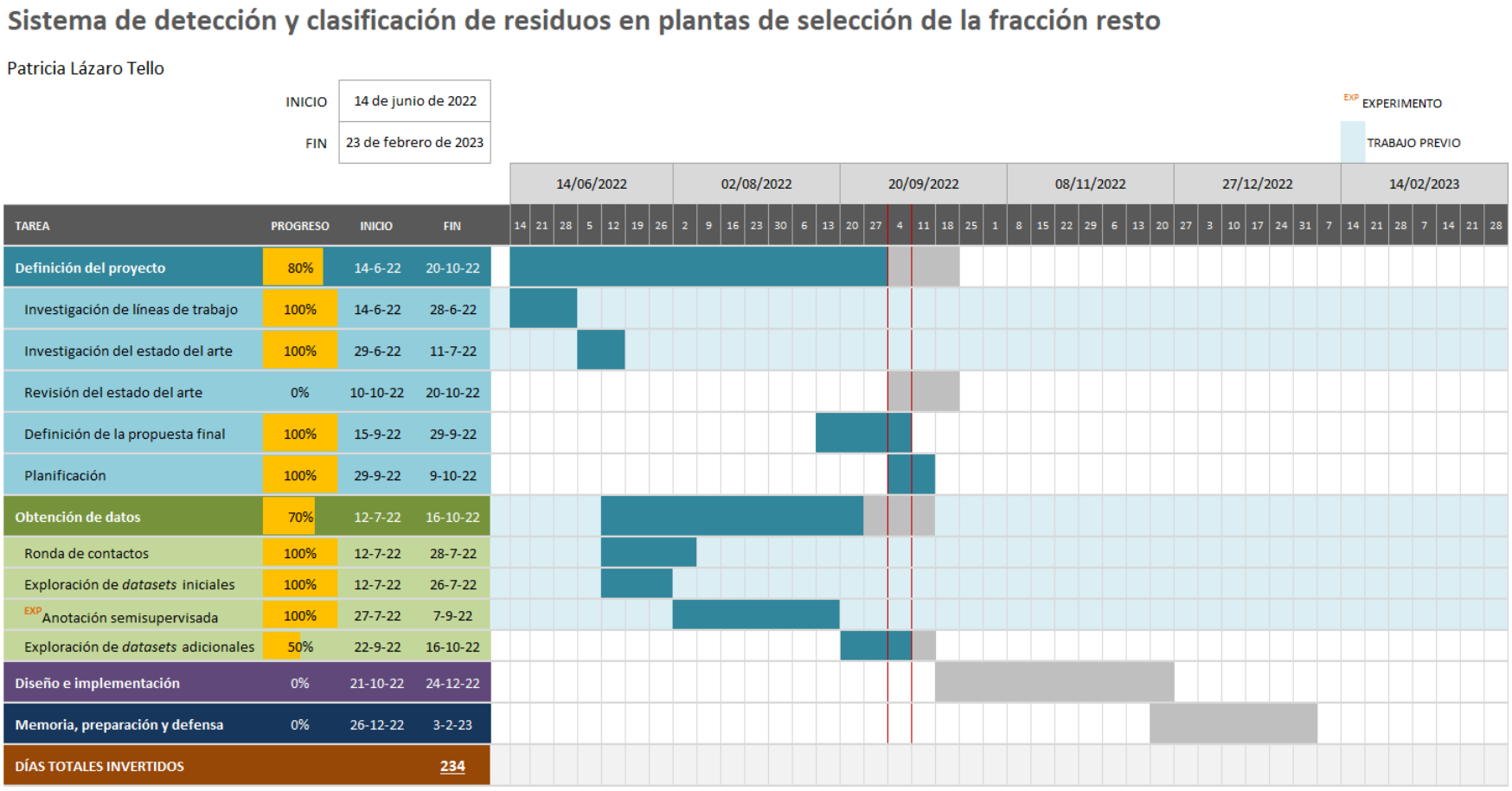


Figura 1.6: Planificación desglosada del proyecto. Incluye el trabajo previo realizado.

Capítulo 2

Estado del Arte

2.1. Visión por Computador

La **Visión por Computador** (*Computer Vision* o CV) es un campo de la inteligencia artificial (AI) en el que se engloban técnicas, métodos y procesos para comprender la información capturada en una imagen o vídeo y traducirla a datos que puedan ser tratados por un ordenador (Richmond Alake 2020). Dentro del campo de la CV, existen dos vertientes o conjuntos de técnicas principales (Niall O' Mahony y col. 2019):

- Visión por Computador Tradicional: la CV tradicional es un conjunto de técnicas y algoritmos que nace en los años 70 y se basa en la extracción de características o *features* en imágenes para obtener información adicional.

  Estas *features* han de ser definidas de antemano por personas, lo que supone un esfuerzo e inversión de tiempo considerable. Además, dado que la definición de las *features* es asumida por el ser humano, es posible que no se describan todas las características relevantes para el problema o queden características ocultas.

  Los algoritmos más conocidos de esta vertiente son el algoritmo de detección de objetos SIFT (*Scale Invariant Feature Transform*) o los algoritmos de detección de bordes como los filtros de Sobel o Canny.

- Técnicas de *Deep Learning* (DL): las técnicas de DL son un subconjunto de técnicas de aprendizaje automático o *machine learning* (ML) basadas en redes neuronales que busca resolver los problemas propuestos de una forma similar a la que los resolvería el cerebro humano, mediante la conexión y agrupación de neuronas que realizan operaciones sencillas para conseguir un resultado final complejo.

  Estas técnicas han visto un crecimiento exponencial en sus aplicaciones en los últimos años gracias a los avances del *hardware* y más concretamente de las GPUs, que han permitido el desarrollo de redes más profundas que obtienen mejores resultados.

La diferencia principal (Richmond Alake 2020 y Ilija Mihajlovic 2019) entre los dos enfoques se en-

cuentra en la **definición de las *features***: en CV tradicional, se realiza un análisis exhaustivo de las características a extraer para modelar el dominio del problema, mientras que en DL las tareas de definir y extraer características (también llamadas *feature engineering*) se delegan en la red neuronal, que será la encargada de encontrar los patrones y características en los datos.

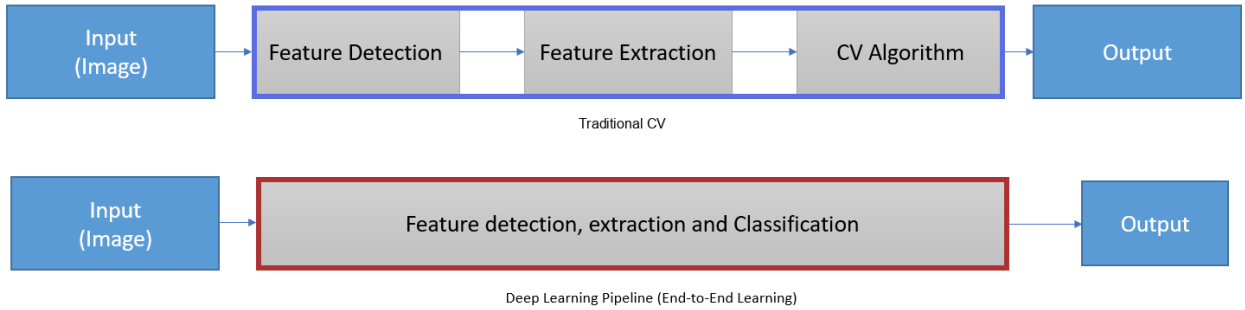


Figura 2.1: El proceso y el responsable de la definición de las *features* es la diferencia más notable entre CV tradicional y DL
Fuente: Niall O' Mahony y col. 2019

Así, el encargado de definir, detectar, extraer y aplicar el algoritmo de CV sobre las imágenes deja de ser la persona para delegarlo en el computador, en la llamada *pipeline* de *Deep Learning* o *end-to-end learning* (aprendizaje extremo a extremo o aprendizaje completo).

La aplicación de técnicas de *Deep Learning*, tanto en visión por computador como en otras áreas (lenguaje natural) se ha popularizado (Zhong-Qiu Zhao y col. 2019) con los avances tecnológicos y de diseño de estructuras neuronales y estrategias de entrenamiento, y la llegada de datos masivos o *big data*, pudiendo sobrepasar las mayores limitaciones que DL tenía hasta este momento: los problemas de *overfitting* o sobreentrenamiento, la escasez de datos para el entrenamiento de la red neuronal y la falta de poder computacional para llevarlo a cabo.

En este trabajo se ha decidido **utilizar y comparar varias técnicas de *Deep Learning* aplicado a Visión por Computador** para obtener un modelo capaz de detectar y reconocer papel y plástico entre los residuos. Esta decisión surge de la necesidad de economizar el tiempo de trabajo y minimizar el número de características ocultas que pudiera tener la alternativa de emplear *feature engineering*, así como de la conveniencia de explorar la frontera del conocimiento en este área de Ciencia de Datos.

En el ámbito del presente proyecto, el problema a resolver es la detección de múltiples objetos de varias clases en una imagen en tiempo real. Habiendo definido el problema, se delimitan las áreas a explorar a **detección de objetos** y **segmentación de instancias**, siempre considerando que el marco de aplicación final es la detección en tiempo real mediante *hardware* no especializado.

2.2. Técnicas de detección y segmentación de objetos

En la literatura (Li Liu y col. 2018) tanto detección como segmentación de objetos se engloban arbitrariamente bajo la categoría de detección de objetos, pese a que la segmentación propone máscaras precisas que se adaptan a la geometría del objeto en vez de las *bounding boxes* más inexactas que

propone la detección.

Esta distinción supone una localización, área y geometría más exactas para el sistema robótico de segregación de residuos; no obstante, también supone un mayor coste computacional que puede ser inasumible en una tarea que debe resolverse en tiempo real. Por este motivo se prestará particular atención al rendimiento de los algoritmos, con especial énfasis en el caso de los de segmentación de instancias.

El **proceso básico de detección de objetos** (Wang Hechun y Zheng Xiaohong 2019) se distribuye en tres pasos:

1. **Proposición de regiones** (*regional proposal*): mediante el uso de ventanas deslizantes (*sliding windows*) de distintos tamaños u otras estrategias, se proponen regiones de la imagen que pueden contener una instancia de los objetos a clasificar.
2. **Extracción de características** (*feature extraction*): se procesa cada región mediante redes neuronales convolucionales (CNNs) para extraer un mapa de características.
3. **Clasificación de regiones** (*regional classification*): se clasifica cada región atendiendo al mapa obtenido en la extracción de características.

Actualmente, existen dos enfoques o vertientes (Zhong-Qiu Zhao y col. 2019) a la resolución del problema, que han dado lugar a dos metodologías de detección de objetos: los **detectores de dos pasadas** (*two stage detector*) y los **detectores de una pasada** (*one stage detector*). En la figura 2.2 se observa la evolución a lo largo del tiempo de las arquitecturas más populares de cada enfoque.

Independientemente del enfoque del algoritmo, los detectores utilizan determinadas técnicas como ***Non-Maximal Suppression*** (NMS) (Jan Hosang, Rodrigo Benenson y Bernt Schiele 2017) e ***Intersection over Union*** (IoU) (Jiahui Yu y col. 2016) con el objetivo de descartar y combinar las *bounding boxes* que se superponen por encima de un umbral.

Las métricas más comunes para evaluar los detectores (Li Liu y col. 2018) son ***mean Average Precision*** (mAP) y *mean Average Recall* (mAR), relegando a un segundo plano otras métricas como *precision* y *recall*.

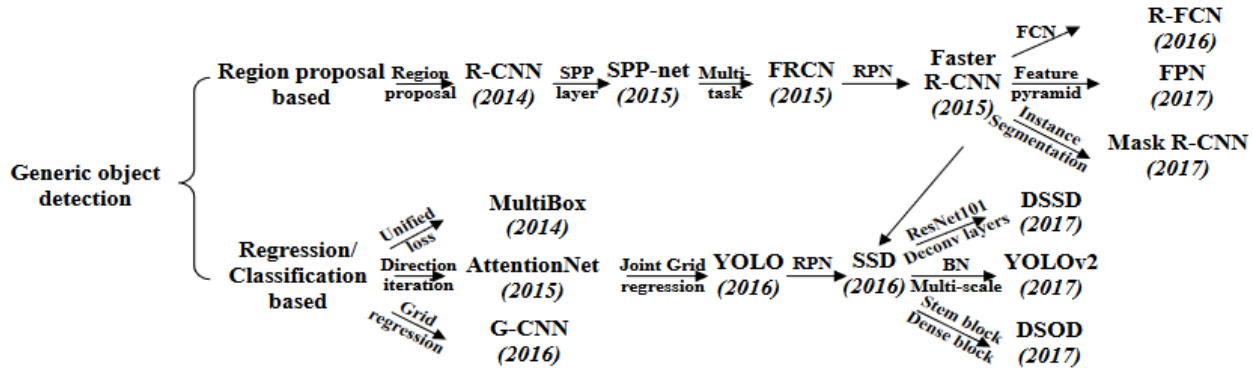


Figura 2.2: Evolución de los modelos de DL de detección y segmentación
Fuente: Zhong-Qiu Zhao y col. 2019

2.2.1. *Two stage detectors*

Los detectores de dos pasadas consideran el problema de detección de objetos en dos partes: proposición de regiones y su clasificación posterior. Por tanto, el detector consta de dos modelos (un modelo propone regiones mediante búsqueda exhaustiva y otro modelo clasifica cada región) que son entrenados alternativamente.

Los algoritmos que siguen este enfoque destacan por su gran precisión, pudiendo implementar segmentación por instancias mediante máscaras, aunque padecen de problemas de rendimiento que los hace inviables para aplicaciones en tiempo real.

Los detectores de dos pasadas más populares corresponden con la familia R-CNN (Ross Girshick y col. 2013) (*Faster* R-CNN (Shaoqing Ren y col. 2015), *Mask* R-CNN(Kaiming He, Georgia Gkioxari y col. 2017)), aunque existen otros como SPP-Net (Kaiming He, Xiangyu Zhang y col. 2014) y FPN (Tsung-Yi Lin, Piotr Dollár y col. 2016).

2.2.2. *One stage detectors*

Los detectores de una pasada consideran el problema de detección de objetos como una única tarea de regresión o clasificación generando un solo modelo. Se utilizan funciones de *loss* más complejas que las descritas en la subsección 2.2.1, debido a que el entrenamiento se realiza en una pasada y por tanto la función de *loss* tiene que combinar las pérdidas de la proposición de regiones y las de la clasificación de las regiones propuestas.

El objetivo fundamental de los detectores de una pasada es cumplir los requerimientos de tiempo de respuesta de las aplicaciones en tiempo real siendo más eficientes computacionalmente. No obstante, se ha de considerar el balance entre la velocidad de inferencia del modelo y la disminución de precisión que le acompaña.

Los detectores de una pasada basados en *bounding boxes* más populares incluyen la familia YOLO (Joseph Redmon y col. 2015), SSD (Wei Liu y col. 2015), FCOS (Zhi Tian y col. 2019) y RetinaNet (Tsung-Yi Lin, Priya Goyal y col. 2017), entre otros.

Además, en los últimos años los detectores de una pasada están evolucionando hacia la segmentación de instancias, con YOLACT (Daniel Bolya y col. 2019), PolarMask (Enze Xie y col. 2019) o SOLO (Xinlong Wang y col. 2019) como ejemplos prominentes de segmentadores de instancias de una pasada.

2.3. Evitando el sobreentrenamiento: *Transfer learning*

Las redes neuronales y modelos propuestos para detección de objetos siguiendo cualquiera de los enfoques definidos cuentan con múltiples capas ocultas con una gran cantidad de neuronas y parámetros en cada capa, lo que puede derivar en *overfitting* o sobreentrenamiento si el *corpus* de imágenes de entrenamiento no es suficientemente grande. Para reducir este *overfitting* se utilizan técnicas de *transfer learning* y *fine-tuning* (Victor Roman 2020).

En la literatura de *transfer learning* (Sinno Jialin Pan y Qiang Yang 2010), se entiende por dominio

el espacio de características y su distribución de probabilidad. Dicho de otra manera, el dominio en este trabajo corresponde al *corpus* de imágenes.

Asimismo, la tarea es el espacio de clases y la función predictiva que transforma una instancia del dominio en una clase. Para los efectos de este proyecto, la tarea corresponde con la clasificación en papel y plástico.

Definidos estos conceptos, la técnica de ***transfer learning*** consiste en utilizar un conjunto de datos fuente de dominio y/o tarea distinta para pre-entrenar el modelo. Seguidamente, se realizará un segundo entrenamiento con el conjunto de datos destino, congelando las capas de la red del extractor de características.

Con este entrenamiento previo, se consigue que la red aprenda un conocimiento que pueda extrapolar al conjunto de datos destino; este conocimiento puede ser desde el aprendizaje de tareas genéricas como la detección de bordes en la imagen hasta un aprendizaje más específico del dominio destino.

El segundo concepto, ***fine-tuning***, es una especialización de *transfer learning* que se aplica sobre el modelo pre-entrenado en los conjuntos de datos fuente y destino. Se basa en descongelar algunas capas del extractor de características y el clasificador, y volver a entrenar la red con el conjunto de datos destino con un *learning rate* muy inferior al utilizado en pasos anteriores.

El objetivo del *fine-tuning* es realizar pequeñas modificaciones sobre los pesos de la red para mejorar ligeramente su rendimiento, es decir, especializa el modelo poco a poco para la tarea y dominio del conjunto de datos destino.

2.4. *Deep Learning* en CV para clasificación de residuos

Respecto al dominio de los residuos urbanos, su clasificación y segregación para su posterior reaprovechamiento, destaca la variedad de áreas de aplicación (Weisheng Lu y Junjie Chen 2022) de acuerdo a diferentes criterios de segmentación.

En primer lugar, analizando las fuentes de residuos se pueden establecer tres categorías: **residuos urbanos**, residuos industriales y comerciales, y residuos de construcción y demolición. El objeto de estudio de este proyecto son los residuos urbanos, la categoría más popular (Weisheng Lu y Junjie Chen 2022) según la fuente de residuos y la que más conjuntos de datos públicos tiene dispuestos.

En segundo lugar, si se segmenta por objetivos se obtienen las tareas de **detección de residuos** y reconocimiento de residuos. La detección de residuos hace referencia a la localización y clasificación de los residuos en la imagen, mientras que en el reconocimiento sólo se realiza la clasificación. Respecto a los conjuntos de datos según el objetivo, cabe destacar que existe un número inferior de *datasets* públicos aptos para la detección

Por último, de acuerdo al momento temporal en que se realiza la clasificación y segregación de los residuos se obtienen dos líneas de trabajo: la detección o reconocimiento en el origen y la realizada en una **planta de tratamiento de residuos**. La mayoría de los conjuntos de datos públicos disponibles

pertenecen a la detección o reconocimiento en origen, encontrándose pocos conjuntos de datos de plantas de tratamientos de residuos que no sean privados.

De todo lo anterior se desprende que el dominio de aplicación dentro del ámbito del reciclaje en que se enmarca el proyecto es el de detección de residuos urbanos en plantas de tratamiento.

Enfoque	Modelo	Origen de datos		
		RM	ICI	C&M
ML tradicional	*Linear Discriminant Analysis (LDA)* [1]	1	-	-
	Nearest Neighbor (NN) [2]	**6**	-	-
	Artificial Neural Network (ANN) [3]	**3**	-	-
	Support Vector Machine (SVM) [4]	**3**	-	-
	Clasificador basado en reglas [5]	-	2	-
DL	R-CNN [6]	-	-	1
	Fast R-CNN [7]	1	-	-
	***Faster* R-CNN** [8]	**6**	-	1
	***Mask* R-CNN** [9]	**3**	-	1
	RetinaNet [10]	1	-	-
	YOLO [11]	2	-	-
	YOLACT [12]	-	-	1
	SSD [13]	**3**	-	1
ML+DL	-	-	-	-

Cuadro 2.1: Estudios dedicados a la detección de residuos según el enfoque, modelo y origen de datos definidos. Fuentes:

[13]Nobuyoshi Yabuki, Naoto Nishimura y Tomohiro Fukuda 2018
[8,13]Daniel Octavian Melinte, Ana-Maria Travediu y Dan N. Dumitriu 2020
[3]Takuya Kiyokawa y col. 2021
[9]Spencer Ploeger, Matthew Bolan y Lucas Dasovic 2022
[1,2,3,4,5,6,7,8,9,10]Weisheng Lu y Junjie Chen 2022
[13]Zhifei Zhang y col. 2019
[8,9,11]Jiewen Feng y col. 2021
[9,11]Wei-Lung Mao y col. 2022
[12]Seunguk Na y col. 2022
[11]Remi Cuingnet y col. 2022

Respecto a los algoritmos y modelos utilizados, cabe destacar que existen tres líneas de trabajo: la línea de algoritmos tradicionales de ML, los modelos *end-to-end* de DL y los modelos híbridos que combinan DL y algoritmos tradicionales de ML. En la tabla 2.1 se puede observar el volumen de estudios dedicados a la detección de residuos, diferenciando por línea de trabajo y modelo aplicado, así como por el origen de datos (ver Glosario para más detalles de los acrónimos utilizados). Esta tabla no busca ser exhaustiva.

De la visión general del estado del arte en el dominio en que se centra este proyecto se pone de manifiesto la atención que atrae la detección de residuos urbanos, así como las técnicas y modelos más utilizados en cada línea de trabajo. También es importante destacar que no existe ningún trabajo que estudie modelos híbridos en materia de detección de residuos dentro de la literatura examinada.

Dentro de los modelos de *machine learning* tradicional, los más populares para la detección de resi-

duos urbanos son los algoritmos de *Nearest Neighbor*, *Support Vector Machine* y *Artificial Neuronal Network*. La mayoría de estudio de esta rama se centran en los residuos urbanos, sin entrar en la detección de residuos de construcción y demolición.

En los modelos de *Deep Learning* destaca la popularidad de los modelos centrados en detección de residuos de construcción y demolición respecto a las otras ramas. El número de estudios dedicados a la detección de residuos urbanos es ligeramente superior que en los modelos de *machine learning* tradicional, y están basados principalmente en modelos de proposición de regiones (RPN), también llamados detectores de dos fases, con especial énfasis en la familia R-CNN.

A la hora de evaluar el rendimiento de los modelos, se utilizan una variedad de métricas como la *mean Average Precision* (mAP), la *accuracy* o la *precision*, por lo que no siempre es posible comparar dos modelos de forma directa.

Para los casos en que se proporcionen varias métricas, se elegirá la mAP como punto de comparación; si no se proporciona pero se puede obtener de los datos, se calculará. Adicionalmente, se define el umbral de aplicación en tiempo real en 20FPS (*frames per second*).

2.4.1. *Two stage detectors*: *Faster* R-CNN y *Mask* R-CNN

Los detectores de dos pasadas se caracterizan por realizar detecciones más precisas que los detectores de una pasada. Dentro de los modelos de proposición de regiones más populares para la detección de residuos urbanos destacan ***Faster* R-CNN**, que resuelve la tarea de detección de objetos, y **Mask R-CNN**, que resuelve la tarea de segmentación de instancias.

Estudio	Modelo	mAP	$T_{procesamiento}$*
Oluwasanya Awe y Robel Mengistu 2017	*Faster* R-CNN	0.683	-
Wei-Lung Mao y col. 2022	*Faster* R-CNN	0.937	180ms
Piotr T. Nowakowski y Teresa Pamuła 2020	*Faster* R-CNN	**0.93**	-
Jiewen Feng y col. 2021	*Faster* R-CNN	0.78	570ms
	Mask R-CNN	**0.93**	109ms
Remi Cuingnet y col. 2022	*Faster* R-CNN	0.87	137ms
Daniel Octavian Melinte, Ana-Maria Travediu y Dan N. Dumitriu 2020	*Faster* R-CNN	0.816	90ms
Spencer Ploeger, Matthew Bolan y Lucas Dasovic 2022	*Mask* R-CNN	0.56	-
Pedro F. Proença y Pedro Simões 2020	*Mask* R-CNN	0.194	-

*Tiempo de procesamiento de una imagen en inferencia. Las características de la imagen.
*Los tiempos son orientativos del orden de magnitud en ms.
*Las características de la imagen y del *hardware* utilizado en cada estudio difieren.

Cuadro 2.2: Resultados de detectores de dos pasadas para la clasificación de residuos urbanos

Las métricas obtenidas en los estudios previos (ver tabla 2.2) muestran que los modelos de *Mask* R-CNN obtienen resultados más dispares que los modelos de *Faster* R-CNN. En general, no obstante, se obtienen valores superiores al 90 % de mAP.

Respecto a los tiempos de procesamiento, en los estudios que publican sus tiempos de procesamiento de una imagen en *inference mode* (el modelo entrenado realizando detecciones en modo de *test*) se observa que no se obtiene un rendimiento digno de una aplicación en tiempo real: el modelo más rápido funciona a 4FPS, frente al ideal de 20FPS.

2.4.2. *One stage detectors*: SSD y YOLO

Los detectores de una pasada se caracterizan por realizar detecciones más rápidas, aunque potencialmente menos precisas. Dentro de esta categoría de técnicas de detección de objetos destacan **SSD** y **YOLO**, los detectores más populares de una pasada. Ambos resuelven tareas de detección de objetos, que es computacionalmente menos intensiva que la segmentación de instancias.

Estudio	Modelo	mAP	$T_{procesamiento}$*
Daniel Octavian Melinte, Ana-Maria Travediu y Dan N. Dumitriu 2020	SSD	**0.973**	40ms
Zhifei Zhang y col. 2019	SSD	-	-
Takuya Kiyokawa y col. 2021	SSD	0.79	-
Jiewen Feng y col. 2021	YOLOv3	0.84	42.83ms
Wei-Lung Mao y col. 2022	YOLOv3	**0.921**	4.5ms**

*Tiempo de procesamiento de una imagen en inferencia. Las características de la imagen varían.
*Los tiempos son orientativos del orden de magnitud en ms.
*Las características de la imagen y del *hardware* utilizado en cada estudio difieren.
**Resultados obtenidos en una GPU dedicada (NVIDIA Geforce RTX 2080 Super)

Cuadro 2.3: Resultados de detectores de una pasada para la clasificación de residuos urbanos

Las métricas obtenidas en los estudios previos (ver tabla 2.3) muestran resultados más que adecuados en las mAP, con casi un 20 % de variación entre estudios del mismo modelo. Respecto al tiempo de procesamiento, los estudios que lo publican obtienen resultados muy cercanos al tiempo real, a 9FPS frente a los 20FPS ideal, con *hardware* genérico.

2.4.3. Modelos de *Deep Learning* investigados en este proyecto

De las secciones 2.4.1 y 2.4.2 se desprende la idoneidad de ciertos modelos en la tarea de detección de residuos domésticos. Los detectores basados en proposición de regiones ofrecen un tiempo de procesamiento un orden de magnitud superior al de los detectores de una pasada, que no padecen de una pérdida de precisión significativa.

Además, cabe destacar que en la literatura revisada no existen estudios sobre la detección de residuos que integren los algoritmos tradicionales de ML con las redes neuronales de *Deep Learning*; por ello, se ha decidido incorporar a este estudio un modelo híbrido para su comparación con los modelos de *Deep Learning*.

Como conclusión de las técnicas y modelos de *Deep Learning* aplicado a la detección de residuos domésticos en plantas de tratamiento, se determina estudiar, evaluar y comparar modelos de *Deep Learning* de una y dos pasadas, así como modelos híbridos donde una red neuronal actúa de extractor

de características y un algoritmo de ML tradicional realiza la tarea de clasificación. Se compararán los modelos elegidos tanto en las métricas de evaluación como en el tiempo de inferencia.

Respecto a los modelos de DL de dos pasadas, se escoge ***Faster* R-CNN** debido al compromiso de rendimiento y velocidad, así como la idoneidad del formato de entrada de parámetros (*bounding boxes* frente a máscaras).

En cuanto a los modelos de DL de una pasada, se determina utilizar el modelo **SSD** por ser de los más populares dentro de la detección de residuos de una sola pasada, habiéndose demostrado su rendimiento y velocidad en la sección 2.4.2.

En último lugar, se crearán modelos híbridos utilizando los modelos de *Faster* R-CNN y SSD previamente entrenados como extractores de características y **SVM**, ***Nearest Neighbor*** como clasificadores. Esta decisión parte de la popularidad y el buen rendimiento de estos dos algoritmos en la literatura revisada (Weisheng Lu y Junjie Chen 2022).

2.5. *Datasets* disponibles para la clasificación de residuos

Los principales inconvenientes de las técnicas de *Deep Learning* son la necesidad de un gran volumen de datos y de poder computacional para el entrenamiento de los modelos. En esta sección se aborda el primero de los citados inconvenientes.

El conjunto de datos de entrenamiento determina el dominio de aplicación del modelo, tanto en que define la tarea o problema al que se enfrenta el proyecto como en que establece el espacio de clases al que se enfrenta dicho modelo.

En este sentido se considera que un conjunto de datos está descrito por la tarea y el espacio de clases. En el dominio de clasificación de residuos, se añaden a la definición el origen de los residuos y la complejidad del *background* de la imagen.

En la literatura examinada se encuentra que la mayoría de conjuntos de datos utilizados en los estudios se tratan de corpus privados dedicados a la tarea de detección y/o segmentación de instancias en dominios especializados (por ejemplo, segregación de papel y plástico).

Si bien los conjuntos de datos dedicados a la tarea de clasificación tienen una mayor variedad de clases, los referentes a la tarea de detección ven su espacio de clases reducido a un dominio muy especializado o a un dominio simplificado. A continuación se expone una lista no exhaustiva de *datasets* disponibles, así como una breve descripción de sus características:

1. **TACO** (Pedro F. Proença y Pedro Simões 2020): corpus *open source* y colaborativo de imágenes de basura en el momento de generación del residuo. Consta de 28 categorías y 60 subclases con una distribución desbalanceada.

2. **OpenLitterMap** (Seán Lynch 2017): conjunto *open source* y colaborativo de imágenes de basura en el momento de generación del residuo. No obstante, las imágenes no se encuentran disponibles para su descarga y uso.

3. **MJU-Waste** (Tao Wang y col. 2020): *dataset* de imágenes de basura en contexto simple, clasificadas como residuos o no residuos.

4. **UAVVaste** (Marek Kraft y col. 2021): corpus de 772 imágenes de basura en vista aérea en exteriores, clasificadas como residuos o no residuos.

5. **Drinking Waste Classification** (Arkadiy Serezhkin 2021): imágenes en contexto simple de latas de aluminio, botellas de cristal, botellas PET (envases de plástico transparente y ligero) y botellas HDPE (envases de plástico muy opaco de alta resistencia).

6. **ResortIT** (Maria Koskinopoulou y col. 2021): conjunto de 16.000 imágenes sintéticas que simulan ser tomadas encima de una cinta en planta de tratamiento. Su espacio de clases corresponde con cartón, aluminio, botellas PET y nylon.

7. **ZeroWaste** (Dina Bashkirova y col. 2021): corresponde con 10.715 *frames* de vídeo tomados en una planta de tratamiento de papel y plástico, con 6.212 de ellos sin anotar para realizar aprendizaje débilmente supervisado. Consta de 4 clases: cartón, plástico blando, plástico duro y metal; considera el papel como *background*. Adicionalmente incorpora instancias de TACO y ResortIT para aumentar el corpus.

Del anterior listado de conjuntos de datos se desprende el interés de la comunidad científica por la detección de plásticos y, en menor medida, papel y cartón. La mayoría de los conjuntos de datos se han tomado sobre un contexto simple o sobre el momento de generación del residuo. Además, no existen conjuntos de datos anotados para la detección de residuos orgánicos.

Dadas las dificultades para obtener un conjunto de datos en el contexto de una planta de tratamiento que posea un espacio de clases amplio (residuos orgánicos, papel y cartón, envases de plástico y aluminio, vidrio, y residuos no reciclables) se ha decidido plantear el proyecto como un trabajo de detección para la segregación de papel y plástico.

En cualquier caso, se plantea la posibilidad de ampliar el conjunto de datos seleccionado (**ZeroWaste**) con corpus adicionales o llevar a cabo aprendizaje semisupervisado de otros *datasets* manualmente anotados para ampliar el espacio de clases, así como realizar un primer entrenamiento de los modelos con un conjunto de datos sintético auxiliar (**ResortIT**).

Los resultados de la exploración de todas estas posibilidades y las conclusiones que de ello se desprenden quedan recogidas en el capítulo Selección del conjunto de datos.

Capítulo 3

Selección del conjunto de datos

Herramientas utilizadas

Para el Análisis Exploratorio de los Datos o EDA (*Exploratory Data Analysis*), se han utilizado las siguientes herramientas:

- Jupyter Notebook
- Python 3.10
- Librerías de Python: pandas, numpy
- Librerías de Python: maptlotlib, seaborn

3.1. Obtención de conjuntos de datos completos

En el primer análisis realizado en el capítulo 2: Estado del Arte, se aprecia que no existe un conjunto de datos similar al conjunto de datos objetivo de este proyecto –un *dataset* de las seis categorías de reciclaje en que se segregan los residuos en España, tomadas desde encima de la cinta transportadora en la planta de segregación y selección.

Con la intención de obtener un conjunto de datos que represente las categorías generales de reciclaje y el tipo de residuos más comunes en España, se ha contactado con diversas entidades dedicadas al reciclaje para obtener datos de las categorías no reflejadas en los *datasets* ya obtenidos.

Concretamente, se ha contactado con Ecoembes (Ecoembes 2022), Saica (Saica 2022), URBASER (URBASER 2022), FCC Medio Ambiente (FCC Medio Ambiente 2022), Ecovidrio (Ecovidrio 2022) y Tractament i Seleccio de Residus, SA; no obstante, no se ha obtenido respuesta de ninguna de las cinco primeras y aunque se pudo grabar la cinta de reciclaje en la última planta, el vídeo no tiene la calidad suficiente para utilizarse en este proyecto, por lo que se ha decidido explorar nuevos *datasets* para completar las categorías restantes y en última instancia, enfocar el proyecto a la segregación de papel y plástico.

3.2. *Datasets* principal y complementario

3.2.1. `ZeroWaste`

El conjunto de datos principal utilizado para el entrenamiento de los modelos de detección de residuos propuestos es **ZeroWaste** (Dina Bashkirova y col. 2021). Se trata de un conjunto de datos con varias

versiones:

- **`zerowaste-f`**: conjunto de 4.414 imágenes de 1920x1080px para entrenamiento supervisado. Las imágenes se encuentran totalmente anotadas.
- **`zerowaste-s`**: conjunto de 6.212 imágenes de 1920x1080px sin anotaciones para utilizar conjuntamente con `zerowaste-f` en entrenamiento semisupervisados.
- **`zerowaste-w`**: conjunto de 2.410 imágenes de 1920x1080px para entrenamiento débilmente supervisado, donde 1.202 *frames* se encuentran anotados y 1.208 no contienen anotaciones.
- **`zerowaste-aug`**: conjunto de 4.503 *frames* de 1920x1080px para entrenamiento supervisado, con anotaciones en todas las imágenes. Se han utilizado objetos recortados del conjunto de datos TACO (Pedro F. Proença y Pedro Simões 2020) para balancear las clases.

Este estudio se centra en la detección de residuos de tipo **papel** y **plástico**, sin diferenciar entre papel y cartón o entre plástico rígido y blando. Los *datasets* **ZeroWaste** realizan una clasificación en 4 clases, que quedan asociadas con las clases utilizadas en el trabajo de la siguiente manera:

- Cartón → Papel
- Plástico rígido o duro → Plástico
- Plástico blando → Plástico
- Metal → X

Debido a que se va a realizar un pre-entrenamiento con **ResortIT** (Maria Koskinopoulou y col. 2021), se descarta la versión `zerowaste-aug`. Se descarta también la versión débilmente supervisada `zerowaste-w`, ya que contiene menos anotaciones que la versión versión totalmente supervisada `zerowaste-f`.

Finalmente, como el estudio se realiza sobre modelos con entrenamiento supervisado, se decide seleccionar como conjunto de datos principal la versión **`zerowaste-f`**, aunque no se descarta la extracción de un corpus reducido de `zerowaste-s` para comprobar el poder predictivo de los modelos sobre imágenes no anotadas.

Para cada versión del conjunto de datos, se proporcionan tanto anotaciones en formato COCO (ver el anexo B) como máscaras para segmentación de instancias. Como el objetivo del proyecto es la detección de residuos, se utilizarán las anotaciones en formato COCO, transformándolas en observaciones en un *dataframe*.

Se obtiene un total de 26.766 anotaciones para las 4.414 imágenes, que se agrupan en los conjuntos de entrenamiento (67.25 %), validación (13.77 %) y test (18.97 %). Respecto a las clases, se obtiene un 66.32 % de anotaciones de papel y cartón, un 32.25 % de anotaciones de todo tipo de plástico y un 1.43 % de anotaciones de metal (que serán eliminadas para el conjuntos de datos final).

3.2.2. `ResortIT`

Respecto al conjunto de datos complementario, **ResortIT**, es un *dataset* completamente sintético creado a partir de residuos de plástico rígido y blando, metal y cartón, recortados y superpuestos sobre distintos fondos. Las imágenes tienen 800x800px y sólo contienen particiones de entrenamiento

Figura 3.1: Muestra de datos de **ZeroWaste** (izquierda) y **ResortIT** (derecha).
En azul se muestran las anotaciones de papel y cartón, y en amarillo las de plástico.
Fuente: elaboración propia

(74.07 %) y validación (25.93 %). También cabe destacar que sólo se encuentran anotaciones en formato COCO, sin máscaras para segmentar instancias.

Se obtiene un total de 43.200 anotaciones, de las cuales el 50 % corresponden a papel y cartón, el 50 % corresponde a todo tipo de plásticos, y el 25 % corresponde a metal. La proporción de plásticos es superior debido a que en la asociación de clases del proyecto se incluyen los plásticos rígidos y blandos en la misma categoría.

Si se comparan las dos muestras de datos (ver figura 3.1), se observa que la calidad de los datos en el conjunto de datos ResortIT es significativamente inferior a la de ZeroWaste. Esta es una de las razones que fundamentan descartar la categoría de metales del alcance de este proyecto, ya que los modelos aprenderían principalmente de datos de calidad inferior (10.800 anotaciones de ResortIT contra 382 de ZeroWaste).

En la figura 3.2 se observa la distribución de las clases en los dos conjuntos de datos por separado y en la unión de ambos *datasets*. Se aprecia que aunque el conjunto de datos final se encuentra balanceado (51 % de plástico y 49 % de papel), en **ZeroWaste** se obtiene mayor concentración de anotaciones de papel y en **ResortIT** más anotaciones de plástico.

Esto puede suponer que los modelos entrenados sobre **ResortIT** primero y posteriormente sobre **ZeroWaste** aprendan peores representaciones de la categoría plástico, ya que el *dataset* ResortIT influenciaría más el aprendizaje que el *dataset* ZeroWaste. Se deberá realizar una validación adicional para la categoría plástico al evaluar los modelos.

3.3. *Datasets* auxiliares candidatos

Dado que en el análisis exploratorio inicial se han obtenido exclusivamente muestras de papel y plástico (y algunas muestras testimoniales de metal) y con el objetivo de obtener un corpus de residuos que incluya todas las categorías que forman parte de la **fracción resto** (papel, plástico, metal, vi-

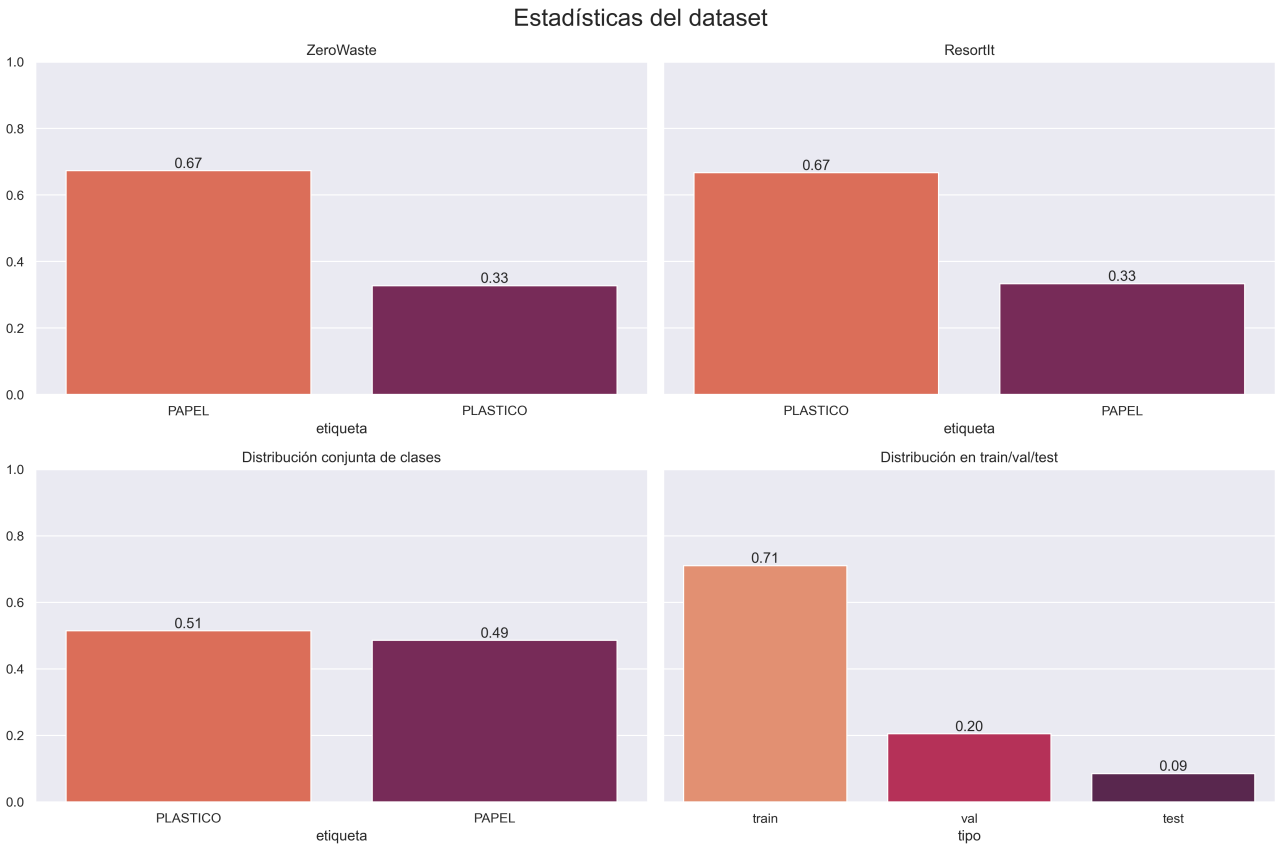


Figura 3.2: Estadísticas de **ZeroWaste** y **ResortIT**
Fuente: elaboración propia

drio, orgánico y otros residuos no reciclables), se han explorado otros *datasets* para complementar el conjunto de datos inicial.

Concretamente, se han explorado el *dataset* `TACO` (Pedro F. Proença y Pedro Simões 2020), `Cig_Butts` (Adam Kelly 2015) y `Drinking Waste` (Arkadiy Serezhkin 2021).

3.3.1. `TACO`

El conjunto de datos `TACO` (o *Trash Annotation in COntext*) es un corpus *open source* de imágenes capturadas con distintos modelos de cámaras en entornos mayormente exteriores, con una o varias anotaciones por imagen, que se guardan en formato COCO.

`TACO` tiene un total de 28 categorías y 60 subclases, con una distribución variada. Después de asociar las clases de `TACO` con las clases que se manejan en el proyecto (las seis categorías del reciclaje), se obtiene una distribución desbalanceada (ver cuadro 3.1).

3.3.2. `Drinking Waste`

El conjunto de datos `Drinking Waste` es un conjunto pequeño de imágenes en contexto plano de botellas de vidrio, tetrapacks, botellas de plástico y latas de aluminio. Las anotaciones se encuen-

tran en formato YOLO y el *dataset* está aumentado artificialmente mediante rotaciones y giros. Sólo contiene anotaciones de plástico y vidrio (ver cuadro 3.1).

3.3.3. `Cig_Butts`

El conjunto de datos `Cig_Butts` es un corpus artificial de imágenes de colillas, con un único objeto por imagen. Las anotaciones están en formato COCO, y todas corresponden con la categoría de residuos no reciclables (otros) (ver cuadro 3.1).

3.3.4. Estadísticas de los conjuntos de datos auxiliares

En el cuadro 3.1 se pueden observar las distribuciones de las observaciones por las categorías del dominio del proyecto. Se aprecia que el conjunto de datos global auxiliar contiene en su mayoría observaciones de plástico y residuos no reciclables (que corresponden principalmente con las colillas de `Cig_Butts`).

La categoría de metal no tienen ninguna observación y la de orgánico sólo cuenta con un 0.06 % (8 observaciones en total). También se aprecia que la categoría de papel apenas tiene observaciones, pero no supone un problema dado que los *datasets* principal y complementario tienen un amplio volumen de observaciones de esa categoría.

Sólo existe un 12.42 % de observaciones de vidrio, que equivale a 1.362 anotaciones, una cantidad no significativa comparada con la muestra de papel y plástico. Además, se obtiene que una gran parte de las observaciones corresponden a la categoría de plástico, que no requiere de más muestras.

Se decide descartar las categorías de metal, vidrio y orgánico por no tener muestras significativas. Se descarta también la categoría de residuos no reciclables debido a que las observaciones obtenidas están fuera de contexto (*Garbage In, Garbage Out*) y a que la muestra tampoco es suficientemente significativa.

Categoría	**% representado**			
	TACO	**Drinking Waste**	**Cig_Butts**	**Total**
Papel	5.89 %	0.0 %	0.0 %	**4.98 %**
Plástico	39.82 %	74.68 %	0.0 %	**54.43 %**
Metal	0.0 %	0.0 %	0.0 %	**0.0 %**
Vidrio	5.18 %	25.31 %	0.0 %	**11.56 %**
Orgánico	0.16 %	0.0 %	0.0 %	**0.06 %**
Otros	49.93 %	0.0 %	100 %	**29.03 %**

Cuadro 3.1: Distribución de las observaciones por categorías

3.4. Conclusiones

Tras examinar cinco conjuntos de datos distintos (**`ZeroWaste`**, **`ResortIT`**, `TACO`, `Cig_Butts` y `Drinking Waste`), se observa un **desbalanceo de clases** importante donde la mayoría de los datos

Figura 3.3: Muestra de datos de **TACO** (izquierda), **Cig_Butts** y **Drinking Waste** (derecha). En amarillo se muestran las anotaciones de plástico, en verde las de vidrio yen gris las de residuos no reciclables.
Fuente: elaboración propia

se encuentran englobados en las categorías de papel y plástico. Además, algunos *datasets* se han tomado fuera de contexto o en un contexto distinto al planteado en el proyecto (cinta transportadora en una planta de selección).

	PAPEL	PLÁSTICO	TOTAL
%	51 %	49 %	100 %
Nº observaciones	30.233	28.551	58.784

Cuadro 3.2: Distribución de las observaciones por clase

Por todo lo anterior, se decide enfocar el objetivo del estudio en la detección y clasificación de papel y plástico, y descartar los conjuntos de datos `TACO`, `Cig_Butts` y `Drinking Waste`. En el desarrollo y entrenamiento de los modelos posteriores se utilizarán únicamente las anotaciones de papel y plástico contenidas en **`ZeroWaste`** y **`ResortIT`** cuya distribución se puede apreciar en el cuadro 3.2 y en la figura 3.2.

Capítulo 4

Entrenamiento de los modelos seleccionados

Herramientas utilizadas

Para el entrenamiento y evaluación de los modelos, se han utilizado las siguientes herramientas:

- Jupyter Notebook
- Python 3.10
- Pytorch Lightning
- Pytorch 1.12.0 y torchvision 0.13.0
- Neptune y Tensorboard como *loggers*
- Librería de Python: Codecarbon
- Librerías de Python: pandas, numpy, scikit-learn
- Librerías de Python: maptlotlib, seaborn

4.1. Métricas de evaluación: *Mean Average Precision*

La métrica ***Mean Average Precision*** (mAP) es la métrica más común (junto con *mean Average Recall* o mAR) para evaluar la bondad de un modelo de detección de objetos. Se fundamenta en los conceptos de matriz de confusión, *Intersection over Union* (IoU), *precision* y *recall*.

La **matriz de confusión** es una herramienta que permite visualizar el rendimiento de un algoritmo de aprendizaje supervisado mediante una tabla de dos dimensiones, en la que se distribuyen las muestras según la clase real de la observación y la clase predicha por el modelo.

En el caso de que se trate de un problema con n clases, se generan tantas matrices de confusión como clases existan y se traduce el problema a n problemas de clasificación binaria. De esta forma, para cada matriz de confusión se obtienen cuatro cuadrantes (figura 4.1):

- ***True Positives*** (TP) o Verdaderos Positivos: el número de muestras verdaderas que el modelo ha predicho como verdaderas.
- ***False Positives*** (FP) o Falsos Positivos: el número de muestras falsas que el modelo ha predicho como verdaderas.
- ***False Negatives*** (FN) o Falsos Negativos: el número de muestras verdaderas que el modelo ha predicho como negativas.

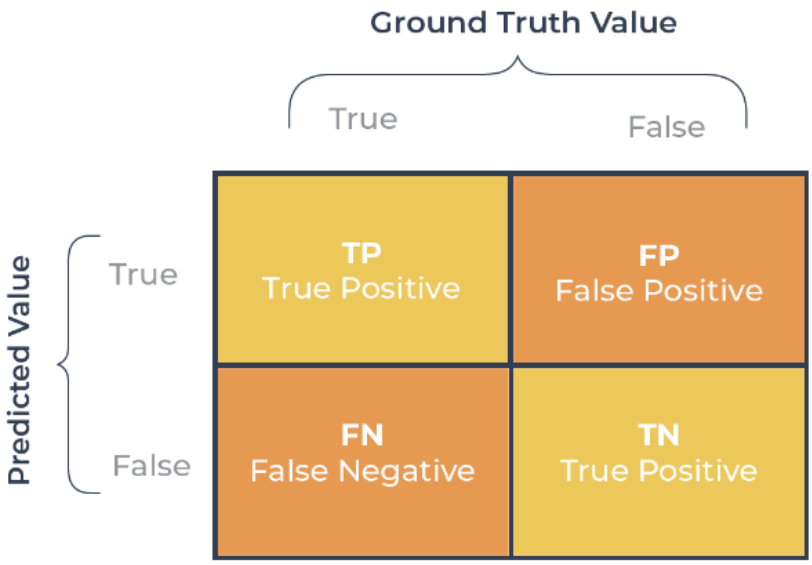


Figura 4.1: Ejemplo de una matriz de confusión y sus cuadrantes.
Fuente: Arize s.f.

- ***True Negatives*** (TN) o Verdaderos Negativos: el número de muestras falsas que el modelo ha predicho como falsas.

De la matriz de confusión se derivan las métricas de **precisión** y ***recall***. La precisión mide cuántos casos predichos como positivos son realmente positivos, mientras que el *recall* o sensibilidad es el ratio de verdaderos positivos entre todos los positivos. Se pueden definir en función de los cuadrantes vistos anteriormente:

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

En el caso de detección de objetos, clasificar una muestra de forma binaria en la matriz de confusión no es tan sencillo: en primer lugar, es necesario emparejar las detecciones (*bounding boxes*) predichas con las detecciones reales. Este emparejamiento se realiza mediante la métrica de ***Intersection over Union*** (IoU), cuya fórmula se encuentra en la figura 4.2.

Dado un *threshold* o umbral de confianza de la métrica de IoU, se calcula para cada pareja *bounding box* predicha y real su índice de IoU, y si dicho índice supera el umbral de confianza dado, se considera que la *bounding box* predicha corresponde con la *bounding box* real.

Una vez obtenidas las correspondencias entre entre predicciones y *bounding boxes* reales, se procede a crear las matrices de confusión para cada clase, la precisión y el *recall*. Además, se calcula también el área bajo la curva de precisión-*recall* para cada clase, comúnmente llamada ***Average Precision***.

La curva de precisión-*recall* se obtiene calculando la precisión y el *recall* de la clase considerando unos *thresholds* para los valores predichos. El umbral define el nivel de confianza necesario para

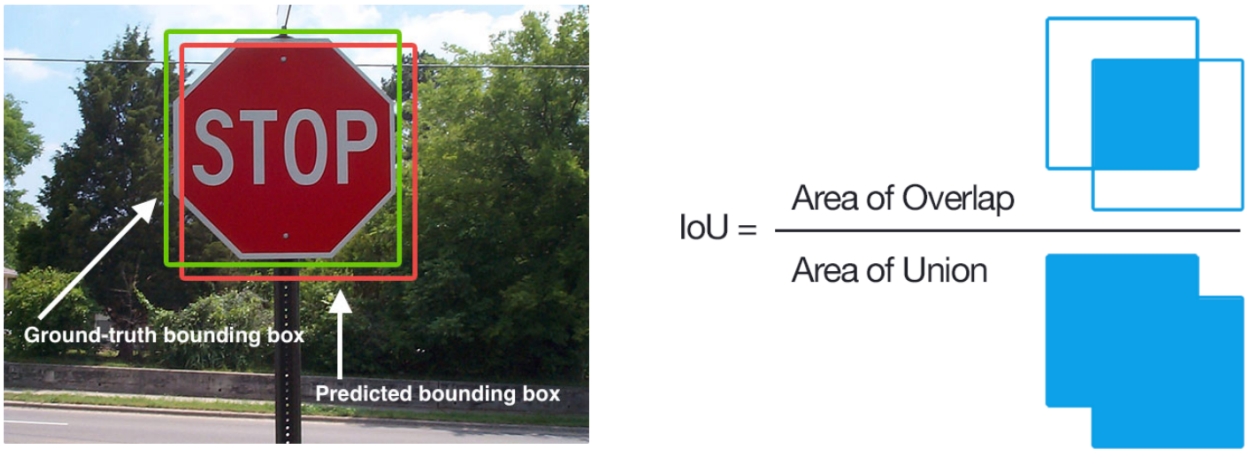


Figura 4.2: *Intersection over Union* (IoU)
Fuente: Adrian Rosebrock 2016

considerar la muestra como perteneciente a la clase. Así, para un umbral bajo la mayoría de las muestras serían de la clase, mientras que para umbrales altos el número disminuiría.

Por último, una vez obtenidas las *Average Precision* de cada clase se realiza la media, obteniéndose así la ***Mean Average Precision***.

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

Esta métrica contiene tanto la precisión como el *recall* por cada clase, por lo que resulta apropiada para calcular la bondad de los modelos de detección de objetos.

4.2. Selección de modelos

Tal y como se concluye en la sección 2.4.3 del capítulo Estado del Arte se selecciona un modelo de detector de una pasada y un modelo de detector de dos pasadas, así como dos algoritmos de clasificación de *Machine Learning* tradicional para explorar un enfoque híbrido en detección de residuos.

Debido a las restricciones temporales y computacionales del trabajo, la selección de los modelos de detección depende en gran medida de las opciones disponibles en el *framework* de desarrollo: **Pytorch**. En el módulo de visión básico (**torchvision**) se encuentran disponibles los siguientes modelos:

- ***Single Shot Detector*** (SSD)
- RetinaNet
- FCOS
- ***Faster* R-CNN**
- *Mask* R-CNN

De los tres modelos de detección de una pasada disponibles, se escoge **SSD** por ser junto a YOLO de los más populares y de los que mejores resultados ha ofrecido de este tipo. Respecto a los detectores

de dos pasadas, se elige ***Faster* R-CNN**, dado que es el más populares, el que mejores resultados ha obtenidos en su tipo y además no se va a llevar a cabo segmentación de instancias.

4.2.1. *Single Shot Detector* o SSD

SSD (Wei Liu y col. 2015) es un modelo de detección de objetos de una sola pasada, es decir, calcula a la vez las *bounding boxes* y las etiquetas o clases (y sus *scores*) asociadas a las mismas.

Se fundamenta en un ***backbone*** entrenado en clasificación de imágenes, en este caso VGG-16, que utiliza como extractor de características o *feature extractor*. El extractor produce múltiples *feature maps* con diferentes escalas para poder detectar objetos más grandes y más pequeños.

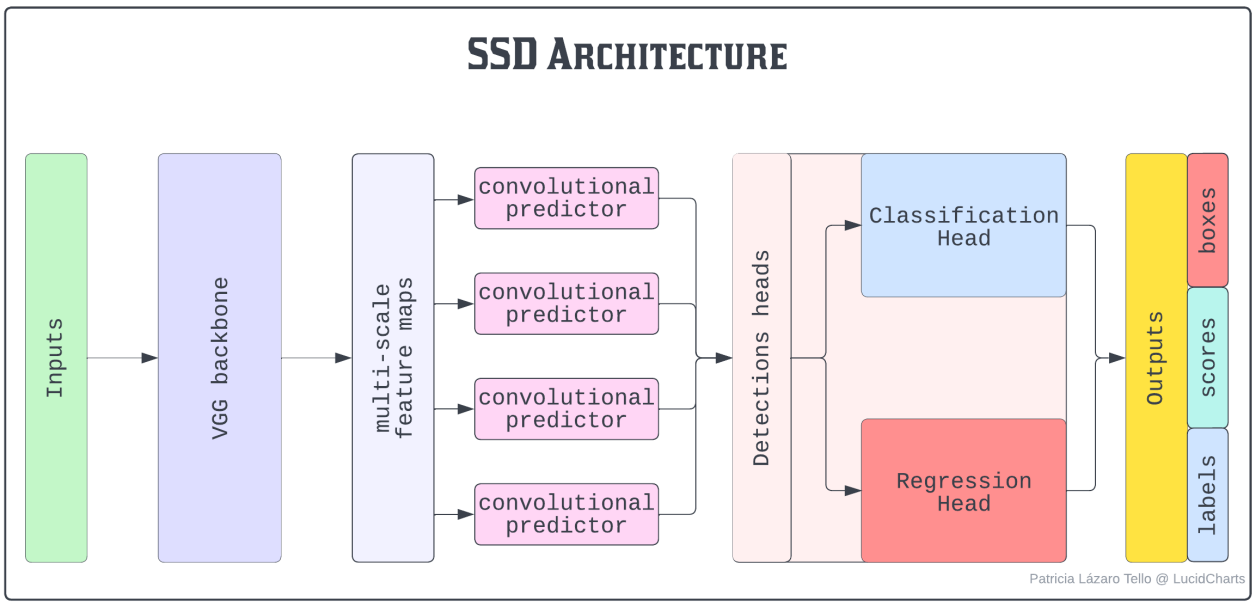


Figura 4.3: Arquitectura del modelo SSD. El entrenamiento de las *bounding boxes* (*regression head*) y de la clasificación (*classification heads*) se realiza simultáneamente.
Fuente: elaboración propia

Para crear las áreas de detección (Sobhan Shukueian 2021), SSD utiliza una cuadrícula para dividir la imagen; cada celda de dicha cuadrícula se encarga de realizar la detección y clasificación de objetos en su zona. Como en cada zona pueden existir múltiples objetos a detectar, se utilizan *anchor boxes* y *aspect ratios* predefinidos.

Por un lado, las *anchor boxes* son *bounding boxes* de posición y tamaño predefinido que se colocan en la celda de la cuadrícula. Como los objetos a detectar pueden variar en forma, a las *anchor boxes* se les añade un *aspect ratio* de los predefinidos para adaptarse mejor a la forma del objeto.

Para detectar a diferentes escalas (objetos pequeños y grandes), la imagen se divide en múltiples cuadrículas con distinto tamaño. Esto produce una gran cantidad de *bounding boxes* que SSD solventa utilizando una estrategia de *matching*.

Para determinar qué *bounding box* predicha corresponde a qué *bounding box* real, se utiliza la métrica

de *Intersection over Union* (IoU), también llamado *Jaccard overlap*. Cuando la métrica es superior al umbral (normalmente 0.5), se considera que corresponde con dicha *bounding box* real.

Por último, para mantener los casos positivos y negativos de *matching*, se establece un ratio 3:1 de casos negativos versus positivos, seleccionando los casos negativos con mayor error.

A su vez, para cada una de las *bounding boxes* predicha, se realiza la clasificación mediante redes neuronales y se obtiene tanto el *score* o probabilidad como la etiqueta de la clase.

4.2.2. *Faster* R-CNN

***Faster* R-CNN** (Shaoqing Ren y col. 2015) es un modelo de detección de objetos de dos pasadas, es decir, el cálculo de las *bounding boxes* y de las etiquetas o clases (y sus *scores*) asociadas se realiza mostrándole a la red dos veces la misma imagen.

Se fundamenta en un ***backbone*** entrenado en clasificación de imágenes, en este caso ResNet-50, que utiliza como extractor de características o *feature extractor*. A diferencia de SSD, la proposición de regiones de interés (RoI) se realiza mediante el algoritmo de *Sliding Window* (Aakarsh Yelisetty 2020) con 9 *anchor boxes* centradas en la ventana.

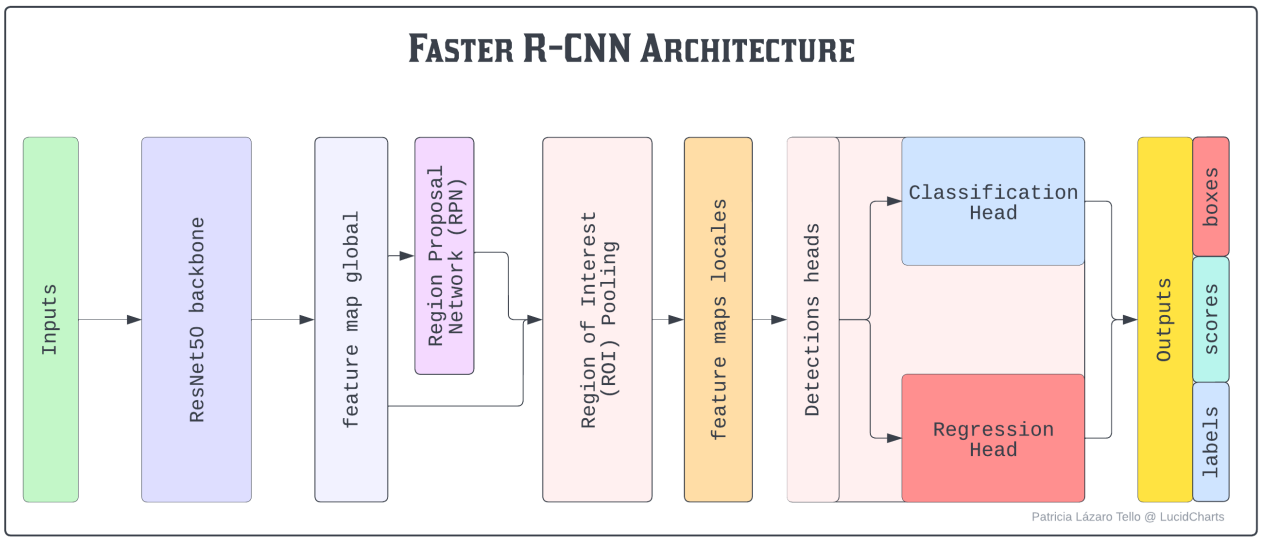


Figura 4.4: Arquitectura del modelo *Faster* R-CNN. El entrenamiento de la *Region Porposal Network* y las cabezas de detección se realiza de forma alternativa.
Fuente: elaboración propia

De esta forma se obtienen las propuestas utilizando un regresor para escoger las regiones que mejor se adapten al objeto. No obstante, el número de regiones propuestas es muy alto, por lo que se utiliza un clasificador binario para filtrar las regiones que tienen un objeto de las que no; es decir, se realiza una primera clasificación en *foreground* y *background*. Esta clasificación se denomina *objectness*.

Adicionalmente, como se realiza la clasificación de las regiones, se obtiene también su *score* o probabilidad de que efectivamente la región contenga un objeto. Este *score* es utilizado para escoger únicamente las top-k propuestas como *Region of Interest* (RoI).

Una vez se ha llevado a cabo la proposición de regiones mediate la *Region Proposal Network* (RPN), se realiza el proceso de selección de *features* asociadas a cada región mediante *Region of Interest* (RoI) *Pooling*.

Esta técnica (Sambasivarao. K 2019) consiste en utilizar un único *feature map* para todas las regiones de interés, de forma que con una sola pasada del extractor de características y las regiones propuestas por la RPN, se obtienen los mapas de características de cada región.

Estos *feature maps* locales son enviados tanto al regresor, que calcula el tamaño, *aspect ratio* y posición de la *bounding box*, como al clasificador, que asigna una clase o etiqueta de acuerdo a la probabilidad o *score* obtenido.

Las fases o pasadas en que se divide el modelo corresponden con una primera fase de extracción de características y proposición de regiones, y una segunda fase de cálculo de *bounding boxes* y clasificación de las mismas. El entrenamiento se realiza fijando una fase y entrenando la otra; por ejemplo, se fija la RPN y se entrenan las cabezas detectoras.

4.2.3. Modelos híbridos

Para la creación de los modelos híbridos se han utilizado como ***backbone*** los modelos de detección de objetos SSD (definido en la sección 4.2.1) y *Faster* R-CNN (definido en la sección 4.2.2). Estos modelos de detección presentan una arquitectura de las cabezas detectoras similar que permite equiparar y comparar los modelos híbridos entre sí fácilmente.

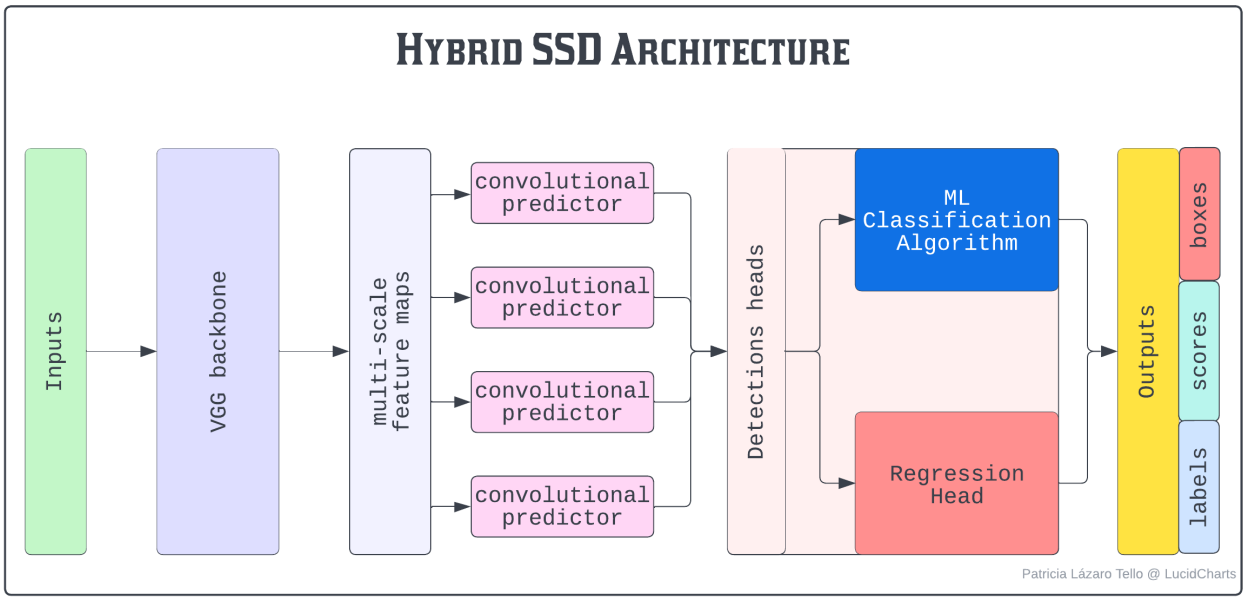


Figura 4.5: Arquitectura del modelo híbrido sobre SSD
Fuente: elaboración propia

En ambos casos se sustituye la red neuronal que actúa como cabeza clasificadora por un algoritmo de *Machine Learning* tradicional. Los algoritmos de clasificación elegidos en el capítulo 2: Estado del

Arte son **SVM** y ***Nearest Neighbor***.

El algoritmo k**-NN**, k vecinos más cercanos o k *nearest neighbors* (Jordi Gironés Roig y col. 2017) es un algoritmo de aprendizaje supervisado y *lazy* de clasificación en que, para cada instancia nueva a clasificar, se calcula la distancia con todas las muestras de entrenamiento y se clasifica según la clase mayoritaria de las k distancias más cercanas.

Este algoritmo es *lazy* o perezoso porque el entrenamiento se produce en el momento de clasificar nuevas instancias. La mayor debilidad del algoritmo es la lentitud del proceso de clasificación, dado que se ha de comparar la nueva instancia con todas las instancias de entrenamiento. Además, el rendimiento puede verse doblemente afectado en caso de que el *feature map* de entrada tenga del orden de cientos o miles de dimensiones (*curse of dimensionality* o la maldición de la dimensión).

Las máquinas de soporte vectorial, *Support Vector Machines* o **SVM** (Jordi Gironés Roig y col. 2017) es un algoritmo de aprendizaje supervisado de clasificación lineal y no lineal considerado de los más potentes en reconocimiento de patrones. El algoritmo SVM busca el hiperplano óptimo que maximiza el margen entre las clases del *corpus* de entrenamiento.

Para encontrar el hiperplano óptimo, SVM se apoya en las funciones *kernel*, que transforman un problema no lineal en uno lineal, y en la maximización del margen, que corresponde con la maximización de la distancia entre los puntos pertenecientes a las diferentes clases.

Una de las mayores ventajas de SVM es que es un algoritmo muy efectivo en *corpus* con muchas dimensiones; por contra, es muy sensible al ruido en el conjunto de datos y el hiperplano de separación es difícil de entender.

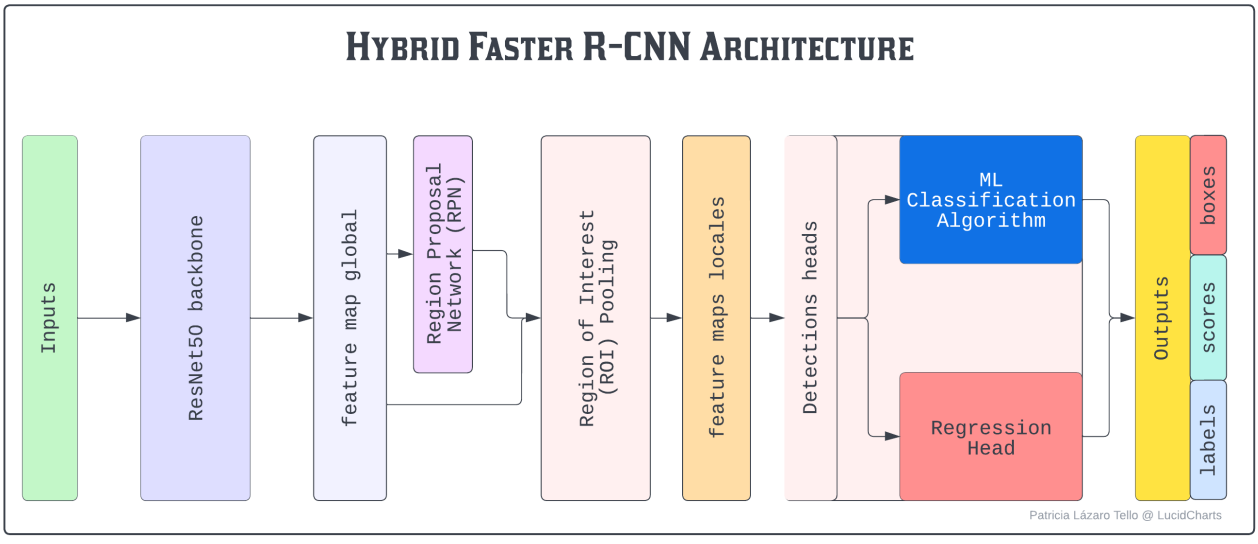


Figura 4.6: Arquitectura del modelo híbrido sobre *Faster* R-CNN
Fuente: elaboración propia

4.3. Proceso de entrenamiento de los modelos

Para entrenar los modelos –tanto los modelos de *Deep Learning* como los modelos híbridos –se ha utilizado un enfoque basado en *Transfer Learning* y *Fine tuning* (Sinno Jialin Pan y Qiang Yang 2010), realizando en primer lugar el entrenamiento sobre el conjunto de datos de ***ResortIT*** y después sobre ***ZeroWaste***.

Se han establecido cinco niveles de *transfer learning*, que se definen según las capas de la red neuronal que están congeladas y descongeladas:

- **Nivel 0**: se entrenan todas las capas, es el equivalente a *train from scratch*.
- **Nivel 1**: se entrenan únicamente las capas que corresponden a las cabezas detectoras del modelo, tanto la cabeza de clasificación como la de regresión.
- **Nivel 2 a Nivel 5**: se descongelan progresivamente más capas del modelo; el número de capas afectadas depende del modelo utilizado, dado que no todos contienen el mismo número de capas. Como normal general, se ha procurado incrementar linealmente el número de parámetros entrenables.

Nivel de *transfer learning*	Parámetros entrenables (SSD/*Faster* R-CNN)	% de parámetros entrenados (SSD/*Faster* R-CNN)
Nivel 1	400.986 / 15.222.799	1.68 % / 35.19 %
Nivel 2	935.634 / 16.406.814	3.92 % / 37.92 %
Nivel 3	1.953.106 / 19.753.246	8.18 % / 45.66 %
Nivel 4	3.395.666 / 34.717.982	14.22 % / 80.25 %
Nivel 5	16.244.306 / 34.717.982	68.03 % / 80.25 %
Nivel 0	23.879.570 / 43.261.278	100 % / 100 %

Cuadro 4.1: Afectación del nivel de *transfer learning* a los modelos de *Deep Learning*. La cabeza de detección de *Faster* R-CNN es considerablemente más grande que la de SSD, así como el propio modelo.

Utilizando este sistema y su nomenclatura asociada, se establece una secuencia de entrenamiento en dos fases: pre-entrenamiento y entrenamiento. Ambas fases tienen la misma estructura y etapas, el cambio se produce en el conjunto de datos utilizado en entrenamiento y validación, así como la métrica a optimizar en el ciclo de entrenamiento.

Un ciclo de entrenamiento consta del entrenamiento sucesivo de un modelo, desde el máximo nivel de congelación (nivel 1 de *transfer learning*) hasta el mínimo (nivel 5 de *transfer learning*), seguido del entrenamiento del modelo completo (nivel 0 de *transfer learning*) y una fase de test para comprobar la bondad del ajuste sobre el *dataset* de test.

La métrica a optimizar en cada fase del entrenamiento difiere: en la primera fase de pre-entrenamiento con el *corpus* de **ResortIT** se utiliza la *loss* de entrenamiento, calculada internamente por los modelos,

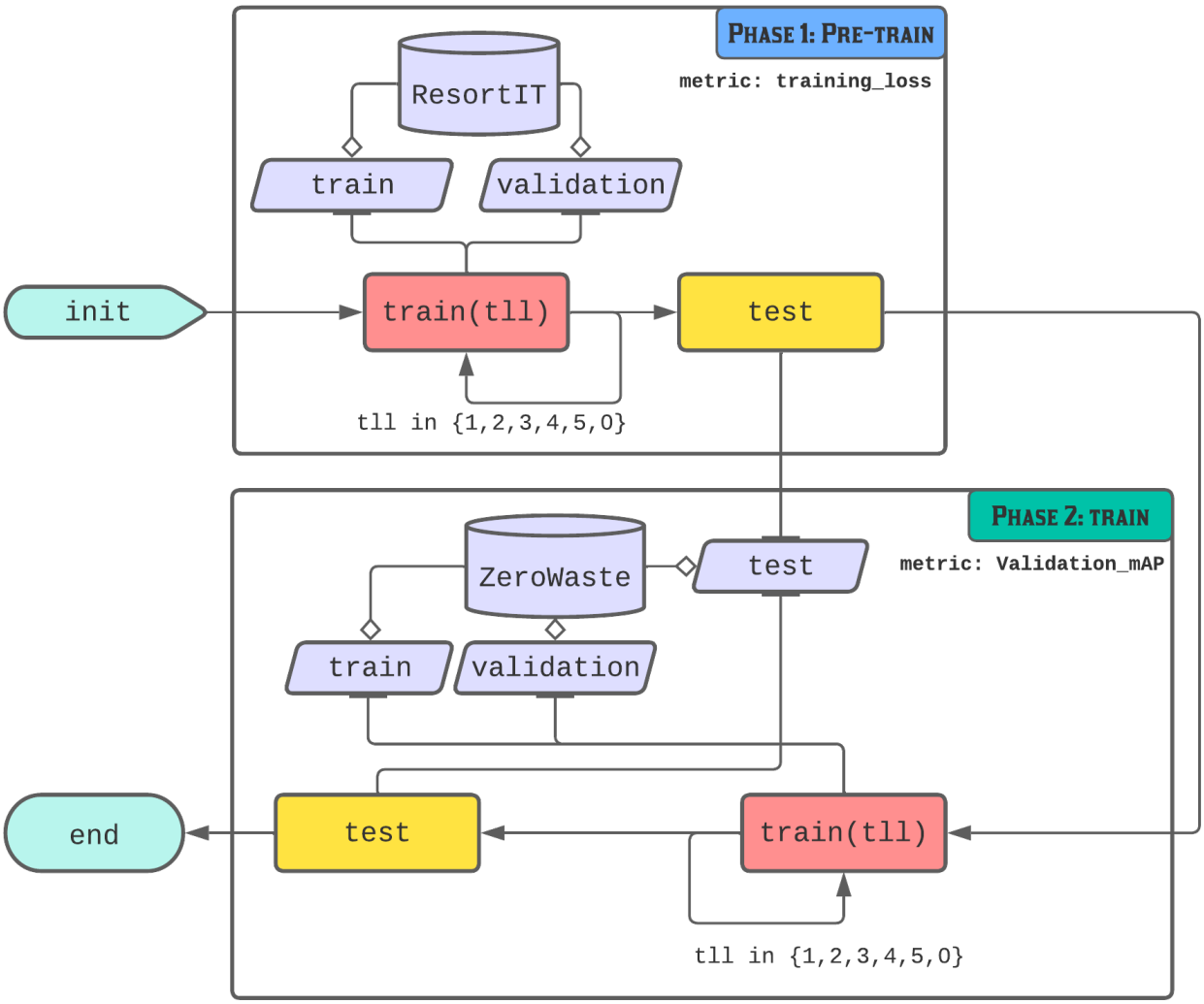


Figura 4.7: El conjunto de datos de test de *ZeroWaste* es utilizado para comprobar la bondad del ajuste en ambas fases durante la fase de test.
Fuente: elaboración propia

mientras que para el entrenamiento con **ZeroWaste** se utiliza la mAP de validación.

Esto se debe a que calcular la *mean average precision* es un **proceso costoso** que consume mucho tiempo (se tarda más en calcular la mAP de un 10 % del *dataset* de validación de ResortIT que en completar el resto de la *epoch*), principalmente debido a que todos los cálculos asociados han de computarse en la CPU y es una métrica compleja en ese respecto.

Dadas las limitaciones temporales del proyecto, se ha decidido utilizar la *loss* de entrenamiento para la primera fase de pre-entrenamiento y calcular la mAP de un 10 % aleatorio del conjunto de validación cada 10 *epochs*.

Para la fase de entrenamiento con el *corpus* de ZeroWaste se ha decidido utilizar la métrica de mAP como medida a optimizar, calculándose cada *epoch* con la totalidad del *dataset* de validación de

ZeroWaste, que es significativamente más pequeño que el de ResortIT.

Adicionalmente, se ha acortado el tiempo de entrenamiento, es decir, se ha limitado el número de *epochs*, siempre respetando el ratio de *epochs*/parámetros entrenables del modelo.

Por último, cabe destacar que dada la diferencia de tamaños y número de parámetros entrenables de los modelos de *Deep Learning* elegidos, la arquitectura *Faster* R-CNN se ha visto más afectada por la limitación temporal que la arquitectura SSD, que en el mismo tiempo ha podido completar más *epochs*.

Respecto a los modelos híbridos, se construyen sobre un modelo entrenado según el método anterior de dos fases. El entrenamiento de estos modelos equivale al entrenamiento del clasificador: se realizan predicciones en el modelo de *Deep Learning*, obteniendo el *feature map* y las *bounding boxes* candidatas.

Las *bounding boxes* se pasan por una fase de *matching* para obtener las etiquetas reales que corresponden a cada una de ellas; después, los *feature maps* y las etiquetas de *ground truth* obtenidas se utilizan en el entrenamiento de clasificador.

4.4. Problemas encontrados

Tal y como se deja entrever en la sección 4.3: Proceso de entrenamiento de los modelos, durante el desarrollo y puesta en marcha del paquete de ***Waste Detection System*** han surgido problemáticas e inconvenientes de diferentes naturalezas. En esta sección se realiza una recopilación de dichos problemas, con una descripción exhaustiva de las causas y soluciones aplicadas.

Configuración del entorno En primer lugar, surgieron problemas de configuración con la máquina virtual de AzureLabs proporcionada por la universidad. Esta máquina tiene un sistema operativo Unix y hasta el momento se había trabajado en un sistema Windows, por lo que se tuvo que adaptar la configuración (*scripts* y dependencias) al ecosistema Unix.

Poder computacional Este fue el principal problema encontrado en el desarrollo, el que más tiempo consumió durante la ejecución del proyecto y también el más complicado y difícil de solventar.

	PC personal	***Shadow PC***
CPU	Intel Xeon X5650 @ 2.67GHz	AMD EPYC 7543P 32-Core @ 2.80GHz
RAM	18GB	16GB
GPU	NVIDIA RTX 3060 Ti	NVIDIA RTX A4500
VRAM	8GB	20GB

Cuadro 4.2: Especificaciones de las máquinas de computación

El entrenamiento de modelos de detección de objetos es una tarea computacionalmente costosa que requiere de *hardware* especializado (tarjetas de vídeo o GPUs con gran cantidad de memoria

dedicada) y una CPU suficientemente potente para calcular la *Mean Average Precision* en un tiempo razonable; por tanto, no puede ejecutarse en un ordenador convencional.

Por tanto, el proyecto requiere de una máquina con suficiente poder computacional tanto en CPU como en GPU y almacenamiento en RAM y VRAM. Estos requisitos se cubren pobremente en la máquina virtual proporcionada por la universidad, que resulta lenta en todos los aspectos. Finalmente, se ha decidido hacer uso de un ordenador personal y un otro en la nube (*cloud computing*) de *Shadow Cloud Computing*, ambos con Windows 10, cuyas especificaciones se encuentran en la tabla 4.2.

Dadas las características de ambas máquinas, se ha decidido utilizar el PC personal para el entrenamiento del modelo SSD por tratarse de un *hardware* más modesto y un modelo más ligero; el *Shadow PC* se utiliza por tanto para el entrenamiento del modelo *Faster* R-CNN, significativamente más pesado que SSD.

Limitaciones temporales de entrenamiento Adicionalmente, para obtener resultados en un tiempo razonable se ha utilizado la métrica de *training loss* en el pre-entrenamiento, que aumenta el riesgo de *overfitting* respecto a ResortIT y proporcionará resultados pobres en las métricas de validación de ResortIT y especialmente en la métrica de test sobre ZeroWaste. El razonamiento detrás de esta decisión se encuentra explicado en la sección 4.3: Proceso de entrenamiento de los modelos.

Adicionalmente, se ha llevado a cabo un reescalado de las imágenes a 640x640 píxeles con el objetivo de acelerar aún más el proceso de entrenamiento –aunque esto supone también una cierta pérdida de precisión.

Capítulo 5

Resultados obtenidos

En este capítulo se presentan los resultados obtenidos del entrenamiento propuesto en el capítulo 4: Entrenamiento de los modelos seleccionados, tratando tanto las propias métricas que evalúan la bondad de los modelos como la huella de carbono y el tiempo requerido de dicho entrenamiento. Para evaluar la bondad del ajuste de los modelos se utiliza la métrica ***mean Average Precision*** (mAP) del conjunto de datos de validación de los correspondientes *datasets*, y la mAP del conjunto de datos de test del *dataset* ZeroWaste.

5.1. Elección de hiperparámetros

Respecto a la elección de hiperparámetros, se ha decidido utilizar los hiperparámetros por defecto de los modelos *Faster* R-CNN y SSD como una primera aproximación al entrenamiento de este tipo de detectores de objetos. Para cada *Transfer Learning Level*, se ha utilizado el *tuner* que ofrece Pytorch Lightning para obtener el *learning rate* inicial óptimo y el *batch size* más grande permitido, que es el que después se ha utilizado en el entrenamiento con ánimo de acelerar los cálculos. En el anexo C se puede encontrar los *learning rate* y *batch size* utilizados en el entrenamiento.

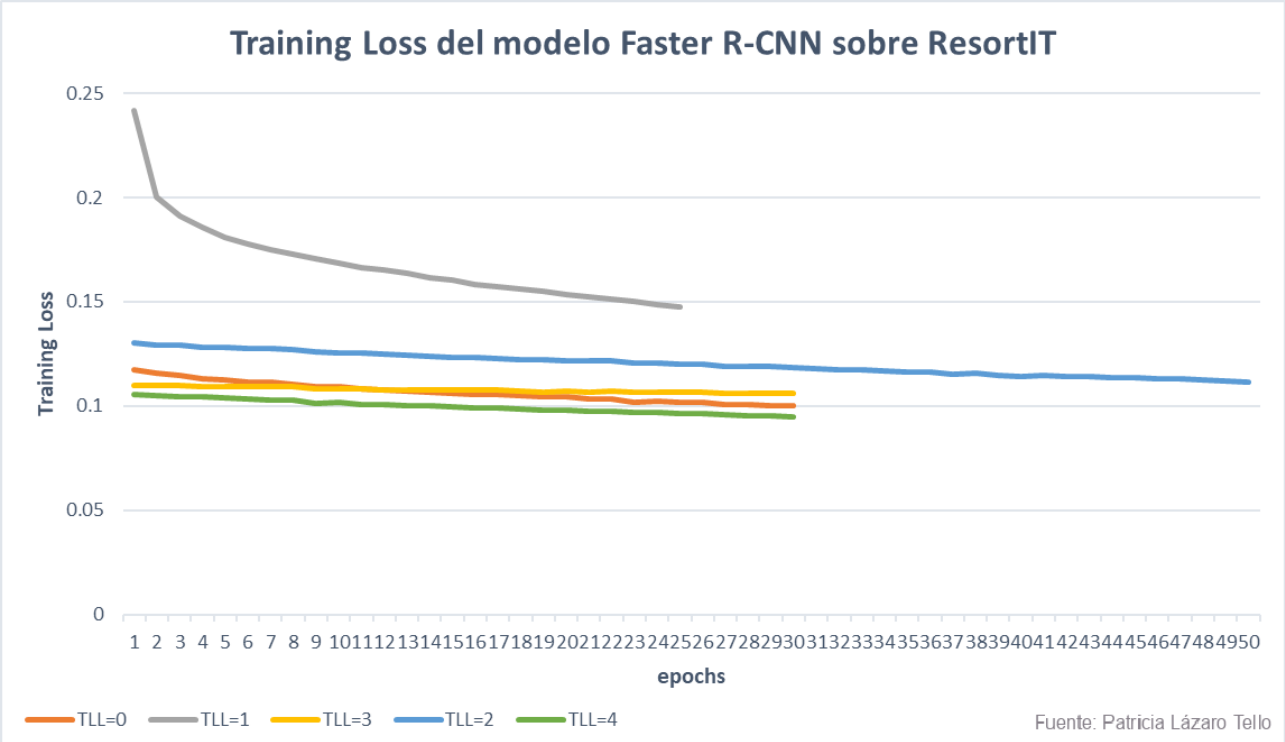
En cuanto al optimizador y *scheduler* de *learning rate*, se ha utilizado el optimizador **Adam** en todas las fases del pre-entrenamiento y posterior entrenamiento, y se ha configurado el *scheduler* como un reductor de *learning rate* cuando la métrica deja de mejorar durante 3 *epochs* seguidas, disminuyendo el *learning rate* tal que $lr_{n+1} = 0{,}9 * lr_n$.

5.2. Modelo *Faster* R-CNN

El modelo *Faster* R-CNN produce mejores resultados de entrenamiento (ver figura 5.1) y validación (ver figura 5.2) sobre el *dataset* ResortIT que sobre ZeroWaste.

Esta diferencia tan notable en los resultados de un modelo sobre los dos conjuntos de datos se debe a un conjunto de circunstancias, tales como la **distinta complejidad de los conjuntos de datos** y la diferencia entre las **funciones de optimización**, así como las ya mencionadas en la sección 4.4: Problemas encontrados.

Por un lado, ZeroWaste se trata de un conjunto de datos marcadamente complejo, con anotaciones



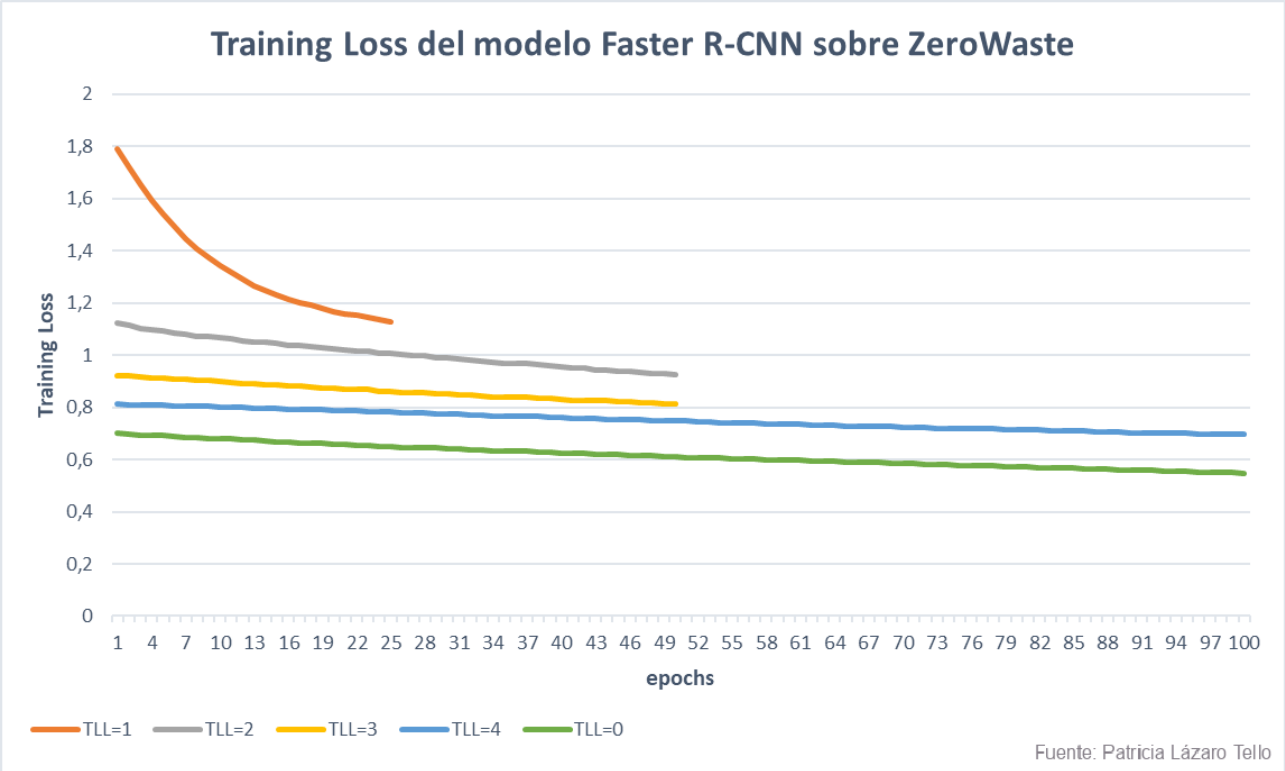
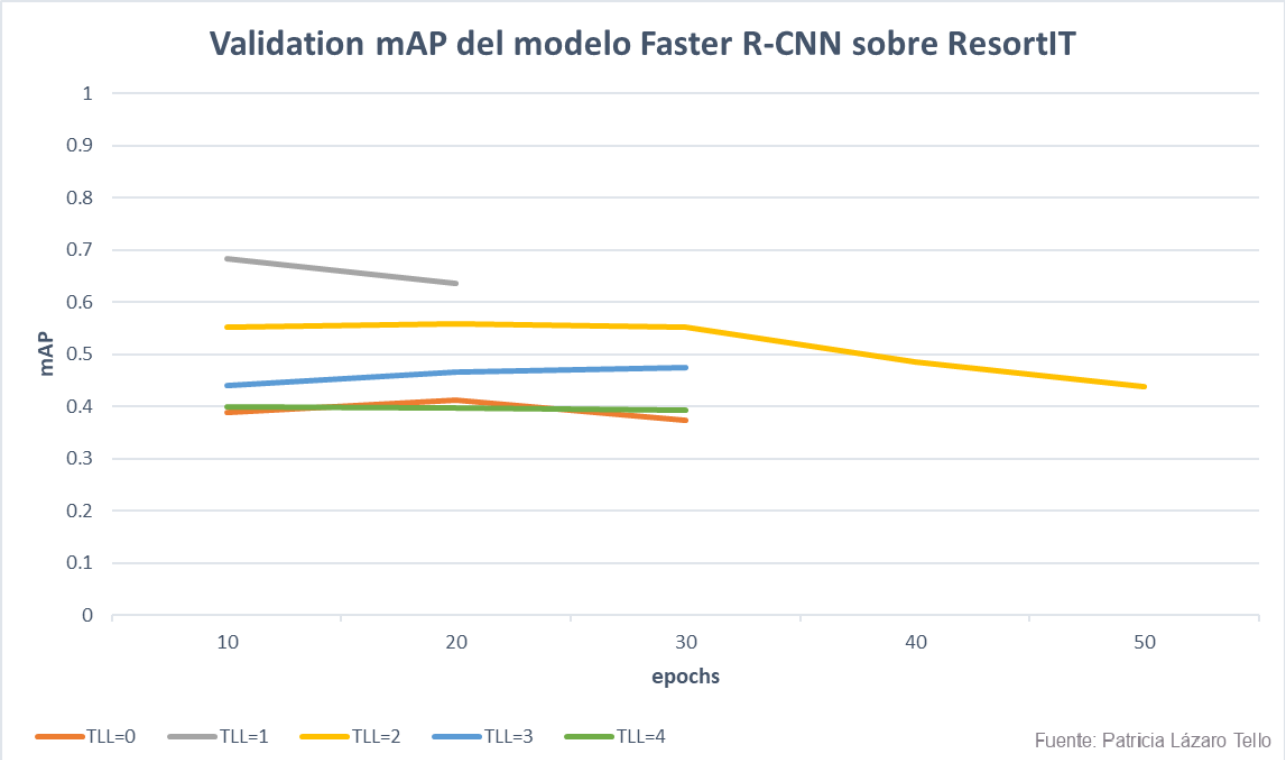


Figura 5.1: Comparativa de la función de pérdida del modelo *Faster* R-CNN sobre los conjuntos de ResortIT y ZeroWaste.
Fuente: elaboración propia.



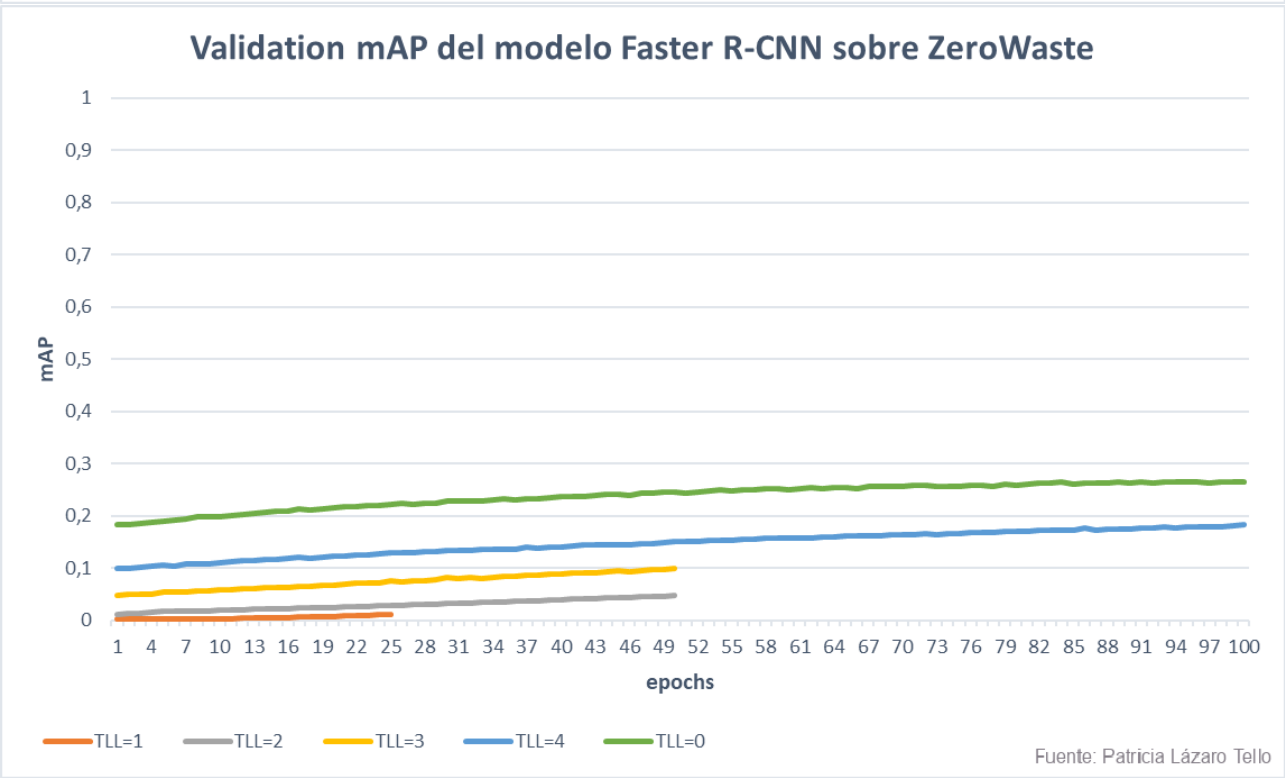


Figura 5.2: Comparativa de la *mean Average Precision* del modelo *Faster* R-CNN sobre los conjuntos de ResortIT y ZeroWaste.
Fuente: elaboración propia.

superpuestas y de distintos tamaños; es decir, se trata de un *dataset* no sintético y que, por tanto, captura las situaciones que se dan en la realidad.

ResortIT, por otro lado, es un conjunto de datos sintético con un número limitado de anotaciones por imagen, mayormente sin superposiciones de dichas imágenes y con tamaños de anotaciones similares. En este sentido, la red neuronal converge hacia una solución óptima en un menor número de *epochs*.

Predicciones versus Anotaciones reales

Fuente: Patricia Lázaro Tello

Figura 5.3: Comparativa de las predicciones y anotaciones reales en *Faster* R-CNN sobre ZeroWaste. Fuente: elaboración propia.

Respecto a la diferencia en las funciones de optimización, el modelo entrenado sobre ResortIT utiliza la función de *loss* de entrenamiento mientras que el modelo entrenado sobre ZeroWaste utiliza la función de mAP de validación.

Dataset	**TLL=1**	**TLL=2**	**TLL=3**	**TLL=4**	**TLL=5**	**TLL=0**
ResortIT	0.0101	0.0072	0.0066	0.0068	N/A	0.0018
ZeroWaste	0.0126	0.0579	0.1099	0.1800	N/A	0.2701

Cuadro 5.1: Valores de mAP de validación del modelo *Faster* R-CNN sobre el dataset correspondiente y mAP de test sobre ZeroWaste

Esto implica que el modelo entrenado sobre ResortIT se ajusta más a los datos de entrenamiento que el modelo entrenado sobre ZeroWaste, que pretende generalizar y obtener conocimiento para detectar todo tipo de papel y plástico, sin ceñirse específicamente a los datos de entrenamiento.

Este sobreajuste u *overfitting* sobre ResortIT se puede observar con claridad en la figura 5.2, en

la que se aprecia que la mAP de validación no es monótona creciente, es decir, que cuanto más se entrena el modelo sobre este conjunto de datos, menos capacidad de generalizar tiene.

En el caso del modelo entrenado sobre ZeroWaste, este sobreajuste u *overfitting* no se da; a grandes rasgos se puede afirmar que la mAP de validación es monótona creciente porque el modelo aprende a generalizar cuanto más se entrena. Cabe destacar que llegado cierto punto del entrenamiento (a partir del *Transfer Learning Level* 4), el modelo comienza a aprender a un ritmo superior, llegando a alcanzar una mAP de validación de 0.26.

Por último, conviene resaltar que en la tabla 5.1 se aprecia de nuevo el sobreajuste del modelo entrenado sobre ResortIT y el aprendizaje progresivo del entrenamiento sobre ZeroWaste. Así mismo, también es necesario enfatizar que, aunque los resultados sobre el conjunto de datos de test son más positivos en el entrenamiento sobre ZeroWaste que sobre ResortIT, siguen siendo resultados muy pobres. Sería necesario seguir entrenando el modelo con *data augmentation* o ajustando mejor los hiperparámetros propios del modelo; por ejemplo, cambiando los umbrales de IoU y clasificación. También tendría un efecto positivo utilizar las imágenes sin reescalado.

Dataset	**TLL=1**	**TLL=2**	**TLL=3**	**TLL=4**	**TLL=5**	**TLL=0**	**Total**
ResortIT	8.35h	13.72h	12.15h	11.58h	N/A	13.32h	59.12h
ZeroWaste	2.829h	5.96h	6.746h	20.15h	N/A	13.66h	49.34h

Cuadro 5.2: Tiempos de entrenamiento del modelo *Faster* R-CNN en *Shadow* PC

En cuanto al **tiempo de entrenamiento** (ver tabla 5.2), el procesado de las imágenes de entrenamiento y validación de ResortIT lleva casi 1 hora por *epoch* en el PC personal –considerando que el *batch size* es muy inferior en esta configuración por tratarse de una GPU menos potente, –utilizando 1 hora más para el cómputo de la *mean Average Precision* del conjunto de validación. El *dataset* ZeroWaste, por ser considerablemente más pequeño, tarda 30 minutos en el procesado de las imágenes y 45 minutos en el cómputo de la mAP.

Respecto a la **huella de carbono** (ver tabla 5.3), se observa que la utilización del procesador (CPU) del PC y su correspondiente huella aumentan al entrenar sobre el conjunto de datos de ZeroWaste; esto se debe a que se ejecuta con más frecuencia el cómputo de la mAP al entrenar sobre ZeroWaste que al entrenar sobre ResortIT.

Dataset	**CPU**	**GPU**	**RAM**	**Total energía consumida**	**Huella de carbono**
ResortIT	6.95 kW/h	9.82 kW/h	0.08 kW/h	16.85 kW/h	3.2 kg de CO_2
ZeroWaste	5.7 kW/h	5.7 kW/h	0.06 kW/h	11.45 kW/h	2.18 kg de CO_2

Cuadro 5.3: Huella de carbono del entrenamiento del modelo *Faster* R-CNN

5.3. Modelo SSD

El modelo SSD produce mejores resultados de entrenamiento (ver figura 5.4) y validación (ver figura 5.5) sobre el *dataset* ResortIT que sobre ZeroWaste. Como se explica en la sección 5.2, esto se debe fundamentalmente a las diferencias ya mencionadas entre los conjuntos de datos.

Respecto al modelo entrenado sobre ResortIT, en la gráfica 5.5 se observa que, salvo para el modelo entrenado con *Transfer Learning Level* 1, la mAP de validación se mantiene estable en torno a 0.3-0.4, sin llegar a disminuir ni aumentar considerablemente. Para TLL=1, el empeoramiento drástico de la mAP junto con la mejora de la *loss* de entrenamiento indican que el modelo padece de *overfitting*; en el resto de casos el modelo deja de aprender de forma significativa de los datos de entrenamiento, evitando el *overfitting*.

Dataset	**TLL=1**	**TLL=2**	**TLL=3**	**TLL=4**	**TLL=5**	**TLL=0**
ResortIT	0.0169	0.0135	0.0161	0.0144	0.0161	0.0012
ZeroWaste	0.0178	0.0189	0.0195	0.0207	0.0588	0.0778

Cuadro 5.4: Valores de mAP de validación del modelo SSD sobre el dataset correspondiente y mAP de test sobre ZeroWaste

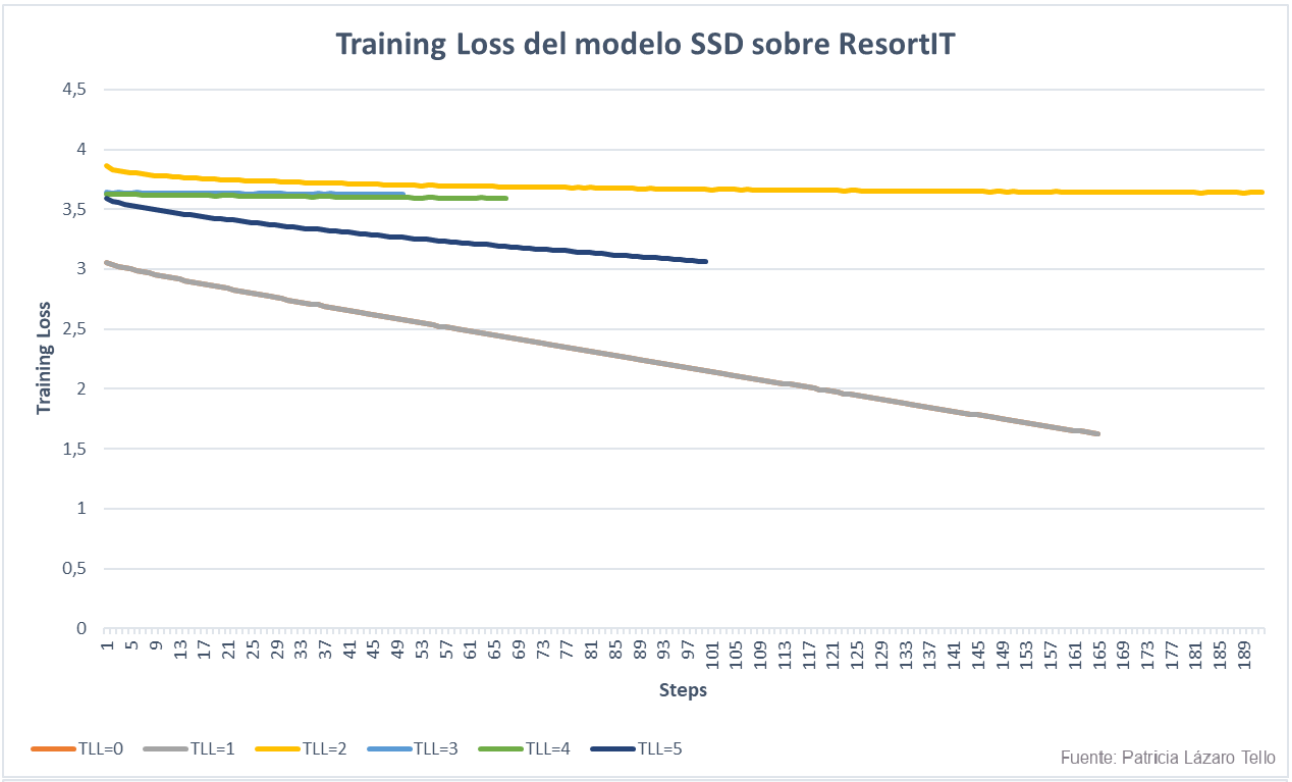
En cuanto al modelo entrenado sobre ZeroWaste, se observa el mismo fenómeno que afecta al modelo *Faster* R-CNN: la *loss* de entrenamiento es drásticamente superior que en el modelo entrenado sobre ResortIT y la mAP de validación es muy inferior a la obtenida sobre ResortIT. El modelo apenas mejora a partir del *Transfer Learning Level* 2, pero tampoco empeora de forma significativa. Se puede concluir que no se produce *overfitting*; los resultados tan pobres que produce el modelo se pueden achacar a que los hiperparámetros propios del modelo no son óptimos, las imágenes no tienen la calidad suficiente o que el conjunto de anotaciones proporcionado es pequeño y debería aumentarse mediante técnicas de *data augmentation*.

Dataset	**TLL=1**	**TLL=2**	**TLL=3**	**TLL=4**	**TLL=5**	**TLL=0**	**Total**
ResortIT	11.35h	22.35h	5.52h	7.78h	11.63h	25.88h	84.51h
ZeroWaste	3.85h	24.85h	1.875h*	5.79h*	7.85h*	8.56h* 8.27h*	61.04h

Cuadro 5.5: Tiempos de entrenamiento del modelo SSD en el PC personal
*: resultados obtenidos en *Shadow* PC

En relación con el **tiempo de entrenamiento** (ver tabla 5.5), el procesado de las imágenes de entrenamiento y validación de ResortIT lleva aproximadamente 20 minutos por *epoch* en el PC personal, utilizando 20 minutos más para el cómputo de la *mean Average Precision* del conjunto de validación. El *dataset* ZeroWaste, por ser significativamente más pequeño que ResortIT, tarda 15 minutos en el procesado de imágenes y 10 minutos en el cómputo de la mAP.

Contrastando los tiempos de entrenamiento de SSD y *Faster* R-CNN, se aprecia una diferencia de un orden de magnitud entre los dos modelos. Esta diferencia no es tan marcada en la tabla 5.5, ya



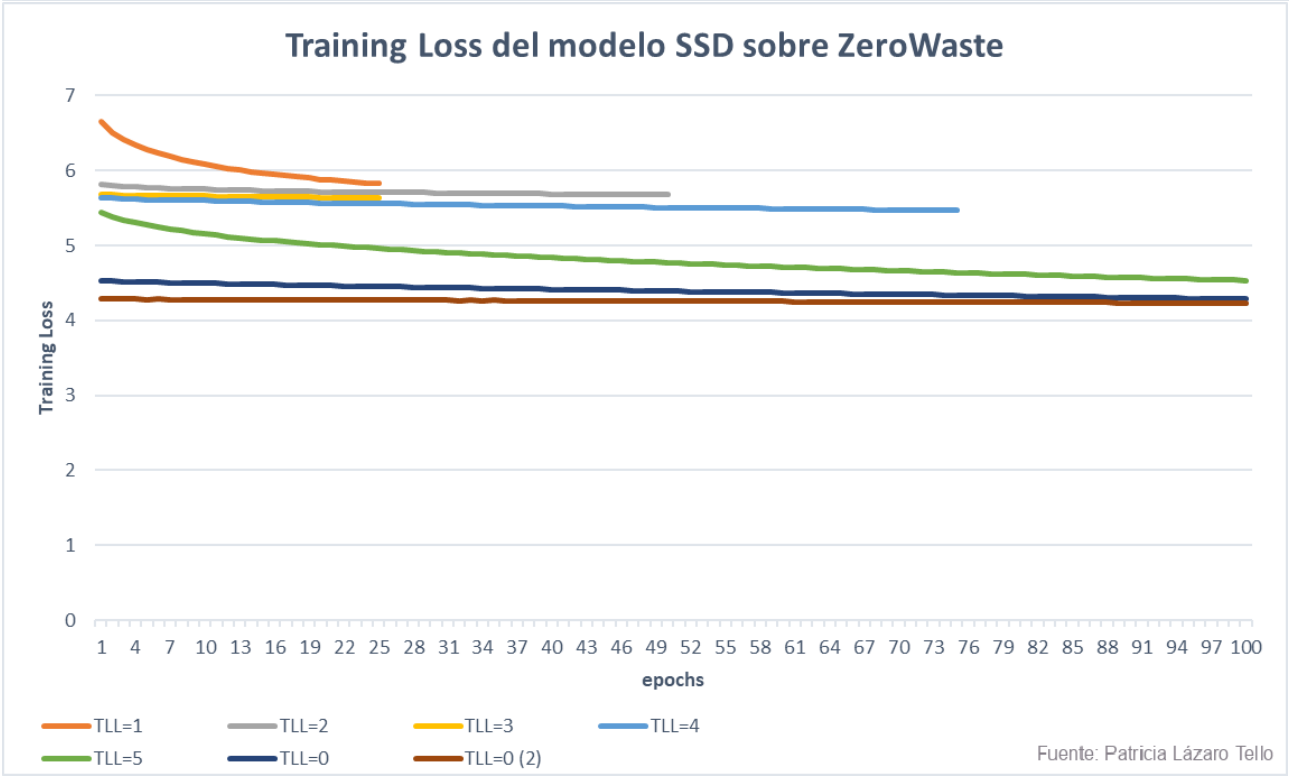
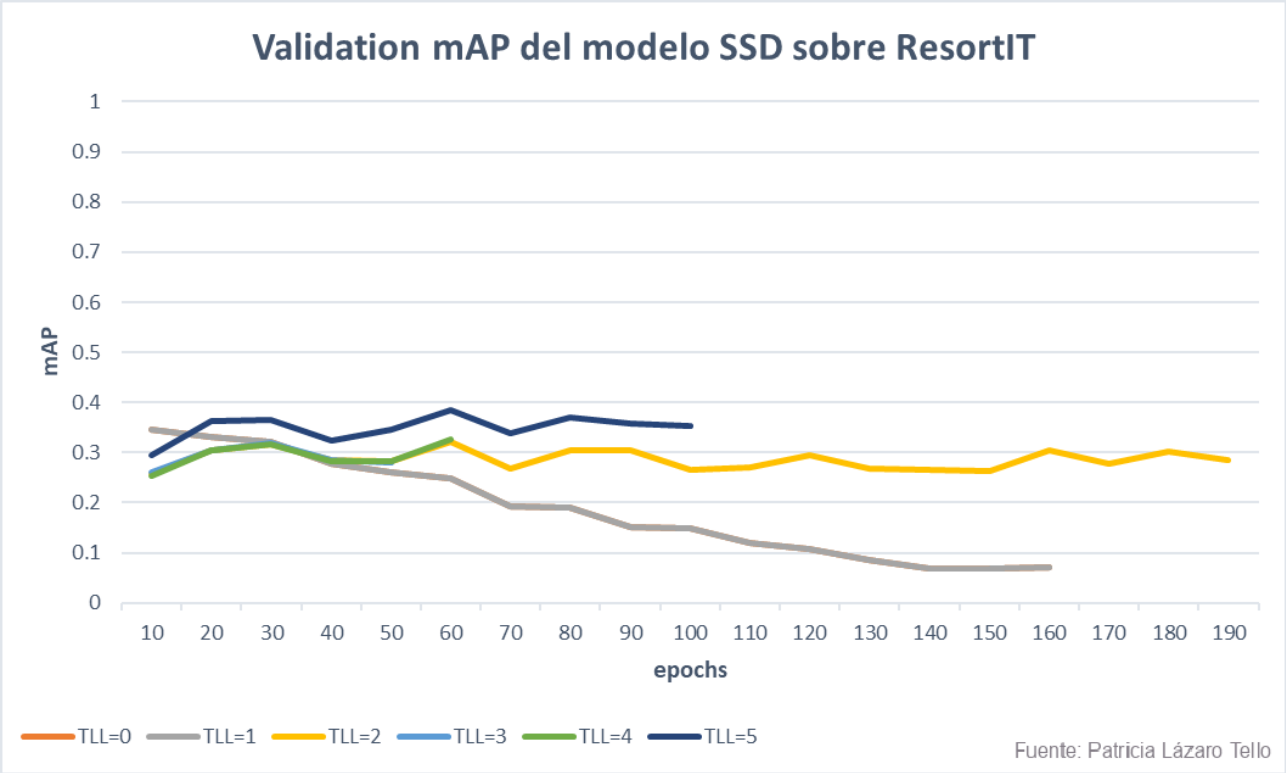


Figura 5.4: Comparativa de la función de pérdida del modelo SSD sobre los conjuntos de ResortIT y ZeroWaste.
Fuente: elaboración propia.



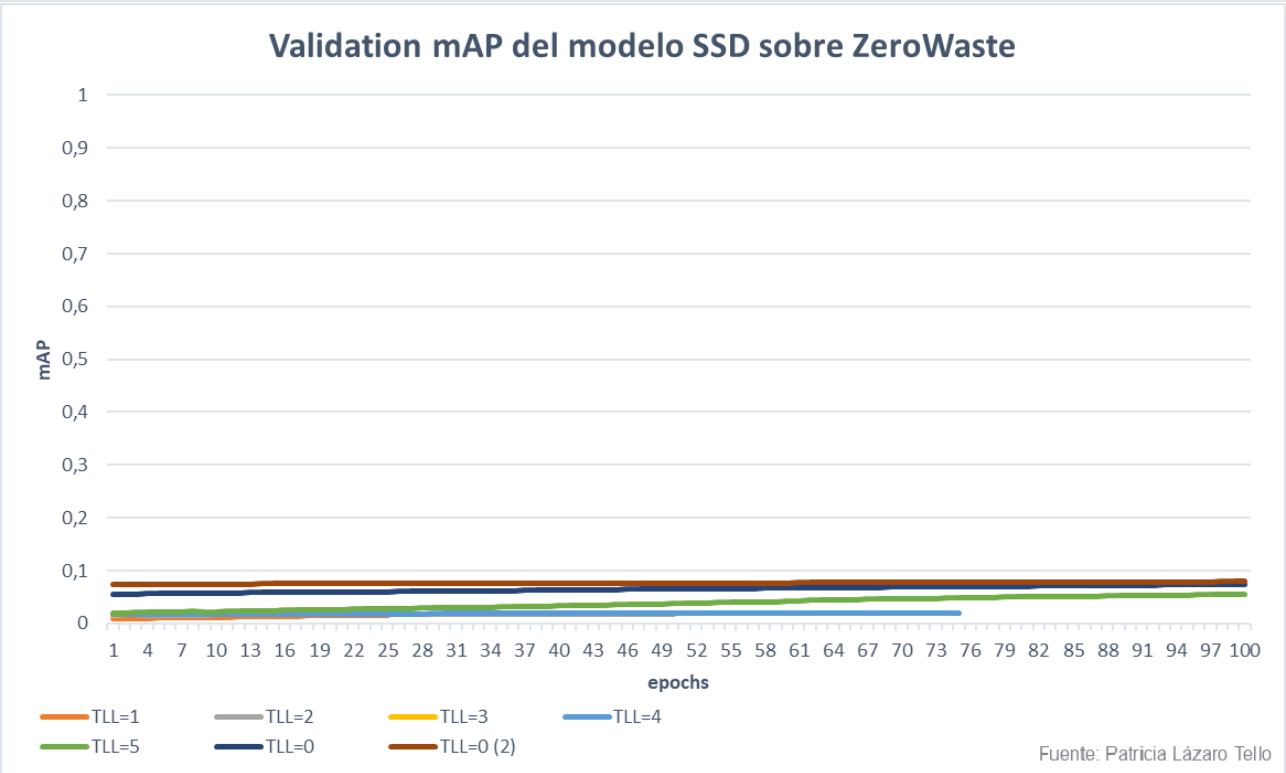


Figura 5.5: Comparativa de la *mean Average Precision* del modelo SSD sobre los conjuntos de ResortIT y ZeroWaste.
Fuente: elaboración propia.

que una parte de los cálculos del modelo SSD se ha realizado sobre el PC personal, más lento que *Shadow* PC. Este último ordenador es el que se ha utilizado para entrenar el modelo *Faster* R-CNN y parte del modelo SSD (indicado con un asterisco en la tabla).

Predicciones versus Anotaciones reales

Fuente: Patricia Lázaro Tello

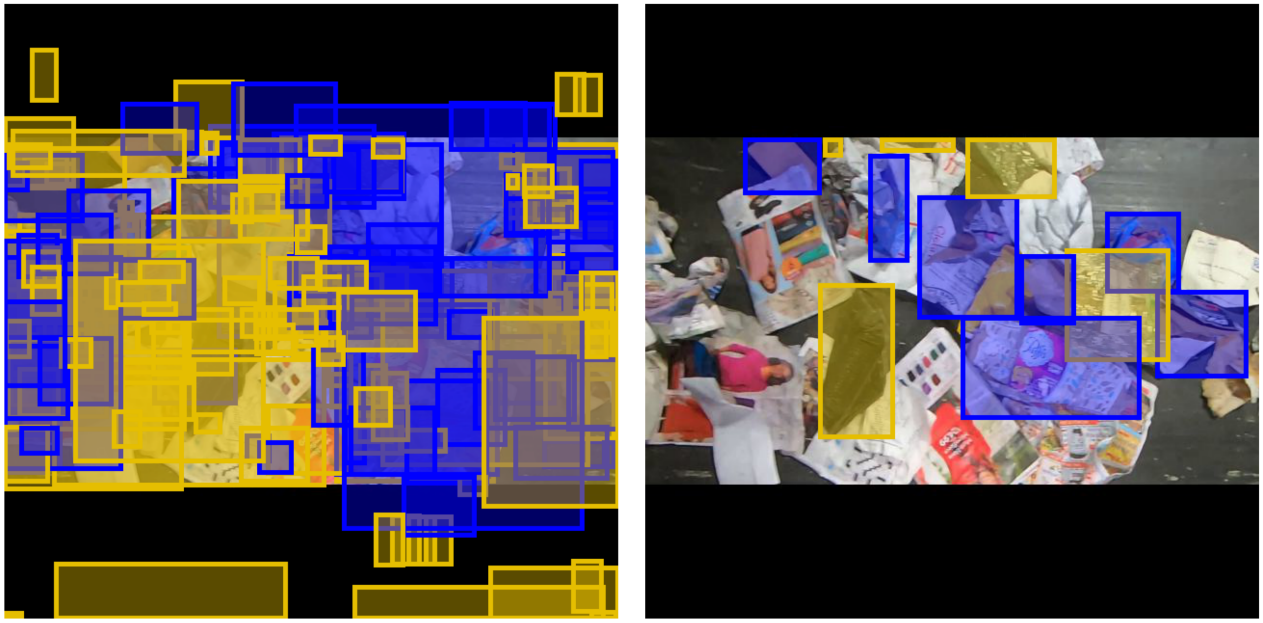


Figura 5.6: Comparativa de las predicciones y anotaciones reales en SSD sobre ZeroWaste.
Fuente: elaboración propia.

La diferencia de tiempos de entrenamiento entre los modelos es esperada, dado que SSD se trata de un modelo de una sola pasada que prioriza velocidad sobre precisión, mientras que *Faster* R-CNN es un modelo de dos pasadas cuyo foco se encuentra en su precisión y secundariamente en su velocidad (es el más rápido de la familia R-CNN).

Dataset	**CPU**	**GPU**	**RAM**	**Total energía consumida**	**Huella de carbono**
ResortIT	3.65 kW/h	6.54 kW/h	0.13 kW/h	10.23 kW/h	1.94 kg de CO2
ZeroWaste	6.12* kW/h	2.11* kW/h	0.1* kW/h	8.32* kW/h	1.58* kg de CO2

Cuadro 5.6: Huella de carbono del entrenamiento del modelo SSD
*: resultados parciales obtenidos en *Shadow* PC

Respecto a la **huella de carbono** (ver tabla 5.6), se observa que la utilización de la CPU del PC y su correspondiente huella aumentan al entrenar sobre el conjunto de datos de ZeroWaste, dado que se ejecuta el cómputo de la mAP con más frecuencia al entrenar sobre este *dataset* que sobre ResortIT.

5.4. Modelos híbridos

Los modelos híbridos planteados en el capítulo 4: Entrenamiento de los modelos seleccionados parten de un modelo –*Faster* R-CNN o SSD –ya entrenado sobre ResortIT y posteriormente ZeroWaste, al que se le añade un algoritmo de *Machine Learning* tradicional como cabeza de clasificación.

Debido a que los resultados obtenidos por *Faster* R-CNN (un mAP de test de 0.27) y SSD (un mAP de test de 0.08) son poco precisos, **no se ha podido obtener propuestas de detecciones** en el conjunto de datos de entrenamiento, imposibilitando la clasificación de las mismas y el entrenamiento del modelo de clasificación.

Una opción de bajo coste computacional y temporal explorada es el entrenamiento de un modelo de detección de objetos con un *backbone* propio y muy ligero, consistente en una capa convolucional y una de normalización. De esta forma, sólo se entrenan el generador de *bounding boxes* y la cabeza del modelo.

Por cuestiones de tiempo, sólo se ha entrenado un modelo SSD con *backbone* propio, por ser el más rápido de entrenar, pero el *framework* provee al usuario de la opción de entrenar un modelo *Faster* R-CNN con este *backbone*.

Adicionalmente, para acelerar el aprendizaje, se han utilizado los pesos obtenidos en el entrenamiento del modelo sobre ZeroWaste, y se ha realizado el entrenamiento únicamente sobre este conjunto de datos (es decir, no se ha llevado a cabo una fase de pre-entrenamiento con ResortIT).

No obstante, este entrenamiento tampoco ha proporcionado predicciones útiles y, en consecuencia, **no se han podido entrenar los modelos híbridos**.

5.5. Comparativa y conclusiones

En la tabla 5.7 se pueden observar los resultados de los modelos SSD y *Faster* R-CNN en cuanto a la métrica objetivo, el *Mean Average Precision* del conjunto de test de ZeroWaste. En primer lugar, se aprecia que SSD consigue mejores marcas que *Faster* R-CNN al ser entrenados los dos modelos sólo sobre ResortIT, mientras que al añadir el entrenamiento sobre ZeroWaste a la ecuación, *Faster* R-CNN obtiene resultados muy superiores a los de SSD.

De estas apreciaciones se puede inferir que **el modelo *Faster* R-CNN es más rápido aprendiendo y ajustándose a los datos que el modelo SSD**: ambos empiezan con un mAP similar en el primer nivel de *transfer learning* sobre ResortIT, pero la mAP de *Faster* R-CNN decae mucho más rápido que la mAP de SSD al avanzar el entrenamiento. A su vez, tras el primer nivel de entrenamiento sobre ZeroWaste, los dos modelos tienen métricas similares (el modelo SSD incluso mantiene cierta ventaja sobre el modelo *Faster* R-CNN), pero en seguida *Faster* R-CNN comienza a aumentar su mAP de test mientras SSD, aunque también incrementa los valores de la métrica, lo hace a un ritmo muy inferior.

En cuanto a tiempos de entrenamiento, la **comparación entre los entrenamientos de ambos modelos no se puede aplicar** dado que la mayor parte del entrenamiento del modelo SSD se ha llevado

a cabo sobre el PC personal, considerablemente menos potente que el *Shadow* PC.

Continuando con la comparativa entre modelos, se aprecia que en materia de tiempos de inferencia SSD funciona a **10FPS** (10 imágenes procesadas en 1 segundo; cada imagen tarda 91.53ms en ser procesada) mientras que *Faster* R-CNN obtiene sólo **4FPS** (4 imágenes procesada por segundo; cada imagen tarda 210.85ms en ser procesada).

Dataset	**Modelo**	**TLL=1**	**TLL=2**	**TLL=3**	**TLL=4**	**TLL=5**	**TLL=0**
ResortIT	*Faster* R-CNN	0.0101	0.0072	0.0066	0.0068	N/A	**0.0018**
	SSD	**0.0169**	**0.0135**	**0.0161**	**0.0144**	0.0161	0.0012
ZeroWaste	*Faster* R-CNN	0.0126	**0.0579**	**0.1099**	**0.1800**	N/A	**0.2701**
	SSD	**0.0178**	0.0189	0.0195	0.0207	0.0588	0.0778 0.0830

Cuadro 5.7: Comparativa los modelos *Faster* R-CNN y SSD sobre la mAP de test

En ningún caso se alcanza el objetivo de un detector de residuos en tiempo real (20 FPS), aunque cabe destacar que los resultados anteriores se han obtenido con el modelo sin optimizar para inferencia. Los modelos pueden ser exportados a formato ONNX (Microsoft 2018) o TensorFlow Lite (Google 2017) para maximizar el rendimiento y la portabilidad, mejorando el número de imágenes procesadas por segundo e incluso consiguiendo rendimiento a tiempo real.

En este caso, se ha decidido realizar la exportación a formato ONNX, mejorando sustancialmente los resultados obtenidos en cuanto a velocidad de inferencia, como se muestra en la tabla 5.8. El modelo SSD procesa una imagen cada 67ms, acercándose al umbral de tiempo real establecido en 20 FPS, mientras que el modelo *Faster* R-CNN obtiene un tiempo de procesado de imágenes de 235.76ms –en este caso, el tiempo de procesado es ligeramente superior en el modelo ONNX que en el modelo Pytorch.

	Pytorch	**ONNX**
Faster R-CNN	4 FPS	4 FPS
SSD	10 FPS	14 FPS

Cuadro 5.8: Comparativa de velocidad de inferencia de los modelos según su exportación

Para realizar el *benchmark* de tiempos de inferencia, se ha ejecutado el código correspondiente a la predicción del *dataset* de test un total de 100 veces. Con estas iteraciones se pretenden resolver los problemas más comunes en *benchmarking* de modelos, como son que la GPU funcione más lenta al principio (***GPU warm-up***) o que el tiempo dedicado a transferir datos de memoria a la GPU sea variable.

Capítulo 6

Conclusiones y trabajos futuros

El **cambio climático** es un tema de actualidad que se agrava con cada año que pasa sin que el ser humano cambie sus costumbres y su estilo de vida. Dentro de los cambios en los hábitos que se presentan para frenar y revertir este fenómeno se encuentra el **reciclaje**, que genera un impacto muy pequeño en la vida diaria de las personas pero supone un efecto importante en la lucha contra el cambio climático.

Aunque gran parte del esfuerzo en la segregación de residuos recae sobre la población consumidora, también es necesario prestar atención a la categorización de residuos a nivel de los recolectores, ya que a veces se produce confusión en la población a la hora de reciclar según qué residuos o, en algunos casos, no es posible llevar a cabo la segregación a nivel de consumidor.

A este respecto y con el fin de recuperar y reciclar el mayor porcentaje posible de residuos reciclables, se propone el desarrollo de un sistema de detección y categorización de residuos en la fracción resto con la intención de una eventual puesta en marcha en una planta de segregación.

Este proyecto plantea las bases del *framework* de detección de residuos, realizando una primera aproximación a una API de entrenamiento y evaluación de modelos. Debido a falta de datos abiertos, sólo se ha podido entrenar modelos de detección de papel y plástico, pero el sistema está organizado para escalar con más categorías de forma orgánica.

Respecto a los modelos entrenados, se han realizado pruebas sobre el modelo de detección en dos pasadas ***Faster* R-CNN** y el modelo de detección en una pasada **SSD**. En cuanto a los modelos híbridos, aquellos con un modelo de *Deep Learning* como base con un clasificador de *Machine Learning* tradicional, no ha sido posible realizar un entrenamiento/evaluación, ya que requieren uno modelo base sólido que no se ha conseguido por falta de tiempo y de recursos.

Una de las conclusiones más importantes que se extrae del trabajo completado es la **potencia y tiempo necesarios para entrenar un sistema de detección de objetos** en imágenes frente a un sistema de clasificación de imágenes tradicional.

De los resultados obtenidos en el capítulo 5: Resultados obtenidos se deduce que la decisión de realizar un pre-entrenamiento con un conjunto de datos sintético marcadamente distinto del conjunto

de datos final no aporta una mejora sustancial en el rendimiento final del modelo.

También cabe remarcar en último lugar que del análisis de *datasets* y su literatura correspondiente en los capítulos 2 y 3 se desprende que hay una ausencia significativa de conjuntos de datos no sintéticos y abiertos que permitan este tipo de proyectos, lo cual introduce nuevas barreras a la investigación en temas de detección y categorización de reciclaje.

6.1. Cumplimiento de los objetivos iniciales

Los objetivos iniciales del trabajo se resumen en el **desarollo de un sistema de detección y clasificación de residuos en plantas de selección**; este objetivo se ha cumplido parcialmente, ya que el sistema de detección desarrollado no categoriza los seis tipos de residuos en que se dividen los restos, si no sólo dos: **papel** y **plástico**.

También se ha de considerar la calidad de los datos obtenidos, que no es suficiente para llevar a término el objetivo inicial: el conjunto de datos seleccionado sólo cuenta con residuos de papel y plástico e, idealmente, **el conjunto de datos objetivo debería mostrar residuos de todas las categorías mezclados**.

Por tanto, aunque es una primera aproximación, cabe destacar que en futuras líneas de trabajo se ha de tener en cuenta esta anotación para no entrenar los modelos sobre **conjuntos de datos estancos** (donde no todas las categorías están representadas a la vez).

6.2. Análisis del seguimiento de la planificación y la metodología

La metodología inicial propuesta en la sección 1.4 es la metodología **CRISP-DM**; el proyecto se ha adherido a la misma de acuerdo con las entregas parciales. En primer lugar, se ha realizado un análisis exhaustivo del área de estudio en el capítulo 2: Estado del Arte.

En este mismo capítulo se obtiene un conocimiento preliminar de los datos, que se expande en el capítulo 3: Selección del conjunto de datos con el *Exploratory Data Analysis* o EDA, donde se seleccionan los conjuntos de datos a utilizar y se preparan para su uso.

En el capítulo 4: Entrenamiento de los modelos seleccionados se seleccionan los modelos a entrenar y se configuran las métricas que se utilizarán para medir la bondad del ajuste de dichos modelos. Por último, en el capítulo 5: Resultados obtenidos se han implementado y evaluado los modelos ya mencionados.

En cuanto a la planificación, el calendario inicial propuesto se encuentra en la sección 1.5; no ha sido posible adherirse por completo a esta planificación inicial en cuanto a tiempos estimados por las razones que se presentan a continuación, pero sí se han cumplido los plazos establecidos de las entregas parciales. Respecto a las diferencias entre tiempo estimado y tiempo real dedicado, se puede entender que sus causas son:

- **Problemas de configuración**: durante la fase de implementación del proyecto, se proporcionó desde la universidad una máquina Linux de Azure para realizar el entrenamiento de los mode-

los. No obstante, la configuración del proyecto estaba pensada para un entorno Windows, lo que produjo complicaciones a la hora de configurarla y, finalmente, se decidió abandonar la idea de utilizarla debido a incompatibilidades. Este problema supuso aproximadamente 50 horas adicionales no estimadas debido a la lentitud de la máquina y los reintentos.

- **Tamaño de los *datasets***: el conjunto de datos ZeroWaste y sobre todo ResortIT están comprendidos de un gran número de imágenes y un aún mayor número de anotaciones sobre las que entrenar. Aunque este tamaño es positivo para mejorar el rendimiento y la bondad del ajuste de los modelos, también supone un sobrecoste en el tiempo de entrenamiento.
- **Entrenamiento de los modelos**: los modelos escogidos, especialmente *Faster* R-CNN, cuentan con un número de parámetros entrenables importante, lo que implica la necesidad de realizar más *epochs* de entrenamiento. También son pesados, razón por la que el proceso de entrenamiento se ve ralentizado.
- **Lentitud en las pruebas**: es un problema derivado de los dos puntos anteriores; como los modelos son pesados y el conjunto de datos es también muy grande, llevar a cabo las pruebas correspondientes para realizar tareas de *debug* es muy lento –del orden de horas para comprobar si la tendencia inicial en un entrenamiento era correcta.

6.3. Análisis de los impactos previstos e imprevistos

En esta sección se evalúa el impacto del proyecto en materia de sostenibilidad, ético-social y diversidad que se presenta en la sección 1.3: Impacto en sostenibilidad, ético-social y diversidad, y se analiza si han aparecido nuevos impactos no previstos y su efecto.

Respecto a los impactos previstos en la sección 1.3, en primer lugar es necesario destacar que la mayoría están íntimamente relacionados con la puesta en marcha del proyecto en una planta de selección y reciclaje, y por tanto no se puede evaluar todavía el impacto del proyecto en estas materias.

No obstante, el impacto negativo en sostenibilidad y la huella ecológica negativa se han mitigado parcialmente gracias a una búsqueda de hiperparámetros optimizada mediante el *tuner* de la librería Pytorch Lightning –la idea original para realizar esta fase era utilizar una búsqueda mediante *grid search* o algoritmos genéticos.

En cuanto a impactos no previstos, cabe destacar que la creación y publicación gratuita del *framework* de detección de residuos escrito y documentado en inglés, el lenguaje universal en este tipo de proyectos, supone un impacto positivo en las dimensiones de diversidad y ético-social, ya que permite que cualquier persona, sin importar su género, etnia, religión o cualquier otro atributo diferenciador, pueda acceder al *framework*, utilizarlo y expandirlo.

6.4. Líneas de trabajo futuras

En el transcurso del proyecto no ha sido posible abarcar todas las líneas de investigación que se plantean en la introducción por limitaciones principalmente temporales, aunque también existen líneas de otras áreas que no se han podido abarcar por falta de cualificación. Por tanto, los siguientes puntos quedan pendientes de investigación para la mejora del *framework* tanto en calidad del trabajo existente como para ampliar sus funcionalidades:

- **Desarrollo de un sistema robótico para la segregación de los residuos detectados**: esta línea de investigación es el proyecto complementario a este trabajo y el que supondrá para el mismo el salto del plano teórico al práctico.
- **Inclusión de nuevas categorías de residuos**: con esta línea de investigación se pretende aumentar el alcance del proyecto; cuantas más categorías de residuos se puedan detectar con el *framework*, más útil será.
- **Análisis práctico de detectores de objetos de dos pasadas**: en este trabajo se ha seleccionado *Faster* R-CNN como detector de dos pasadas a evaluar por ser el más popular y el que mejores resultados produce en la bibliografía reciente; no obstante, una comparativa de detectores de dos pasadas proporcionaría más fundamento a la decisión tomada o un detector de residuos de dos pasadas más preciso y fiable.
- **Análisis práctico de detectores de objetos de una pasada**: en este proyecto se ha seleccionado SSD como detector de una pasada a evaluar por ser de los más populares y potentes en su categoría. Sin embargo, otros detectores de una pasada como YOLO también gozan de una gran popularidad y rendimiento, y una comparativa más exhaustiva de los detectores de una pasada podría proporcionar un detector de residuos de una pasada más preciso y fiable.
- **Investigación en los modelos híbridos (DL+ML tradicional) de la detección de residuos**: este es un campo poco explorado en el área, y es posible que un acercamiento al problema con modelos mixtos produzca detectores más precisos y fiables. Adicionalmente, cabe destacar la investigación en creación de cabezas regresoras (la parte del modelo que se encarga de generar las *bounding boxes*) mediante algoritmos de regresión de *Machine Learning* tradicional.
- **Desarrollo de un sistema de seguimiento de residuos**: esta línea de trabajo supone una optimización en materia de rendimiento y tiempo sobre el sistema de detección de residuos y está muy ligado a las características físicas (dirección y sentido de la cinta de transporte, condiciones de iluminación) de la puesta en marcha del proyecto.

Glosario

ODS	Objetivos de Desarrollo Sostenible
MSW	*Municipal Solid Waste* o residuos urbanos
RM	residuos domésticos
ICI	residuos de la industria y comercio
C&M	residuos de la construcción y demolición
API	*Application Programming Interface*. Es el conjunto de puntos de entrada que expone un programa para facilitar la comunicación con otros sistemas.
AI	*Artificial Intelligence* o Inteligencia Artificial
DL	*Deep Learning*
CV	*Computer Vision* o Visión por Computador
ML	*Machine Learning* o Aprendizaje Automático
GPU	Tarjeta de vídeo
Pipeline	Flujo de trabajo estandarizado desde el la entrada del modelo a la salida
Overfitting	Se dice que un modelo sufre de *overfitting* cuando se ajusta demasiaod a los datos de entrenamiento y pierde la capacidad de generalizar
Bounding box	Conjunto de 4 puntos en una imagen que definen el área en la que se encuentra la instancia del objeto
Sliding windows	Técnica de búsqueda exhaustiva de regiones que se basa en deslizar la *bounding box* por la imagen
Non-Maximal Suppression (NMS)	técnica de optimización de *bounding boxes* que elimina aquellas propuestas que se superponen por encima de cierto umbral
Intersection over Union (IoU)	métrica que cuantifica el grado de superposición entre dos *bounding boxes*
mean Average Precision (mAP)	métrica que cuantifica la precisión media del modelo para todas las clases en conjunto
mean Average Recall (mAR)	métrica que cuantifica el ratio de verdaderos positivos del modelo para todas las clases en conjunto
R-CNN	Ross Girshick y col. 2013
***Fast* R-CNN**	Ross Girshick 2015
***Faster* R-CNN**	Shaoqing Ren y col. 2015
***Mask* R-CNN**	Kaiming He, Georgia Gkioxari y col. 2017
SSD	*Single Shot MultiBox Detector* (Wei Liu y col. 2015)
YOLO	*You Only Look Once* (Joseph Redmon y col. 2015)

Garbage In, Garbage Out o **GIGO**	la calidad de las salidas o *output* de un modelo está directamente relacionada con la calidad de las entradas o *input*. De esta forma, si las entradas (imágenes y anotaciones en el caso de este proyecto) no tienen la calidad suficiente, el modelo tampoco devolverá predicciones de calidad
RPN	*Region Proposal Network*
RoI	*Region of Interest*

Bibliografía

Aakarsh Yelisetty (13 de jul. de 2020). “Understanding Fast R-CNN and Faster R-CNN for Object Detection. Let’s understand these state-of-the-art Region Proposal based Convolution Neural Networks in detail”. En: *Towards Data Science*. URL: https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97 (visitado 19-12-2022).

Adam Kelly (2015). *Cigarette Butt Dataset and Trained Weights*. URL: https://www.immersivelimit.com/datasets/cigarette-butts.

Adrian Rosebrock (7 de nov. de 2016). *Intersection over Union (IoU) for object detection*. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

Albumentations Docs (2019). *Bounding boxes augmentation for object detection*. URL: https://albumentations.ai/docs/getting_started/bounding_boxes_augmentation/.

Anaconda, Inc. (2017). *Conda. Package, dependency and environment management for any language—Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, Fortran, and more.* URL: https://docs.conda.io/en/latest/index.html (visitado 20-12-2022).

Arize (s.f.). *Confusion Matrix*. URL: https://arize.com/glossary/confusion-matrix/.

Arkadiy Serezhkin (2021). *Drinking Waste Classification*. URL: https://www.kaggle.com/datasets/arkadiyhacks/drinking-waste-classification.

Daniel Bolya y col. (2019). *YOLACT: Real-time Instance Segmentation*. eprint: arXiv:1904.02689.

Daniel Octavian Melinte, Ana-Maria Travediu y Dan N. Dumitriu (2020). “Deep Convolutional Neural Networks Object Detector for Real-Time Waste Identification”. En: *Applied Sciences* 10.20. ISSN: 2076-3417. DOI: 10.3390/app10207301.

Di Tian y col. (jul. de 2022). “Review of object instance segmentation based on deep learning”. En: *Journal of electronic imaging* 31.4. ISSN: 1017-9909. DOI: 10.1117/1.JEI.31.4.041205.

Dina Bashkirova y col. (2021). *ZeroWaste Dataset: Towards Deformable Object Segmentation in Cluttered Scenes*. eprint: arXiv:2106.02740.

Ecoembes (2022). *Ecoembes: Reduce, Reutiliza y Recicla*. URL: https://www.ecoembes.com/es.

Ecovidrio (2022). *Reciclaje de envases de vidrio en España | Ecovidrio*. URL: https://www.ecovidrio.es/.

Enze Xie y col. (2019). *PolarMask: Single Shot Instance Segmentation with Polar Representation*. eprint: arXiv:1909.13226.

Eurostat (sep. de 2020). *Municipal waste by waste management operations*. URL: https://ec.europa.eu/eurostat/databrowser/view/env_wasmun/default.

FCC Medio Ambiente (2022). *FCC Medio Ambiente*. URL: https://www.fccma.com/.

Google (2017). *TensorFlow Lite: ML for Mobile and Edge*. URL: https://www.tensorflow.org/lite.

Ilija Mihajlovic (25 de abr. de 2019). "Everything You Ever Wanted To Know About Computer Vision. Here's A Look Why It's So Awesome." En: *Towards Data Science*. URL: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e (visitado 17-10-2022).

Jan Hosang, Rodrigo Benenson y Bernt Schiele (2017). *Learning non-maximum suppression*. DOI: 10.48550/ARXIV.1705.02950. URL: https://arxiv.org/abs/1705.02950.

Jason Calaiaro (jul. de 2022). "AI-guided robots are ready to sort your recyclables". En: *IEEE Spectrum*. URL: https://spectrum.ieee.org/ai-guided-robots-are-ready-to-sort-your-recyclables.

Jiahui Yu y col. (oct. de 2016). "UnitBox: An Advanced Object Detection Network". En: *Proceedings of the 24th ACM international conference on Multimedia*. ACM. DOI: 10.1145/2964284.2967274. URL: https://doi.org/10.1145%2F2964284.2967274.

Jiewen Feng y col. (2021). "Garbage Disposal of Complex Background Based on Deep Learning With Limited Hardware Resources". En: *IEEE Sensors Journal* 21.18, págs. 21050-21058. DOI: 10.1109/JSEN.2021.3100636.

Jordi Gironés Roig y col. (2017). *Minería de datos: modelos y algoritmos*. ISBN: 978-84-9116-904-8.

Joseph Redmon y col. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. eprint: arXiv:1506.02640.

Kaiming He, Georgia Gkioxari y col. (2017). *Mask R-CNN*. eprint: arXiv:1703.06870.

Kaiming He, Xiangyu Zhang y col. (2014). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". En: DOI: 10.1007/978-3-319-10578-9_23. eprint: arXiv:1406.4729.

Li Liu y col. (2018). *Deep Learning for Generic Object Detection: A Survey*. DOI: 10.48550/ARXIV.1809.02165. URL: https://arxiv.org/abs/1809.02165.

Marek Kraft y col. (2021). "Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle". En: *Remote Sensing* 13.5. ISSN: 2072-4292. DOI: 10.3390/rs13050965.

Maria Koskinopoulou y col. (2021). "Robotic Waste Sorting Technology: Toward a Vision-Based Categorization System for the Industrial Robotic Separation of Recyclable Waste". En: *IEEE Robotics and Automation Magazine* 28.2, págs. 50-60. DOI: 10.1109/MRA.2021.3066040.

Mark Everingham y col. (jun. de 2010). "The Pascal Visual Object Classes (VOC) challenge". En: *International Journal of Computer Vision* 88, págs. 303-338. DOI: 10.1007/s11263-009-0275-4.

Microsoft (2018). *ONNX Runtime: Optimize and Accelerate Machine Learning Inferencing and Training*. URL: https://onnxruntime.ai.

Ministerio para la transición ecológica y el reto demográfico (2022). *Otras fracciones: fracción resto*. URL: https://www.miteco.gob.es/es/calidad-y-evaluacion-ambiental/temas/prevencion-y-gestion-residuos/flujos/domesticos/fracciones/otros/.

Naciones Unidas (oct. de 2022a). *¿Qué es el cambio climático?* URL: https://www.un.org/es/climatechange/what-is-climate-change.

— (oct. de 2022b). *Causas y efectos del cambio climático*. URL: https://www.un.org/es/climatechange/science/causes-effects-climate-change.

— (oct. de 2022c). *Objetivos y metas de desarrollo sostenible*. URL: https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/.

Niall O' Mahony y col. (oct. de 2019). "Deep Learning vs. Traditional Computer Vision". En: *Advances in Computer Vision*. DOI: 10.1007/978-3-030-17795-9.

Nikhil Venkat Kumsetty y col. (2022). "TrashBox: Trash Detection and Classification using Quantum Transfer Learning". En: *2022 31st Conference of Open Innovations Association (FRUCT)*, págs. 1-6.

Nobuyoshi Yabuki, Naoto Nishimura y Tomohiro Fukuda (2018). "Automatic Object Detection from Digital Images by Deep Learning with Transfer Learning". En: *Advanced Computing Strategies for Engineering*. Springer International Publishing, págs. 3-15.

Oluwasanya Awe y Robel Mengistu (2017). "Final Report : Smart Trash Net : Waste Localization and Classification". En:

Patricia Lázaro Tello (2022). *Waste Detection System*. URL: https://plazarotello.github.io/waste-detection-system-docs/ (visitado 20-12-2022).

Pedro F. Proença y Pedro Simões (2020). "TACO: Trash Annotations in Context for Litter Detection". En: *ArXiv* abs/2003.06975.

Pete Chapman y col. (2000). *CRISP-DM 1.0 Step-by-step data mining guide*. The CRISP-DM consortium.

Piotr T. Nowakowski y Teresa Pamuła (2020). "Application of deep learning object classifier to improve e-waste collection planning." En: *Waste management* 109, págs. 1-9.

Remi Cuingnet y col. (2022). "PortiK: A computer vision based solution for real-time automatic solid waste characterization – Application to an aluminium stream". En: *Waste Management* 150, págs. 267-279. ISSN: 0956-053X. DOI: https://doi.org/10.1016/j.wasman.2022.05.021.

Richmond Alake (23 de sep. de 2020). "A Beginner's Guide To Computer Vision. A friendly introduction to one of the important fields within Artificial Intelligence". En: *Towards Data Science*. URL: https://towardsdatascience.com/a-beginners-guide-to-computer-vision-dca81b0e94b4 (visitado 16-10-2022).

Ross Girshick (2015). *Fast R-CNN*. eprint: arXiv:1504.08083.

Ross Girshick y col. (2013). *Rich feature hierarchies for accurate object detection and semantic segmentation*. eprint: arXiv:1311.2524.

Saica (2022). *Saica Natur: El valor del "Zero"*. URL: https://www.saica.com/es/saica-natur/.

Sambasivarao. K (22 de abr. de 2019). "Region of Interest Pooling. The technique which made object detection faster and viable". En: *Towards Data Science*. URL: https://towardsdatascience.com/region-of-interest-pooling-f7c637f409af (visitado 19-12-2022).

Sarah Frost y col. (2019). "CompostNet: An Image Classifier for Meal Waste". En: *2019 IEEE Global Humanitarian Technology Conference (GHTC)*, págs. 1-4.

Sashaank Sekar (2021). *Waste Classification data*. URL: https://www.kaggle.com/datasets/techsash/waste-classification-data.

Seán Lynch (2017). "OpenLitterMap.com – Open Data on Plastic Pollution with Blockchain Rewards (Littercoin)". En: DOI: https://doi.org/10.1186/s40965-018-0050-y.

Seunguk Na y col. (2022). "Development of an Artificial Intelligence Model to Recognise Construction Waste by Applying Image Data Augmentation and Transfer Learning". En: *Buildings* 12.2. ISSN: 2075-5309. DOI: 10.3390/buildings12020175. URL: https://www.mdpi.com/2075-5309/12/2/175.

Shaoqing Ren y col. (2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. eprint: arXiv:1506.01497.

Sinno Jialin Pan y Qiang Yang (2010). “A Survey on Transfer Learning”. En: *IEEE Transactions on Knowledge and Data Engineering* 22.10, págs. 1345-1359. DOI: 10.1109/TKDE.2009.191.

Sobhan Shukueian (14 de jun. de 2021). “How Single Shot MultiBox Detector (SSD) Real-Time Object Detection Technique works?” En: *Medium*. URL: https://shukueian.medium.com/how-single-shot-multibox-detector-ssd-real-time-object-detection-technique-works-47e87bee2562 (visitado 19-12-2022).

Spencer Ploeger, Matthew Bolan y Lucas Dasovic (2022). “Automated Identification of Used Beverage Cans for Deposit Return using Deep Learning Methods”. En: *2022 IEEE Conference on Technologies for Sustainability (SusTech)*, págs. 165-172. DOI: 10.1109/SusTech53338.2022.9794235.

Takuya Kiyokawa y col. (2021). “Robotic Waste Sorter With Agile Manipulation and Quickly Trainable Detector”. En: *IEEE Access* 9, págs. 124616-124631. DOI: 10.1109/ACCESS.2021.3110795.

Tao Wang y col. (2020). “A Multi-Level Approach to Waste Object Segmentation”. En: DOI: 10.3390/s20143816. eprint: arXiv:2007.04259.

Tsung-Yi Lin, Michael Maire y col. (2014). *Microsoft COCO: Common Objects in Context*. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312.

Tsung-Yi Lin, Piotr Dollár y col. (2016). *Feature Pyramid Networks for Object Detection*. eprint: arXiv:1612.03144.

Tsung-Yi Lin, Priya Goyal y col. (2017). *Focal Loss for Dense Object Detection*. eprint: arXiv:1708.02002.

URBASER (2022). *URBASER | Compañía global de Gestión Medioambiental*. URL: https://www.urbaser.com/.

Victor Roman (27 de mar. de 2020). “CNN Transfer Learning and Fine Tuning”. En: *Towards Data Science*. URL: https://towardsdatascience.com/cnn-transfer-learning-fine-tuning-9f3e7c5806b2 (visitado 19-10-2022).

Wang Hechun y Zheng Xiaohong (2019). “Survey of Deep Learning Based Object Detection”. En: *Proceedings of 2019 2nd Internation Conference on Big Data Technologies (ICBDT 2019)*, págs. 149-153. ISBN: 978-1-4503-7192-6. DOI: 10.1145/3358528.3358574.

Wei Liu y col. (2015). “SSD: Single Shot MultiBox Detector”. En: DOI: 10.1007/978-3-319-46448-0_2. eprint: arXiv:1512.02325.

Wei-Lung Mao y col. (2022). “Deep learning networks for real-time regional domestic waste detection”. En: *Journal of Cleaner Production* 344, pág. 131096. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2022.131096.

Weisheng Lu y Junjie Chen (feb. de 2022). “Computer vision for solid waste sorting: A critical review of academic research”. En: *Waste Management* 142, págs. 29-43. ISSN: 0956-053X. DOI: https://doi.org/10.1016/j.wasman.2022.02.009.

Xinlong Wang y col. (2019). *SOLO: Segmenting Objects by Locations*. eprint: arXiv:1912.04488.

Zhi Tian y col. (2019). *FCOS: Fully Convolutional One-Stage Object Detection*. eprint: arXiv:1904.01355.

Zhifei Zhang y col. (2019). “Industrial Robot Sorting System for Municipal Solid Waste”. En: *Intelligent Robotics and Applications*. Springer International Publishing, págs. 342-353. ISBN: 978-3-030-27532-7.

Zhong-Qiu Zhao y col. (2019). “Object Detection With Deep Learning: A Review”. En: *IEEE Transactions on Neural Networks and Learning Systems* 30.11, págs. 3212-3232. DOI: 10.1109/TNNLS.2018.2876865.

Apéndice A

Waste Detection System

El presente proyecto ha resultado en la elaboración de un paquete de Python y un conjunto de *notebooks* de Jupyter para el entrenamiento y utilización de las arquitecturas y modelos propuestos durante este documento. Este conjunto se ha denominado ***Waste Detection System*** (Patricia Lázaro Tello 2022).

A.1. Requisitos de sistema

La instalación y puesta en marcha del paquete requiere de la previa instalación del entorno **conda** (Anaconda, Inc. 2017). Se ha de tener acceso al comando `conda` a través del *Powershell* en Windows o la línea de comandos en Unix.

La instalación de dependencias y la correcta ejecución de los *scripts* y módulos incluidos en el paquete ha sido probada únicamente en Windows 10; por tanto, no se puede asegurar la correcta instalación y/o ejecución del paquete en otras versiones de Windows o en el ecosistema Unix.

A.2. Guía de instalación

Los siguientes pasos guían al usuario en la instalación de las dependencias y entorno requeridos por el paquete *Waste Detection System*. Al final de esta guía, el usuario contará con un nuevo entorno de conda bajo el nombre de `waste-detector` en el que estarán instaladas las dependencias necesarias para ejecutar los *scripts* del paquete.

1. Descargar el repositorio del enlace: github.com/plazarotello/waste-detection-system
2. Abrir un *Powershell* o línea de comandos *bash* en la raíz del repositorio
3. Ejecutar el *script* de instalación de dependencias
    3.1. Para Windows: ejecutar el comando `.\scripts\setup.ps1`
    3.2. Para Unix: ejecutar el comando `. ./scripts/setup.sh`

A.3. Cómo utilizar el paquete *Waste Detection System*

El paquete de Python ***Waste Detection System*** puede ser utilizado tanto en *scripts* de Python como en *notebooks* de Jupyter. El uso previsto del paquete incluye la ejecución de las funciones de entrenamiento, *test*, búsqueda de hiperparámetros para los modelos y la carga/guardado de pesos.

La ejecución de estas funciones se encuentra canalizada a través del punto de entrada o *entry point* que se encuentra en el *script* `main.py`. A continuación se muestran ejemplos de las funciones anteriormente mencionadas, disponibles en el *script* `sample.py`:

***Script* A.1:** Ejemplo de uso de *Waste Detection System* (`sample.py`)

```python
import os
import pandas as pd
# ------------------------------------------------------------------------------
# MSW Detector packages
from waste_detection_system import shared_data as base, main
# ==============================================================================
os.environ["PYTHONUNBUFFERED"] = '1'
# ==============================================================================

# 1) Load dataset
with open(base.FINAL_DATA_CSV, 'r', encoding='utf-8-sig') as final_file:
    final_dataset = pd.read_csv(final_file)

# ------------------------------------------------------------------------------
# 1.1) Preprocess ZeroWaste dataset and extract samples
zerowaste = final_dataset[final_dataset['dataset'] == 'final']
zerowaste_train = zerowaste[zerowaste['type'] == 'train']
zerowaste_val = zerowaste[zerowaste['type'] == 'val']
zerowaste_test = zerowaste[zerowaste['type'] == 'test']

zerowaste_train_sample = zerowaste_train[zerowaste_train['path'].isin(
    zerowaste_train[['path']].sample(frac=0.3).path.tolist())]
zerowaste_val_sample = zerowaste_val[zerowaste_val['path'].isin(
    zerowaste_val[['path']].sample(frac=0.3).path.tolist())]
zerowaste_test_sample = zerowaste_test[zerowaste_test['path'].isin(
    zerowaste_test[['path']].sample(frac=0.3).path.tolist())]

# ------------------------------------------------------------------------------
# 1.2) Preprocess ResortIT dataset and extract samples
resortit = final_dataset[final_dataset['dataset'] == 'complementary']
resortit_train = resortit[resortit['type'] == 'train']
resortit_val = resortit[resortit['type'] == 'val']
resortit_test = resortit[resortit['type'] == 'test']

resortit_train_sample = resortit_train[resortit_train['path'].isin(
    resortit_train[['path']].sample(frac=0.3).path.tolist())]
resortit_val_sample = resortit_val[resortit_val['path'].isin(
    resortit_val[['path']].sample(frac=0.3).path.tolist())]
```

```
39
40 # ------------------------------------------------------------------------
41 # 2) Creates and prints the structure and trainable layers of some models
42 # This is an optional step, but very useful when debugging model architectures
43 print('FASTER R-CNN / TLL=3')
44 main.models.pretty_summary(main.models.get_fasterrcnn(2, 3))
45 print('SSD /TLL=1')
46 main.models.pretty_summary(main.models.get_ssd(2, 1))
47
48 # ------------------------------------------------------------------------
49 # 3) An example of hyperparameter search
50 print('HYPERPARAMETER SEARCH')
51 main.hyperparameter_search(
52     name='ssd-zerowaste-hyper',
53     config=base.MODELS_DIR/'hyper-options.json',
54     dataset=zerowaste_train_sample,
55     selected_model=main.models.AVAILABLE_MODELS.SSD,
56     num_classes=2,
57     tll=3,
58     weights=None,
59     find_batch_size = False,
60     metric='Validation_mAP'
61 )
62
63 # ------------------------------------------------------------------------
64 # 4) An example of train
65 print('TRAIN')
66 main.train(
67     train_dataset=resortit_train_sample,
68     val_dataset=resortit_val_sample,
69     name='fasterrcnn-resortit-sample',
70     config=base.MODELS_DIR/'faster-r-cnn-sample.json',
71     num_classes=2,
72     tll=1,
73     resortit_zw=0,
74     selected_model=main.models.AVAILABLE_MODELS.SSD,
75     limit_validation=0.1,
76     metric='training_loss'
77 )
78
79 # ------------------------------------------------------------------------
80 # 5) This is how to load the best model checkpoint
81 best_model_path_tll0 = base.MODELS_DIR / 'fasterrcnn-zerowaste' / 'tll0_3.ckpt'
82
83 # ------------------------------------------------------------------------
84 # 6.1) In the test phase we run the model through the dataset in inference mode
85 #      and obtain the test mAP
86 print('TEST')
87 main.test(
88     best_model_path_tll0,
89     selected_model=main.models.AVAILABLE_MODELS.FASTERRCNN,
90     resortit_zw=1,
```

```
    test_dataset=zerowaste_test_sample
)

# ------------------------------------------------------------------------
# 6.2) In the benchmark phase we run the model through the dataset in inference
#      mode and obtain the average milliseconds of processing time for one image
#      and print the differences between the real annotations and the ones
#      obtained by the model
print('BENCHMARK')
average_ms = main.benchmark(
    best_model_path_tl10,
    test_dataset=zerowaste_test_sample
)

# ------------------------------------------------------------------------
# 6.3) Create the ONNX model and run the benchmark inference
best_model_path_tl10_onnx = main.optimize_model_for_inference(
    best_model_path_tl10,
    zerowaste_train_sample
)
main.benchmark_optimized(
    best_model_path_tl10_onnx,
    test_dataset=zerowaste_test_sample
)
```

Este *script* de ejemplo no es exhaustivo y representa únicamente las funciones principales del paquete ***Waste Detection System***. En el siguiente link se encuentra la documentación completa del paquete: plazarotello.github.io/waste-detection-system-docs.

Apéndice B

Bounding boxes: formatos de anotaciones

La detección de objetos mediante *Deep Learning* es un campo relativamente nuevo en el que los estándares de facto en multitud de facetas todavía no están asentados. En el caso de las anotaciones en forma de *bounding boxes*, han surgido diferentes formatos de representación asociados a los *datasets* y modelos de detección más populares. En este anexo se describen los formatos de *bounding boxes* y anotaciones más comunes, así como una pequeña comparativa en la figura B.1.

B.1. Formato Pascal VOC

El formato **Pascal VOC** (Mark Everingham y col. 2010) es un formato de anotaciones basado en ficheros XML, donde la *bounding box* está definida por las coordenadas XY de la esquina superior izquierda y de la esquina inferior derecha, sin normalizar. A este formato se le conoce también como **formato XYXY**.

B.2. Formato COCO

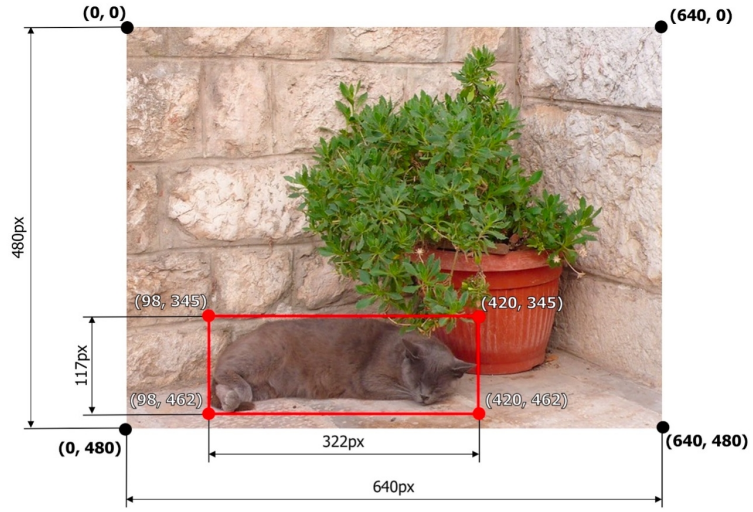
El formato **COCO** (Tsung-Yi Lin, Michael Maire y col. 2014) es un formato de anotaciones basado en ficheros JSON, donde la *bounding box* está definida por las coordenadas XY de la esquina superior izquierda, la anchura y altura de la caja sin normalizar. También se conoce este formato como **formato XYWH**.

B.3. Formato YOLO

El formato **YOLO** (Joseph Redmon y col. 2015) es un formato de anotaciones basado en ficheros TXT, donde la *bounding box* está definida por las coordenadas XY del centro de la caja, la anchura y altura de la misma, normalizadas.

B.4. Formato Albumentations

El formato **albumentations** (Albumentations Docs 2019) es un formato de *bounding boxes* –esto quiere decir que es un formato que sólo hace referencia a la forma de definir las *bounding boxes*, sin referirse a cómo almacenar la información en disco. Es similar al formato Pascal VOC dado que utiliza también las coordenadas XY de las esquinas superior izquierda e inferior derecha, pero su diferencia reside en que estas coordenadas están normalizadas.



pascal_voc
[x_min, y_min, x_max, y_max] ⟶ [98, 345, 420, 462]

albumentations
normalized [x_min, y_min, x_max, y_max] ⟶ [0.153125, 0.71875, 0.65625, 0.9625]

coco
[x_min, y_min, width, height] ⟶ [98, 345, 322, 117]

yolo
normalized [x_center, y_center, width, height] ⟶ [0.4046875, 0.8614583, 0.503125, 0.24375]

Figura B.1: Una misma *bounding box* en los formatos de anotaciones más populares.
Fuente: Albumentations Docs 2019

Apéndice C

Hiperparámetros de los modelos

Dataset	Modelo	TLL	*Batch Size*	*Learning Rate*	*Epochs*
ResortIT	*Faster* R-CNN	TLL=1	64	5.89e-05	25
ResortIT	*Faster* R-CNN	TLL=2	32	4.9e-06	50
ResortIT	*Faster* R-CNN	TLL=3	32	9.33e-07	30
ResortIT	*Faster* R-CNN	TLL=4	32	9.33e-07	30
ResortIT	*Faster* R-CNN	TLL=5	N/A	N/A	N/A
ResortIT	*Faster* R-CNN	TLL=0	8	4.07e-07	30
ZeroWaste	*Faster* R-CNN	TLL=1	32	6.31e-08	25
ZeroWaste	*Faster* R-CNN	TLL=2	32	6.31e-08	50
ZeroWaste	*Faster* R-CNN	TLL=3	32	6.31e-08	50
ZeroWaste	*Faster* R-CNN	TLL=4	32	6.31e-08	100
ZeroWaste	*Faster* R-CNN	TLL=5	N/A	N/A	N/A
ZeroWaste	*Faster* R-CNN	TLL=0	8	6.31e-08	100

Cuadro C.1: Elección de los hiperparámetros (*batch size*, *learning rate* y *epochs*) del modelo *Faster* R-CNN, cuyo entrenamiento se ejecutó en su totalidad en *Shadow* PC.

Dataset	Modelo	TLL	*Batch Size*	*Learning Rate*	*Epochs*
ResortIT	SSD	TLL=1	128	0.001	100
ResortIT	SSD	TLL=2	128	4.07e-07	200
ResortIT	SSD	TLL=3	128	4.07e-07	50
ResortIT	SSD	TLL=4	128	4.07e-07	100
ResortIT	SSD	TLL=5	128	4.07e-07	100
ResortIT	SSD	TLL=0	64	4.07e-07	200
ZeroWaste	SSD	TLL=1	128	3.98e-06	25
ZeroWaste	SSD	TLL=2	128	3.31e-07	50
ZeroWaste	SSD	TLL=3*	128	3.31e-07	25
ZeroWaste	SSD	TLL=4*	128	3.31e-07	75
ZeroWaste	SSD	TLL=5*	128	3.31e-07	100
ZeroWaste	SSD	TLL=0*	8	4.17e-08	100
				1.20e-08	100

Cuadro C.2: Elección de los hiperparámetros (*batch size*, *learning rate* y *epochs*) del modelo SSD, cuyo entrenamiento se ejecutó principalmente en el PC personal. *: hiperparámetros seleccionados en *Shadow* PC.