UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Reconocimiento óptico de envases mediante redes neuronales en un contexto de economía circular

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Cabaña Tissot, Marcos

Tutor/a: Rodas Jordá, Ángel

CURSO ACADÉMICO: 2022/2023

Reconocimiento óptico de envases mediante redes neuronales en un contexto de economía circular.

Trabajo de Fin de Grado

Curso 2022-2023

GIEIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Alumno: Marcos Cabaña Tissot

Tutor: Ángel Rodas Jordá

Agradecimientos:

Quisiera agradecer a mis padres por enseñarme
que hay que seguir aunque las cosas se pongan difíciles
y que la motivación nace de uno mismo.

Agradecer también a mi tutor Ángel,
por dejarme libertad para hacer realidad mi
visión y guiarme en el camino.

Finalmente a mis amigos y familia,
que me han soportado mis ataques de estrés y me han
ayudado con alguna que otra foto para mi base de datos.

Resumen:

El presente trabajo final de grado plantea el uso de técnicas de reconocimiento de imagen sobre diferentes tipos de envases (vidrio, plástico, aluminio, ...) para facilitar su posterior reciclado y/o reutilización, permitiendo así una correcta transición a un modelo de economía circular según las directrices de la ONU para un desarrollo sostenible.

Tras un estudio del contexto actual en materia de reciclado, analizando las posibles aplicaciones y enfoques de las técnicas propuestas en diferentes ámbitos de este sector, el TFG propone el uso de redes neuronales para la discriminación entre los distintos tipos de envases. Para ello se compararán distintas arquitecturas a fin de decidir el modelo que mejor se adapte a las necesidades planteadas.

Palabras clave: Redes neuronales; Visión por computador; Economía circular; Reciclado ; Reutilización; Envases

Resum:

El present treball final de grau planteja l'ús de tècniques de reconeixement dímatge sobre diferents tipus dénvasos (vidre, plàstic, alumini, . . .) per a facilitar el seu posterior reciclatge i/o reutilització, permetent així una correcta transició a un model déconomia circular segons les directrius de lÓNU per a un desenvolupament sostenible.

Després dún estudi del context actual en matèria de reciclatge, analitzant les possibles aplicacions i enfocaments de les tècniques proposades en diferents àmbits dáquest sector, el TFG proposa l'ús de xarxes neuronals per a la discriminació entre els diferents tipus dénvasos. Per a això es compararan diferents arquitectures a fi de decidir el model que millor sádapte a les necessitats plantejades.

Palabras clave: Redes neuronales; Visión por computador; Economía circular; Reciclado ; Reutilización; Envases

Abstract:

The present End-of-Degree Thesis proposes the use of image recognition techniques upon different types of waste containers (glass, plastic, aluminum, ...) to facilitate its later recycling and/or repurpose, allowing in this way a proper transition to a circular economy by following the guidelines proposed by the UN for a more sustainable development.

After studying the current context in terms of recycling, by analysing possible applications and focuses of the techniques proposed on different areas of this sector, the present project proposes the use of neural networks to differenciate between different types of waste containers. For that, some architectures will be tested in order to determine which model adjusts better to the requirements.

Keywords: Neural networks; Computer vision; Circular economy; Recycling; Repurpose; Waste containers

Índice General:

Parte I

Memoria:

Índice:

Índice de figuras

Índice de cuadros

1. Introducción

En la actualidad, la necesidad de frenar el **cambio climático** está dando paso a un cambio de paradigma con respecto a cómo se deben de gestionar los agentes contaminantes que afectan al medio ambiente. La gestión de residuos como la basura es un problema grave, en especial los envases.

Un dato alarmante es que el 90 % de todos los envases de plástico de un solo uso generados nunca han sido reciclados [1]. Esto se debe en parte a sistemas de reciclado poco controlados o mal enfocados en un sistema de economía predominantemente lineal, donde solo una pequeña parte de los residuos se vuelve a poner en circulación.

1.1. Contexto

La Organización de las Naciones Unidas (ONU) ha propuesto una serie de metas a conseguir de cara al final de esta década con el fin de lograr un desarrollo sostenible y minimizar los efectos del cambio climático, entre otras. En particular, hay tres de estos objetivos fuertemente relacionados con la gestión de los envases, cada uno con una serie de indicadores que siguen el progreso de cada país en esta lucha por un mundo más sostenible. Estos objetivos se llaman: *Producción y Consumo Responsable*, *Acción por el Clima* y *Vida Submarina* [2]. Tras un seguimiento de estos y sus respectivos indicadores, la OCDE estató en 2021 que España ha mejorado la eficiencia del uso de los recursos pero esta no es suficiente para cumplir con los objetivos planteados por la ONU para 2030. [3].

Sin embargo, otros países de la Unión Europea como son Dinamarca y Suecia, entre otros, han decido apoyar la integración de sistemas alternativos que invitan a sus ciudadanos a involucrase activamente en el reciclaje de envases cotidianos. Esto se ha conseguido incentivando a la ciudadanía mediante recompensas inmediatas después de depositar estos envases en unas máquinas conocidas comúnmente como *Reverse Vending*. Estas máquinas aceptan envases qué detectan mediante sensores y a cambio devuelven una pequeña cantidad de dinero por cada envase a la persona que los ha depositado. **Dansk Retursystem** [4] es un ejemplo de estas iniciativas impulsadas por los gobiernos que aplican suplementos al precio de los productos envasados que puede ser recuperado al devolverlos a estos puntos, normalmente localizados en los supermercados que los venden. De esta manera, el gobierno Danés ha conseguido reciclar un 92 % de todas las botellas y latas en circulación desde que se implantó este sistema.

1.2. Motivación

Como se ha comentado antes, en los últimos años están surgiendo nuevos paradigmas en cuanto a la gestión de los residuos. Sin embargo, muchos de estos se encuentran limitados por no hacer uso de las últimas técnicas de reconocimiento de imagen.

Las conocidas máquinas de *Reverse Vending*, por ejemplo, se encuentran enormemente limitadas en su tarea de conseguir reciclar más envases, de acuerdo a que solo es capaz de reconocer envases que se encuentren en perfectas condiciones. Muchos de ellos se limitan a escanear códigos presentes en las etiquetas de los envases, los cuales en caso de no tener la etiqueta o que esta no sea legible, no son aceptados por la máquina y contradicen el propósito de implantar una de estas máquinas en primer lugar. Al integrar técnicas de reconocimiento de imágenes, todos estas limitaciones se reducen a mínimos, siendo capaces de reconocer los envases no por códigos impresos en las etiquetas, sino mediante **neuronas artificiales** que encuentran patrones en las imágenes de estos envases, siendo capaz de reconocerlos incluso si se encuentran en malas condiciones.

En los últimos 10 años, los avances en cuanto arquitecturas de **redes neuronales** y sus posibles aplicaciones se han producido de forma exponencial. Es momento de aprovechar estos avances y aplicarlos en los sectores que tan desesperadamente necesitan una mejora como son los que ayudan a frenar el cambio climático y la contaminación del planeta.

1.3. Objetivos:

El presente Trabajo de Fin de Grado tiene como objetivo principal utilizar redes neuronales para la detección de diferentes tipos de envases.

A raíz de este objetivo principal, se pueden extraer los siguientes objetivos secundarios:

- Estudiar diferentes arquitecturas de redes neuronales a fin de elegir una sobre la que entrenar.
- Revisar la documentación disponible de la red elegida para llevar a cabo su entrenamiento.
- Valorar posibles aplicaciones de este modelo en los sistemas de reciclado y reutilización de envases actuales y aportar una posible mejora a un sistema existente de *Reverse Vending*.
- Elaboración de una base de datos de imágenes de envases propia con sus correspondientes anotaciones.

2. Estado del arte

2.1. Conceptos

En esta porción se van a introducir conceptos básicos que servirán de introducción a las redes neuronales y facilitarán la comprensión de los siguientes apartados de este TFG

2.1.1. Machine Learning y Deep Learning

El Aprendizaje Automático o *Machine Learning* (ML) es un área de estudio contenida dentro de la Inteligencia Artificial (IA) que es comúnmente definida como la capacidad de una máquina para imitar aptitudes de la inteligencia humana. Ya en 1950, Arthur Samuel quien fue un pionero en materia de IA, describió el ML como: "el área de estudio que brinda la oportunidad de aprender a los ordenadores sin ser explícitamente programados"[5].

Dentro del *Machine Learning* encontramos el aprendizaje supervisado, el no supervisado y el reforzado. El primero haría referencia a un sistema alimentado con datos de entrenamiento previamente etiquetados. Una vez entrenado, y de haberlo hecho correctamente, el sistema debería ser capaz de reconocer patrones para identificar nuevos datos sin etiquetar. Por otro lado, un aprendizaje no supervisado sería alimentar el sistema para su entrenamiento con datos sin etiquetar. Estas técnicas no supervisadas desembocan en problemas de agrupamiento o *clustering*, por las que el sistema organiza los datos recibidos en diferentes grupos en base a patrones similares que detecta en ellos. Finalmente, el aprendizaje por refuerzo se trata de técnicas por las cuales un sistema aprende a partir de la experiencia. Es decir, si se equivoca en sus predicciones, el sistema es penalizado y si acierta, recompensado. De esta manera el sistema aprende buscando maximizar la recompensa.

El Aprendizaje Profundo o *Deep Learning* (DL) es un área dentro del ML que intenta asemejarse a la forma en la que aprenden los seres humanos. Este procesamiento de los datos se lleva a cabo concatenando un número de capas formadas a su vez por unidades de procesamiento sencillas llamadas neuronas, formando así redes neuronales artificiales.

Para entender las redes neuronales, se debe de conocer qué es una neurona artificial o perceptrón (en inglés también: TLU *Threshold Logic Unit*). Esta se compone de varias entradas asociadas a unos pesos, cuya suma ponderada se calcula para obtener una salida, como se puede ver en la Figura 1.

Las capas son los niveles en los que se organizan las neuronas artificiales dentro

de una red. Los MLP (*multilayer perceptron*) son redes compuestas por una capa de entrada, una o varias capas intermedias ocultas y una capa de salida. En caso de haber múltiples capas ocultas, las más cercanas a la capa de entrada son conocidas como capas inferiores y las más alejadas, capas exteriores.

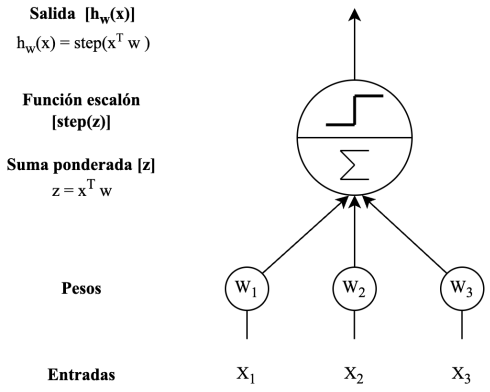


Figura 1: Estructura de una TLU

Durante el entrenamiento de una red neuronal se busca que cuando la red haya sido entrenada, esta sea capaz de generalizar. Es decir, que la red pueda aplicar los patrones aprendidos durante su entrenamiento sobre nuevos datos con los que no se ha entrenado ni ha visto nunca. Las razones por las que una red que no sea capaz de generalizar correctamente y que por lo tanto se intenta evitar, se pueden clasificar de dos maneras:

- ***Overfitting:*** Durante el proceso de entrenamiento, la red aprende con un nivel de detalle tan elevado que este afecta de manera negativa al algoritmo. El modelo detecta patrones más específicos durante su entrenamiento que no siempre se pueden generalizar y extrapolar a otros datos fuera de la muestra con la que se entrena, por lo que pierde capacidad de generalización.
- ***Underfitting:*** Estos son modelos que no son capaces de aprender correctamente los conceptos en su base de datos durante el proceso de entrenamiento, por lo que su rendimiento al enfrentarse a nuevos datos con los que no ha entrenado será bajo.

2.1.2. Detección de objetos para reconocimiento óptico

Uno de las tareas más habituales dentro de ML son los problemas de clasificación. Si enfocamos la clasificación a un ámbito visual, podemos hablar de la clasificación de imágenes. Los problemas de detección, en cambio, no solo son capaces de detectar si el objeto está o no en la imagen, sino que además son capaces de saber en qué región de la imagen se encuentran.

Para que una **red neuronal** sea capaz de localizar un objeto en una imagen, entran en juego técnicas de segmentación que sirven para separar los objetos del fondo. En aprendizajes supervisados, como el de este TFG, previamente se tiene que alimentar la red con imágenes (y la ubicación de los objetos en estas mediante cajas delimitadoras) para que aprendan durante su entrenamiento.

Cuando un ser humano mira una imagen o un vídeo es capaz de reconocer objetos de interés en cuestión de muy poco tiempo. La **detección de objetos** es una técnica de visión por computador que intenta replicar esa inteligencia para ser utilizada por ordenadores.

El eje central de los detectores de objetos basados en DL son las redes neuronales, que son las encargadas de extraer las características de las imágenes que se introducen al sistema para poder relacionarlas con las características que corresponden al objeto que se está intentando detectar.

A la hora de evaluar los sistemas de detección de objetos es interesante comprender el concepto de IoU (*Intersection over Union*), ya que es una métrica muy común entre diferentes tipos de arquitecturas de redes neuronales que sirve para estimar el rendimiento de los modelos. Este da un porcentaje de acierto del área de predicción frente al área que contiene el objeto a detectar.

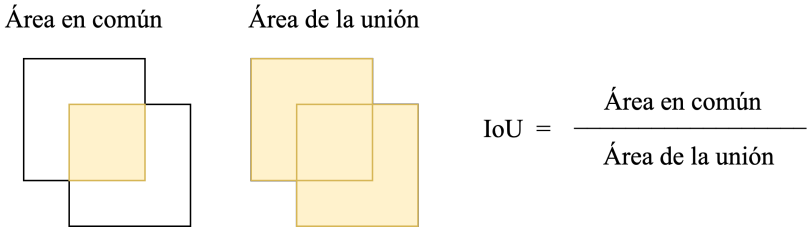


Figura 2: Cálculo de la IoU

Como se ve en la Figura 2, las dos áreas que aparecen en el cómputo de la IoU hacen referencia a cajas delimitadoras (*bounding boxes*) utilizadas para delimitar el área en la que se encuentra el objeto a detectar. Una de las cajas sería el resultado de la predicción, mientras que la otra es la caja con las coordenadas reales del objeto en la imagen (*Groundbox*). Esta última en un sistema de aprendizaje supervisado habría sido anotada previamente al entrenamiento. De esta manera, si la predicción fuera perfecta, el valor de IoU sería de 1. Si en cambio el área en común fuera menor a la de unión, entonces este medidor iría bajando hasta un mínimo de 0, disminuyendo también la fiabilidad de la detección.

En los procesos de entrenamiento se establecen umbrales de IoU para determinar si una detección es considerada válida o no. A continuación una serie de conceptos vinculados al umbral y al IoU:

- **Verdadero positivo (TP):** Es una detección correcta del objeto. Se produce

cuando se detecta el objeto y el IoU es mayor al umbral establecido.

- **Falso positivo (FP):** Es una detección incorrecta. Se produce cuando el modelo detecta un objeto pero al calcular el IoU, su valor es menor al del umbral.
- **Falso negativo (FN):** Es una detección incorrecta, pues sucede cuando existe una *Groundbox* pero el modelo no ha detectado ningún objeto.
- **Verdadero Negativo (TN):** No es un medidor relevante ya que haría referencia a todas las *bounding boxes* posibles que no contienen ni se ha detectado ningún objeto real. En una imagen pueden existir una infinidad de posibles cajas delimitadoras que no deben de detectar ningún objeto.

2.2. Arquitecturas existentes para la detección de objetos

En este subapartado se van a valorar diferentes tipos de arquitecturas a fin de determinar cuál se utilizará para resolver el problema planteado.

2.2.1. R-CNN

En 2014 surgieron las R-CNN (*Region Based Convolutional Neural Network*) con la propuesta de determinar regiones de interés dentro de la imagen y después clasificar las imágenes sobre esas áreas utilizando redes neuronales preentrenadas [6].

Este modelo integra varios algoritmos, pues primero determina las principales áreas de interés, las que pueden ser haste 2000 regiones de tamaños diferentes. Posteriormente estas imágenes son procesadas por una CNN (*Convolutional Neural Network*), aunque a veces se utilizan otros algoritmos, que mediante un clasificador binario determina si las clases detectadas son correctas y elimina aquellas cuyos indicadores de confianza no superen el umbral elegido. Finalmente, se ajustaría la posición del objeto mediante un regresor.

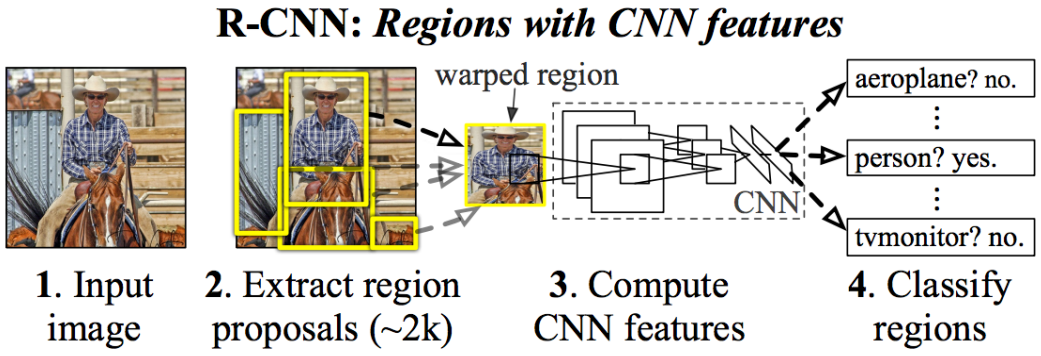


Figura 3: Funcionamiento simplificado de las R-CNN [7]

Debido a tener que implementar todos estos algoritmos durante la detección de objetos, incluso con técnicas que reducen el tiempo de detección (como NMS [8]), la detección de objetos en una única imagen podría durar 25 segundos. Esto una vez entrenada y validada, también teniendo en cuenta que los procesos de entrenamiento de estas redes también son lentos [6].

2.2.2. Fast R-CNN y Faster R-CNN

Como sus nombres indican, son dos modelos basados en las R-CNN que buscan mejorar el tiempo de detección de su predecesor.

- **Fast R-CNN:** Introducido en Abril de 2015, propone la reutilización de algunos recursos extraídos por la CNN para agilizar el entreno y detección de imágenes. Sin embargo, aunque mejora el tiempo de entrenamiento y detección en comparación con el algoritmo inicial, esta mejora no es sustancial ya que para muchas aplicaciones, la detección debería poder ser en tiempo real.
- **Faster R-CNN:** Dos meses después de la aparición de Fast R-CNN, este algoritmo irrumpió mejorando el anterior al añadirle un módulo previo que consiste en una red profunda totalmente convolucional que propone regiones, que posteriormente son utilizadas por Fast R-CNN para detectar mejor los objetos [9]. Además, una gran incorporación en este modelo es el uso de cuadros de anclaje (*fixed anchors*). Estos consisten en una serie de cuadros de anchos y altos predeterminados que sirven para tener una primera aproximación del tamaño del objeto, como se puede ver en las Figuras 4 y 5.

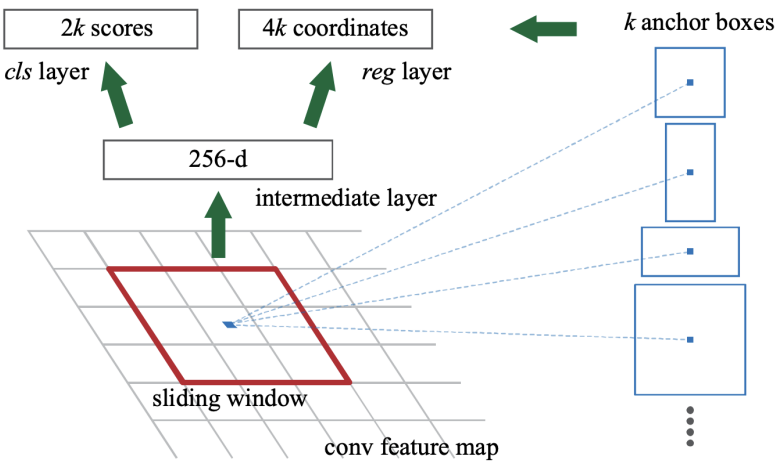


Figura 4: Algoritmo de Region Proposal Network (RPN) [9]

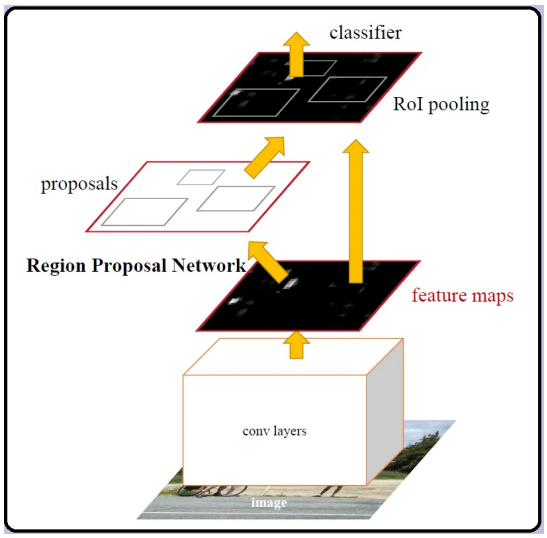


Figura 5: Funcionamiento de Faster R-CNN [9]

2.2.3. YOLO

Publicada por primera vez en 2016, YOLO (*You Only Look Once*) se presentó como un nuevo enfoque a la detección de objetos que utiliza una única red neuronal para predecir *bounding boxes* y las probabilidades de clase directamente de la imagen al completo en una única evaluación. Dado que todo el proceso de detección se lleva a cabo con una única red neuronal, esta se puede optimizar por completo de una manera más directa.

El modelo YOLO divide la imagen de entrada en una cuadrícula de $S \times S$ casillas como se puede observar en la Figura 6. Si el centro de un objeto se encuentra en una de las casillas, esta pasa a ser responsable de la detección del objeto. Cada casilla predice entonces un número B de *bounding boxes* y los índices de confianza para cada una de ellas. Estos índices de confianza reflejan cómo de seguro está la red de que en efecto la caja delimitadora contiene el objeto y cómo de fiable piensa que es la caja predicha, por lo que utiliza la IoU [10].

Ya en su primera versión supera al resto de modelos en cuanto a velocidad de detección [10]. También destaca una comparativa con Fast R-CNN en la que se determina que esta última es casi tres veces más propensa a producir errores de fondo (falsos positivos) que YOLOv1 [10]. Sin embargo, YOLOv1 es casi el doble de propensa a cometer errores de localización de objeto [10].

A día de hoy, existen varias versiones más actualizadas y mejoradas de YOLO. Entre ellas destaca la versión YOLOv8 de Ultralytics, introducida en Enero del presente año 2023, que está diseñada para ser rápida, precisa y fácil de usar para un amplio abanico de problemas de detección de objetos [11].

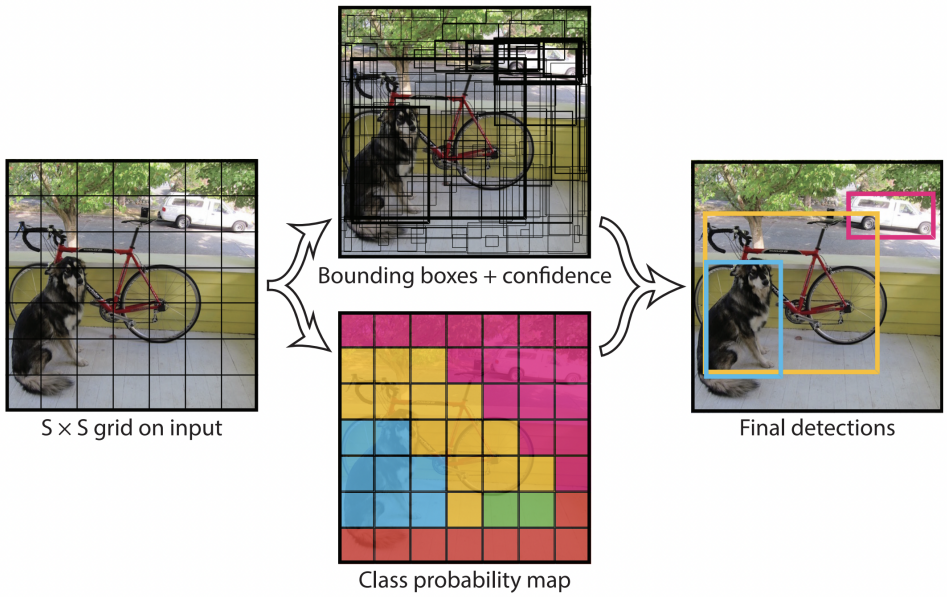


Figura 6: Modelo con cuadrícula de detección YOLO[10]

Esta última versión de YOLO no solo trae consigo mejoras en cuanto a precisión y velocidad, como se ve en la Figura 7, sino que también reduce los requisitos de preprocesamiento de las imágenes. En versiones anteriores de YOLO, las imágenes debían tener un ancho y alto cuyos píxeles fueran un múltiplo de 32. YOLOv8, sin embargo, acepta imágenes de tamaños diferentes, las que escala automáticamente para que cumplan los requisitos. Este escalado no deforma las imágenes sino que añade áreas grises para conseguir altos y/o anchos con píxeles múltiplos de 32 [12].

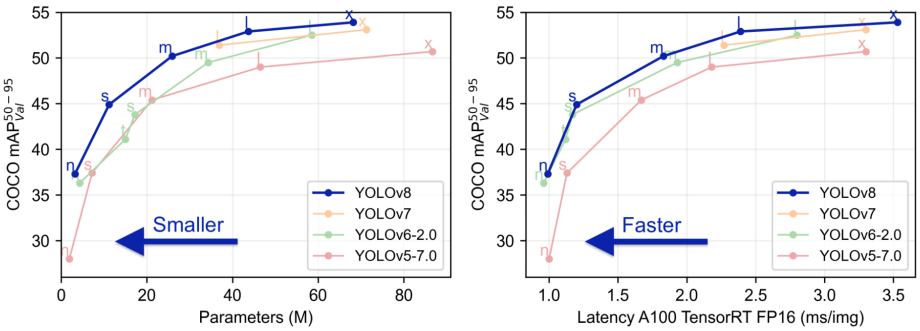


Figura 7: Comparativa entre versiones de YOLO entrenadas con COCO dataset

Para mantenerse en la vanguardia en cuanto arquitecturas de redes neuronales y debido a que este es uno de los modelos más rápido y completos hasta la fecha, se ha elegido utilizar la octava versión de YOLO de Ultralytics para la discriminación de envases. En el siguiente apartado se explicará en qué consiste este modelo en mayor detalle.

2.3. Trabajos similares

Teniendo en cuenta que el uso de redes neuronales para la detección de objetos es bastante reciente, no existen muchos trabajos similares. Durante la etapa de documentación se han buscado proyectos similares que puedan ser relevantes a la hora de hablar de redes neuronales para la detección de objetos.

Los documentos encontrados son los siguientes:

- **Estudio de la arquitectura YOLO para la detección de objetos mediante *deep learning*** [13]. Este trabajo analiza la trayectoria de diferentes arquitecturas de redes neuronales para la detección de objetos en imágenes hasta llegar a YOLO de Ultralytics v3. En él se discuten los conceptos básicos a la hora de entrenar la arquitectura YOLO de aquel momento para reconocer elementos como perros y cuadros.
- **Diseño, implementación y evaluación de técnicas genéricas de detección de C. Elegans mediante redes neuronales convolucionales** [14]. Un Trabajo de Fin de Grado presentado en la ETSID (UPV) que utiliza Yolov5 para la detección de un único tipo de elemento (C. Elegans).
- ***Classification of Trash for Recyclability Status*** [15]. Este trabajo está bastante relacionado con el objeto del presente TFG, pues utiliza redes neuronales para separar imágenes de basura según el criterio de reciclado de Estados Unidos. El separado se hace entonces en seis categorías principales: vidrio, cartón, papel, plástico, metal y restos. Para ello utilizan redes neuronales convolucionales y SVM (*Support Vector Machines*). Además, crean también una base de datos de imágenes propia para llevar a cabo su estudio.

3. YOLOv8 de Ultralytics para detección

Tras elegir la versión más reciente de Yolo Ultralytics, procedemos a adentrarnos en esta para comprenderla mejor. Entre las características principales de YOLOv8 destacan:

- **Arquitecturas de *Backbone* y *Neck* avanzadas:** Los módulos *Backbone* están compuestos por varias capas que se encargan de extraer las características esenciales de las imágenes a diferentes resoluciones. Estas características extraídas se pasan entonces a una arquitectura *Neck* que las fusiona para que se produzca la detección en otro módulo (*Head*).
- **YOLOv8 es *Anchor-Free*:** Esta nueva versión de YOLO ya no predice cajas delimitadoras enteras, sino que directamente se centra en el punto central del objeto. Esta técnica reduce el número de predicciones de cajas delimitadora, lo cual acelera el proceso NMS [8] al no tener que comparar tantas cajas propuestas.
- **Apuesta por mantener el equilibrio entre precisión y velocidad:** Consiguiendo así ser capaz de detectar objetos en tiempo real en una gran variedad de áreas de aplicación.
- **Diferentes modelos preentrenados:** Ofrece modelos diferentes para llevar a cabo diferentes tipos de tareas (detección, segmentación, posición y clasificación) y responder a requisitos de rendimiento diversos.

Para tareas de detección, Ultralytics ofrece 5 modelos preentrenados. El más pequeño de ellos *YOLOv8 NANO* (YOLOv8n.pt) es el más rápido de todos pero también el menos preciso, mientras que el más grande *YOLOv8 EXTRA LARGE* (YOLOv8x.pt) es el más lento pero gana al resto en cuanto a precisión. Esto se puede apreciar en la Figura 8, donde se compara la precisión media (mAP) y la velocidad (*speed*) de los diferentes modelos preentrenados con la base de datos COCO (*Common Objects in Context*) [16], la cual es comúnmente utilizada para evaluar el rendimiento de los modelos de detección de objetos .

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

Figura 8: Comparativa de modelos preentrenados YOLOv8 para detección [11]

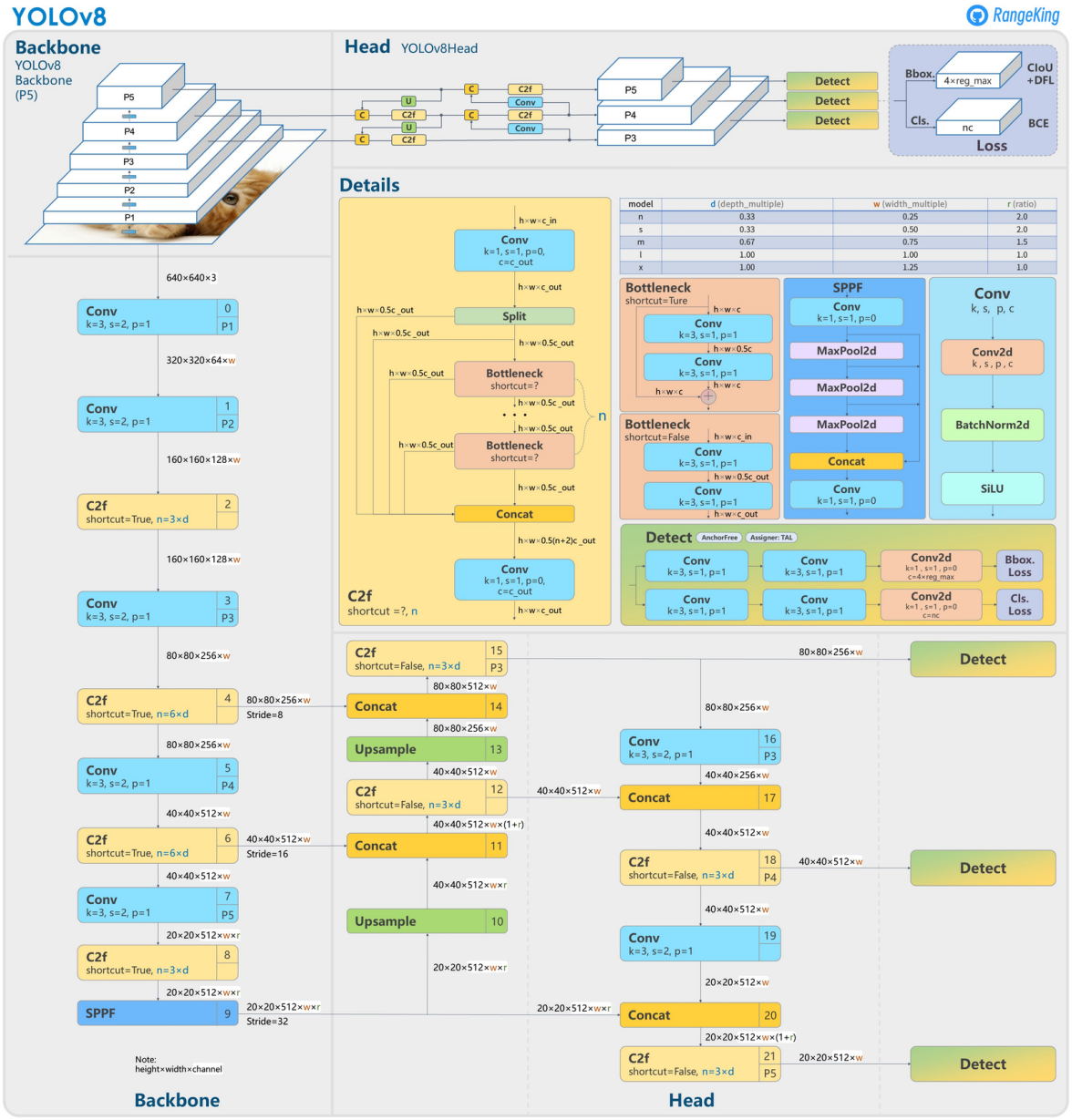


Figura 9: Arquitectura de YOLOv8 [17]

3.1. Requisitos

3.1.1. Instalación

YOLOv8 está diseñado para ser programado en Python. Para poder entrenarlo de manera local, se debe de instalar el paquete de Ultralytics. Este se puede descargar desde el terminal con la siguiente línea: `pip3 install ultralytics`.

3.1.2. Formato de los datos

El formato definido por YOLO de Ultralytics para las bases de datos consiste un archivo `.yaml` que contenga las rutas a los diferentes directorios con la imágenes y anotaciones de la base de datos, así como una lista enumerada empezado por el 0 de todas las clases que se han definido durante la anotación.

El formato del archivo `.yaml` es el siguiente:

- `path:` Este debe de estar seguido de la ruta global donde se encuentre la base de datos.
- `train:` Ruta relativa respecto a `path` donde se encuentra el subset para el entrenamiento.
- `val:` Ruta relativa respecto a `path` donde se encuentra el subset para la validación de la red entrenada.
- `test` **(opcional):** Ruta relativa respecto a `path` donde se encuentra el subset para el testeo de la red entrenada.
- `names:` Contiene la lista de clases enumeradas desde 0 hasta nº clases - 1, con los nombres que se les quiera dar.

```yaml
path: /Users/marcos/miniconda3/envs/torch-gpu/TFG_3/code/data # dataset root dir
train: images/train # train images (relative to "path")
val: images/val     # validation images (relative to "path")
test: images/test   # test images (relative to "path")

# Classes
names:
  0: Envase_Tapa
  1: Tetrabrick
  2: Lata_Bebida
  3: Lata_Conserva
  4: Aerosol
  5: Vidrio_Botella_grande
  6: Vidrio_Botella_peq
  7: Plastico_Botella_grande
  8: Plastico_Botella_peq
  9: Plastico_Botella_otros
  10: Plastico_Vaso
  11: Plastico_Bote
  12: Plastico_Garrafa
  13: Plastico_Envoltorio
```

Figura 10: Ejemplo del contenido del archivo `.yaml`

Dentro de cada directorio mencionado anteriormente (*train, val* y *test*), se deben de crear dos subcarpetas: "*images*" y "*labels*". En la primera subcarpeta es donde deben ir todas las imágenes y en la segunda, las correspondientes anotaciones de esas imágenes que deben de tener el mismo nombre y tener la extensión `.txt`.

El formato de las anotaciones dentro de los archivos de texto tiene que tener una línea de texto para cada objeto que define la clase y las cajas delimitadoras siguiendo esta estructura: `class | x_center | y_center | width | height`.

- `class`: Este es un número entero que sirve para identificar la clase del objeto que está siendo etiquetado definido en el archivo `.yaml`.
- `x_center` e `y_center`: Coordenadas x e y del punto central de la *bounding box* en valores normalizados de 0 a 1. Se normalizan dividiendo la coordenada horizontal (x) por el ancho de la imagen y la coordenada vertical (y), por el alto.
- `width` y `height`: Ancho y alto de la *bounding box* con valores normalizados de 0 a 1. Al igual que los puntos centrales, estos se normalizan dividiendo el ancho de la caja entre el ancho de la imagen y alto de la caja entre el alto de la imagen.

A continuación un ejemplo extraído de la documentación de YOLOv8 de Ultralytics [18] que ilustra la delimitación de las cajas y cómo se deben de anotar:

Figura 11: Ejemplo de dibujo de las *Bounding boxes*

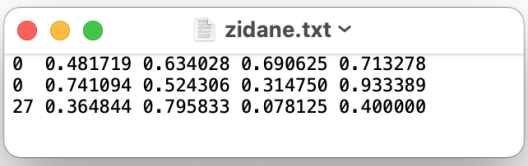


Figura 12: Ejemplo del contenido del archivo `.txt`

En las Figuras 11 y 12 las cajas delimitadoras naranjas hacen referencia a las personas (clase: 0) y la verde a la corbata (clase: 27).

3.2. Funcionalidades

3.2.1. Modos de detección

YOLOv8 para la detección es capaz de detectar objetos en imágenes y en fotogramas de vídeos, así como de dibujar las cajas delimitadoras que los contienen. Todos los modelos de detección de YOLOv8 se pueden entrenar, validar y usar para llevar a cabo predicciones sobre imágenes y vídeos.

Los modos de uso son:

- **Train**: Este se utiliza para entrenar un modelo, desde cero o sobre uno de los modelos preentrenados disponibles en el Cuadro 8, con una base de datos propia. Durante esta etapa se puede especificar el *dataset* y los hiperparámetros de entrenamiento.
- **Val**: Este modo se utiliza para validar un modelo de YOLOv8 una vez haya sido entrenado. La finalidad de este es medir la precisión del modelo y su rendimiento. La información que resulta de este proceso puede servir para retocar los parámetros del entrenamiento y así mejorar el rendimiento de este.
- **Predict**: Este sirve para generar predicciones visuales o en forma de ficheros de texto sobre imágenes no utilizadas durante la fase de entrenamiento. El modelo se utiliza para predecir las clases y sus localizaciones dentro de las imágenes y vídeos que se le introduzcan.

3.2.2. Hiperparámetros durante el entrenamiento

Los parámetros que se pueden configurar para el entrenamiento son variados. De manera predeterminada están configurados por Ultralytics según los que mejor llevan a cabo el entrenamiento de los modelos con la base de datos de COCO, pero estos pueden ser cambiados fácilmente para entrenamientos con bases de datos diferentes.

Entre los argumentos más importantes se encuentran el número de épocas (*epochs*), la paciencia (*patience*), el ritmo de aprendizaje (*learning rate*), el tamaño de lote (*batch size*) y el tamaño de la imagen *imgsz*:

- `epochs:` Se refiere al número de épocas para las que se quiere entrenar. Una época es un paso completo de la base de datos por la red neuronal.
- `patience:` Es el número de épocas máximas que se espera antes de parar el entrenamiento en caso de que no se observe una mejora significante en este. Consiste en una serie de cálculos internos que tiene en cuenta las métricas de entrenamiento.

- `lr0`: Ritmo de aprendizaje inicial. Este hiperparámetro controla el ritmo al que la red neuronal actualiza los pesos mientras busca el mínimo de la función de pérdidas. Suele oscilar entre 0.0001 y 0.01, pues se considera un rango de valores óptimo para la mayoría de los casos.
- `batch size`: Es el número de muestras de imágenes que se procesan antes de que se actualicen los pesos del modelo. Esta debe de ser mayor que cero y menor que el número máximo de muestras en la base de datos. Cuanto mayor sea el número, más recursos demandará al dispositivo en el que se realiza el entrenamiento. Este valor se puede poner en automático (`batch = -1`) para que YOLO determine el número óptimo en base al ordenador.
- `imgsz`: Esta presenta una mejora con respecto a versiones anteriores de YOLO. Anteriormente, el modelo debía ser alimentado con imágenes preprocesadas para que tuvieran un ancho o alto con un valor de píxeles múltiplo de 32. Sin embargo, ahora solo basta con especificar un número entero o valores de ancho y alto, ya las imágenes de la muestra serán escaladas acorde a él. Además, esto no deforma las imágenes, sino que en caso de tener que alterar la imagen para cumplir un determinado alto y/o ancho, YOLOv8 se encarga de añadir un área de píxeles grises para rellenar ese espacio que falta.

3.2.3. Métricas de entrenamiento

Los detalles exactos de cómo Ultralytics calcula estas métricas no están a disposición del público para evitar la vulneración de la propiedad intelectual. Sin embargo, estas métricas se pueden definir a grandes rasgos de la siguiente manera:

- ***Box Loss:*** Representa cómo de bien el modelo puede localizar el centro del objeto y si el grado de éxito en cuanto a cómo ha sido delimitado por la *bounding box*.
- ***Class Loss:*** Es una función de perdida que mide cómo de correcta es la clasificación en clases de cada una de las cajas delimitadoras predichas. Es calculada basándose en la función *Binary Cross-Entropy loss* con los coeficientes de confianza de las cajas delimitadoras predichas por el modelo durante el entrenamiento.
- ***Distribution Focal Loss:*** Ayuda a mejorar el rendimiento en los entrenamientos que utilicen una base de datos con clases poco equilibradas. Es decir, que diferentes clases tengan números dispares de muestras de una a otra. Esto puede ser un problema sobre todo cuando estas clases con pocas instancias tienen muestras de objetos poco frecuentes, lo que puede provocar que aprender a reconocer estos resulte un desafío para la red. Esta función está enfocada en aliviar este problema y asegurarse que el modelo detecta correctamente los objetos menos frecuentes [19].

El objetivo claro durante el entrenamiento es que todas las funciones de perdidas anteriores vayan bajando tras cada época como se puede observar en la Figura 13. Sin embargo, que estas funciones de pérdida sigan esta trayectoria descendiente no garantiza que el modelo esté siendo entrenado correctamente sin problemas de *overfitting*, entre otros. Es importante comprobar que las gráficas no van en sentido ascendente pero no deben ser utilizadas como única métrica para evaluar el entrenamiento de la red. Se consideran más significativos los resultados de la validación de la red entrenada.

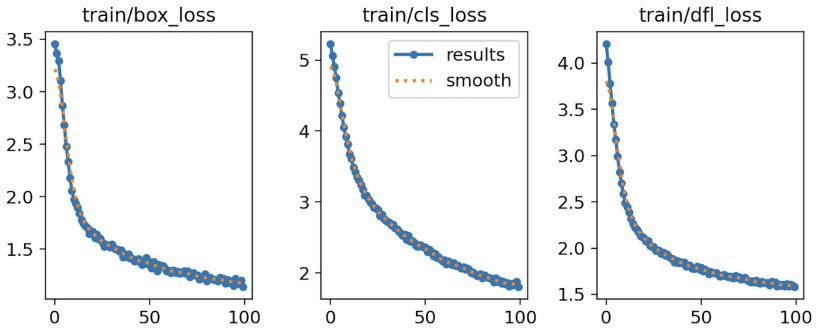


Figura 13: Ejemplo de métricas de entrenamiento después de 100 épocas

3.2.4. Métricas de validación

Para poder entender los resultados de la validación, se deben explicar una serie de métricas que dan información acerca del rendimiento del modelo:

- **Matriz de Confusión**

  Se trata de una matriz cuadrada en la que se comparan los valores predichos por el modelo con los reales, es decir, los anotados manualmente previo entrenamiento. En ella se puede ver muy claro los falsos positivos y los verdaderos positivos. Estos últimos se organizan en la diagonal descendente de la matriz, mientras que los falsos positivos serían todos aquellos fuera de la diagonal. El cambio de blanco a azul representa la cantidad de detecciones, siendo el blanco el menor número de detecciones y el azul oscuro, el mayor. Por lo tanto, una validación buena tendrá los valores más oscuros en la diagonal.

- **Precisión**

  La precisión es una métrica muy importante a la hora de validar un modelo. Indica qué fracción de las cajas delimitadoras propuestas por el sistema son las que verdaderamente contienen el objeto que se intenta detectar. En un modelo ideal, su valor sería igual a 1.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

- **Recall**

  La sensibilidad (*Recall*) de un sistema se refiere a la fracción de objetos de todos los que existen que son encontrados por el detector. Idealmente, la sensibilidad de un sistema sería de 1.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

- **Curva PR**

  Esta gráfica relaciona los conceptos de precisión y sensibilidad. Idealmente, ambas serían igual a 1. Sin embargo, en la práctica esto no ocurre. Si se configura un sistema para obtener una precisión alta, por lo general esto supondrá una disminución de la sensibilidad (el sistema será más preciso en lo que considera un objeto, pero esto reducirá el número de detecciones). En cambio si se busca una sensibilidad mayor, el número de detecciones será mayor conteniendo a su vez una mayor proporción de falsos positivos (descenso en la precisión).

  Un detector es considerado bueno si cuando aumenta la sensibilidad del sistema, la precisión se mantiene estable en valores altos. En la Figura 15 se puede ver la comparativa entre la curva ideal y la de un sistema real con resultados aceptables. En entrenamientos reales, se busca que la curva PR se aproxime lo máximo posible a la ideal.

- **Precisión media (mAP)**

  Esta es una de las métricas más utilizadas para evaluar el rendimiento de un modelo entrenado en YOLO. Consiste en calcular el área bajo la curva PR, resultando en una medida numérica cómoda de comparar entre diferentes modelos. Cuanto mayor sea el valor, significa que el área contenida debajo de la curva PR es mayor (la curva real se aproxima a la ideal).

  YOLOv8 utiliza comúnmente el valor de la mAP con un umbral de IoU de 50 %, aunque se puede calcular para otros umbrales. También se utiliza mAP[50-95], que es la precisión media para valores de IoU desde 0.5 hasta 0.95 en incrementos de 0.05.

  Actualmente, gran parte de los artículos sobre arquitecturas de redes neuronales suelen reflejar el rendimiento de sus modelos con la mAP[50-95] para el *dataset* COCO (como se ha visto anteriormente en la Figura 8).

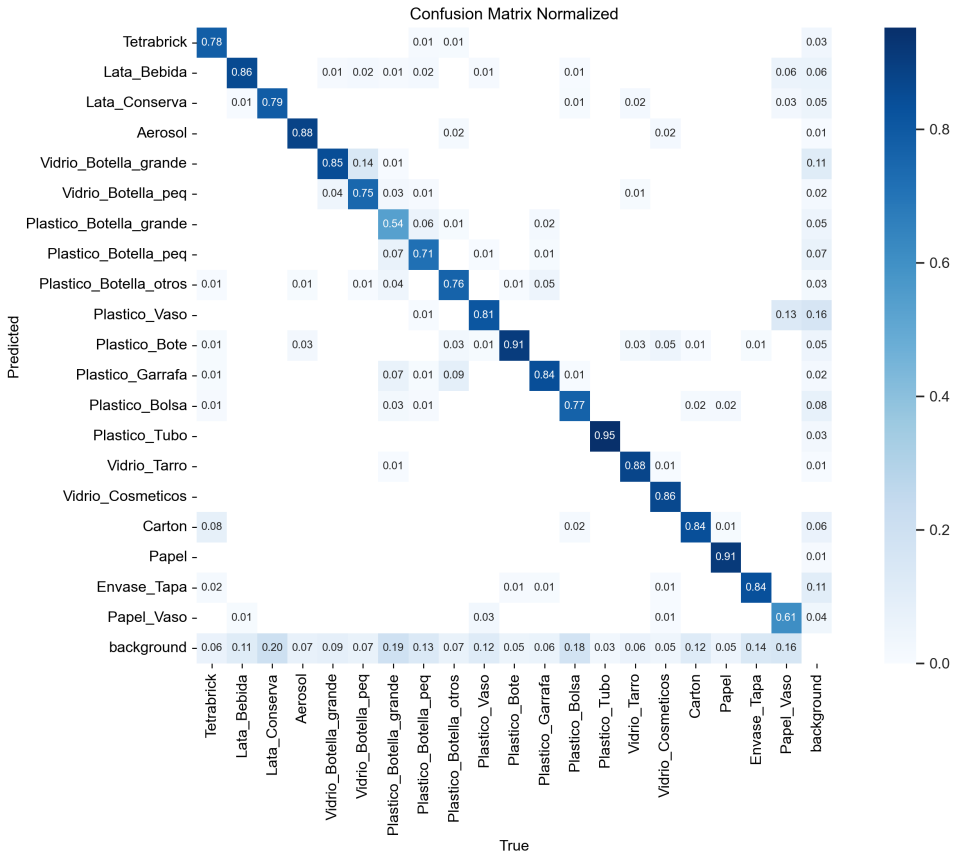


Figura 14: Ejemplo de matriz de confusión

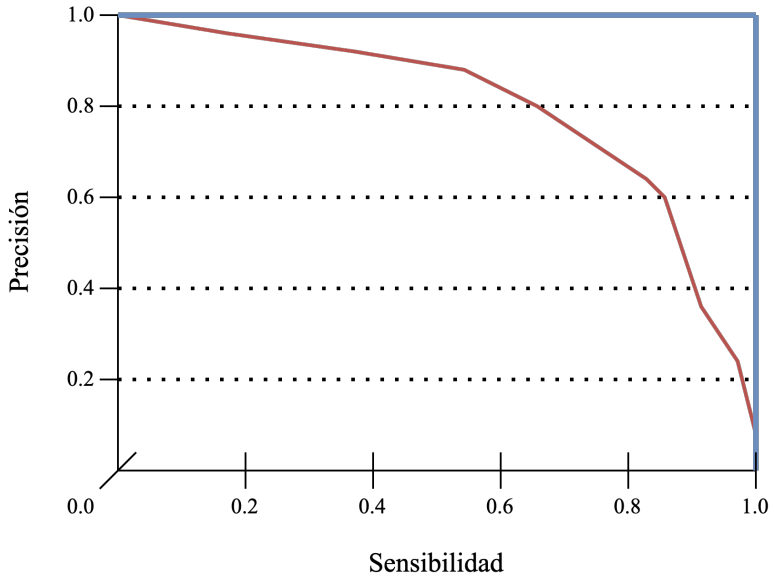


Figura 15: Comparativa de una curva PR ideal (azul) y una real (rojo)

4. Materiales y métodos

En este capítulo se encuentran recogidos una serie de materiales y métodos utilizados durante el desarrollo de este proyecto.

4.1. Base de datos de imágenes

Debido a las especificaciones de este proyecto, no se ha podido encontrar ninguna base de datos anotada que responda por completo a las necesidades de este TFG en cuanto a su definición de las clases a detectar. Es por ello que se ha tenido que confeccionar una base de datos de envases personalizada para entrenar las redes neuronales. A este dataset se le ha bautizado con el nombre de **ContaiNET**.

4.1.1. Obtención de las imágenes

Las imágenes de envases que conforman ContaiNET se han obtenido de diversas fuentes. Se ha contactado a diversos supermercados con oficinas en el territorio español y se les ha pedido utilizar las imágenes de envases que aparecen en sus respectivas páginas web. De los supermercados contactados, Consum y Carrefour cedieron el uso de sus imágenes para ContaiNET.

Asimismo, el dataset también cuenta con una porción de imágenes obtenidas por cuenta propia con una cámara de teléfono móvil, así como una serie de imágenes descargadas automáticamente de Google Imágenes a través de un *script* de Python disponible en el Anexo I.

Para terminar de completar la muestra, se ha hecho uso también de una base de datos de imágenes libre de derechos con imágenes de basura llamada **TrashNET** [15]. Esta base de datos preexistente organiza las imágenes en función del material para su posterior reciclaje pero no tiene en cuenta el tipo de envase, por lo que no todas las imágenes de esta base de datos han sido utilizadas de cara a la confección de ese nuevo dataset que las integra.

A continuación, el número de imágenes obtenido de cada fuente y el total de imágenes que conforman el dataset personalizado de este TFG:

Fuentes:	Carrefour	Consum	Propias	Script.py	TrashNET	**Total**
Nº imágenes:	361	592	582	772	2399	**4706**

Cuadro 1: Composición del dataset personalizado: ContaiNET

4.1.2. Definición de las clases de objetos a detectar

La base de datos ContaiNET ha sido dividida y anotada agrupando la mayoría de los envases de los que se hace uso en la cotidianidad en 20 clases diferentes:

- **Tetrabrick:** Incluye tetrabricks de todos los tamaños.
- **Lata de bebida:** Latas destinadas exclusivamente para contener líquidos (refrescos, bebidas alcohólicas, etc).
- **Lata de conserva:** Todas aquellas latas que su uso esté destinado a contener algo diferente a un líquido (latas de pescado, sopa, tomate triturado, cacao en polvo, pimentón, etc).
- **Aerosol:** Envases destinados a contener altas concentraciones de gas propulsor comprimido (desodorantes, insecticidas, etc.)
- **Botella de plástico grande para bebidas:** Envases grandes exclusivamente de bebidas aptas para el consumo humano con un pico alargado y estrecho en comparación con la base (volumen no inferior a 1L).
- **Botella de plástico pequeña para bebidas:** Similar al anterior pero de menor tamaño (volumen máximo inferior a 1L).
- **Botella de plástico para otros usos:** Incluyen envases sin asa con forma de botella destinado a contener productos de limpieza y/o líquidos no considerados bebida (aceites y vinagres).
- **Garrafa de plástico:** Aquellas botellas de plástico cuya forma presenta un asa prominente (líquido floculante para piscinas o detergente de lavadora), así como garrafas estándar de agua y/o aceite.
- **Bote de plástico:** Todo envase de plástico que se asemeje a una botella sin pico, a una caja o un cubo con tapa (tarrinas de mantequilla, cubos de yogur griego grande, cubos de pastillas de cloro, etc.)
- **Tubo de plástico:** Todo envase cilíndrico y alargado que presente un extremo sellado y otro con una apertura (similar a los envases generalmente utilizados para la pasta de dientes, pintura acrílica, etc).
- **Envoltorio de plástico:** Incluye bolsas, envoltorios y trozos de láminas de plástico.
- **Vaso de plástico:** Vasos de plástico así como envases del tipo de las tarrinas individuales de yogur y postres como gelatina y natillas, entre otros.

- **Botella de vidrio grande:** Todo envase de vidrio grande destinado a contener líquidos aptos para el consumo humano con un pico alargado y estrecho en comparación con la base, que no tiene por qué ser redonda (volumen como mínimo de 75cl).
- **Botella de vidrio pequeña:** Similar a la anterior pero de un tamaño más reducido (envases que no sobrepasen los 75cl).
- **Tarro de Vidrio:** Incluye envases de vidrio destinados a contener alimentos y líquidos que no presenten una abertura ancha y no alargada como los picos de las botellas (tarros de legumbres hidratadas, salsas, aceitunas, etc.)
- **Vidrio para cosméticos:** Todo aquel envase de vidrio destinado a maquillaje, pintura de uñas, perfume, etc.
- **Cajas y cartón:** Envases de cartón que se asemejen a una caja o partes de cajas, así como láminas y trozos de cartón.
- **Papel:** Incluye trozos de papel, servilletas, bolsas y hojas de papel.
- **Vaso de papel:** Incluye todo envase de papel que se asemeje a un vaso, en todas sus tamaños (vasos y cubos de papel).
- **Tapa de envase:** Todo elemento que sirva para tapar un envase. Varía en material (puede ser de plástico o de chapa como la mayoría de las tapas de los tarros de vidrio), en forma (puede tener forma de tapón, de pulverizador, de dispensador como en el caso de los botes de jabón líquido, etc) y en tamaño.

4.1.3. Anotaciones de las clases

Para poder entrenar una red neuronal con aprendizaje supervisado deben de ser anotadas todas y cada una de las imágenes. Este es un proceso largo y tedioso pero necesario, que debe ser llevado a cabo con cuidado para luego obtener métricas relevantes tras la validación de las redes.

En la Tabla 2 se puede observar el número de anotaciones finales por clase. El número de instancias (cada una de las anotaciones que se ha hecho para cada objeto) no es el mismo para cada clase, esto se debe a que algunas clases tienen características comunes más marcadas entre sus objetos, mientras otras clases engloban objetos con características más dispares, teniendo entonces que aumentar la muestra.

Esto es lo que sucede por ejemplo con los tetrabricks (clase en la que aunque cambien los tamaños un poco, la forma presenta pocas variaciones de uno a otro), con las latas de conserva (esta clase presenta más variedad en cuanto a qué es considerada una lata de conserva, pues hay de diferentes formas y tamaños aunque aún presentan también

bastantes similitudes) o las tapas de envases (esta clase es muy dispar, pues recoge tapas de diferentes materiales, formas y propósitos). Es por ello que para la primera de estas, el número de instancias es de alrededor de 300, para la segunda se cuadriplica prácticamente y para la tercera es aproximadamente 12 veces mayor. Esta diferencia en la variedad se puede apreciar en la Figura 16.

Nombre de clase	**Nº de instancias**
Tetrabrick	353
Lata de bebida	750
Lata de conserva	1212
Aerosol	279
Botella de plástico grande (bebida)	305
Botella de plástico pequeña (bebida)	539
Botella de plástico (otros usos)	476
Garrafa de plástico	272
Bote de plástico	1283
Tubo de plástico	607
Envoltorio de plástico	1093
Vaso de plástico	790
Botella de vidrio grande	753
Botella de vidrio pequeña	538
Tarro de vidrio	385
Vidrio para cosméticos	172
Cajas y cartón	1393
Papel	798
Vaso de papel	372
Tapa de envase	4018
Total de instancias	**16388**

Cuadro 2: Total de instancias anotadas por clase

El número de instancias es mayor al de imágenes totales en ContaiNET, ya que la mayoría de imágenes contiene más de un objeto. Debido a que hay un

(a) Tetrabricks

(b) Latas de conserva

(c) Tapas de envases

Figura 16: Ejemplos de variedad dentro de las clases

4.2. CVAT.AI

CVAT.AI es el software en línea que se ha utilizado para dibujar *bounding boxes* en las imágenes de la base de datos y exportar las anotaciones en ficheros `.txt`. Se trata de una herramienta completa e intuitiva que permite de manera gratuita, aunque también ofrece servicios de pago, crear proyectos de anotado para una gran variedad de aplicaciones (anotación de imágenes, vídeos e incluso 3D).

Para el desarrollo de este proyecto solo se han usado las funcionalidades de creación de cajas rectangulares para delimitar los objetos de cada clase. Primero se debe crear un proyecto en CVAT.AI donde se declaran las clases con las que se va a trabajar, luego se crean tareas de anotación donde se suben a la web las imágenes que se pretende anotar y se procede al anotado con las herramientas que ofrece para hacer este proceso lo más dinámico posible. A continuación, una explicación más en detalle de cómo se deben seguir estos pasos.

4.2.1. Creación de un proyecto y tareas

El primer paso es crearse una cuenta gratuita. Esta tiene limitaciones en cuanto al límite de proyectos y tareas que se pueden crear, sin embargo, la versión gratuita cumple cómodamente todas las especificaciones de este proyecto en cuanto anotado.

Tras la creación de la cuenta, aparece una menú en la parte superior izquierda con diferentes secciones. En el apartado de "*Project*" se puede crear un proyecto nuevo. Este deberá tener un nombre y en el apartado de "labels" se deberá introducir una a una todos los nombres de las clases que se van a anotar. Es importante que tanto las clases introducidas en CVAT.AI como las clases del archivo `.yaml` estén en el mismo orden para que la numeración sea la misma.

Una vez creado el proyecto, se crean tantas tareas como se vea necesario (con un máximo de 10 para la versión gratuita). Cada tarea es un subgrupo de anotación de imágenes. Se puede crear un único grupo en el que se introduce la totalidad de las imágenes de la base de datos o, como en el caso particular de este proyecto, se puede dividir en tantas tareas como fuentes de imágenes se han utilizado para crear ContaiNET.

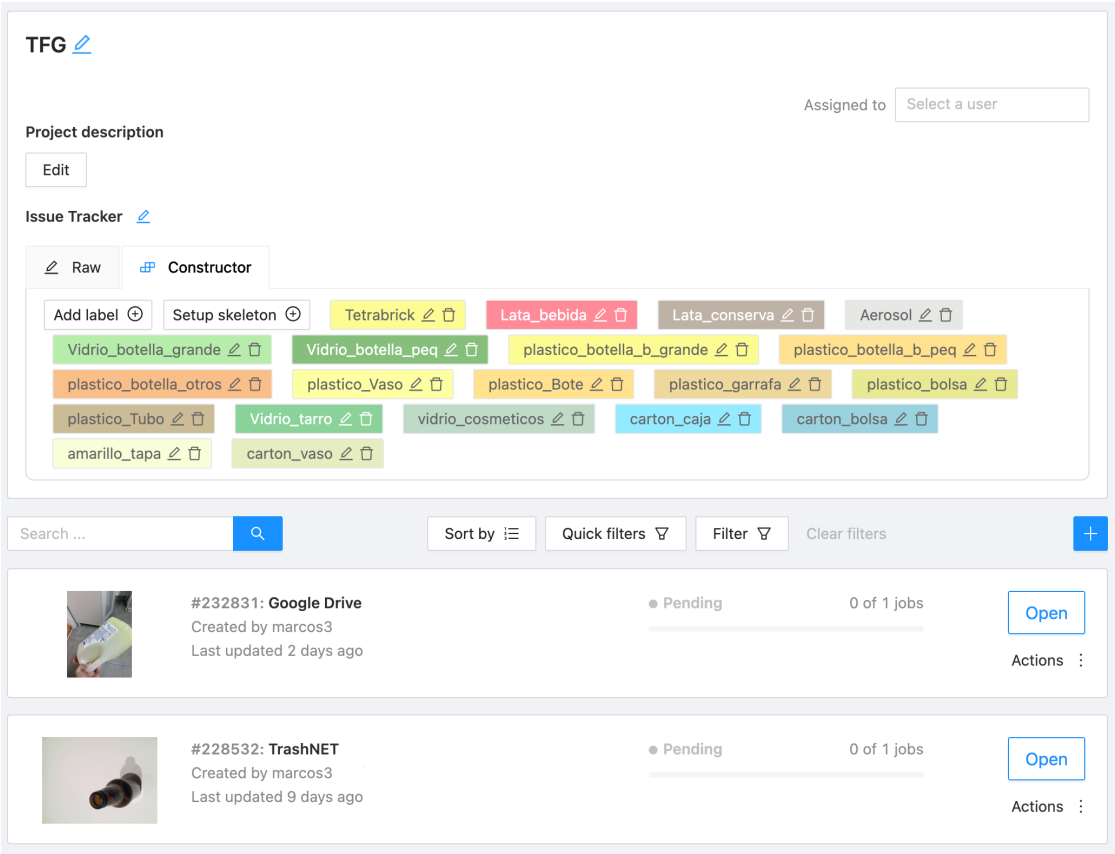


Figura 17: Creación de un proyecto y tareas en CVAT.AI

4.2.2. Anotación de las tareas

Al abrir una tarea de etiquetado aparece una interfaz. En el centro de la pantalla aparece la imagen. A la izquierda de esta se encuentra una barra de herramientas, en la parte superior hay un navegador de imágenes y a la derecha, un listado donde van apareciendo las etiquetas conforme se van creando (indicado con un 2 en la Figura 18).

En la Figura 18 se puede ver este entorno. Para el etiquetado se debe hacer *click* en la herramienta con forma de rectángulo señalada con el número 1. En ella se elige la clase que se quiere etiquetar, con un método de etiquetado en dos puntos y seguidamente se elige la opción "*Shape*" como en la Figura 19. De esta manera se puede dibujar la caja como está indicado en 3 de la Figura 18.

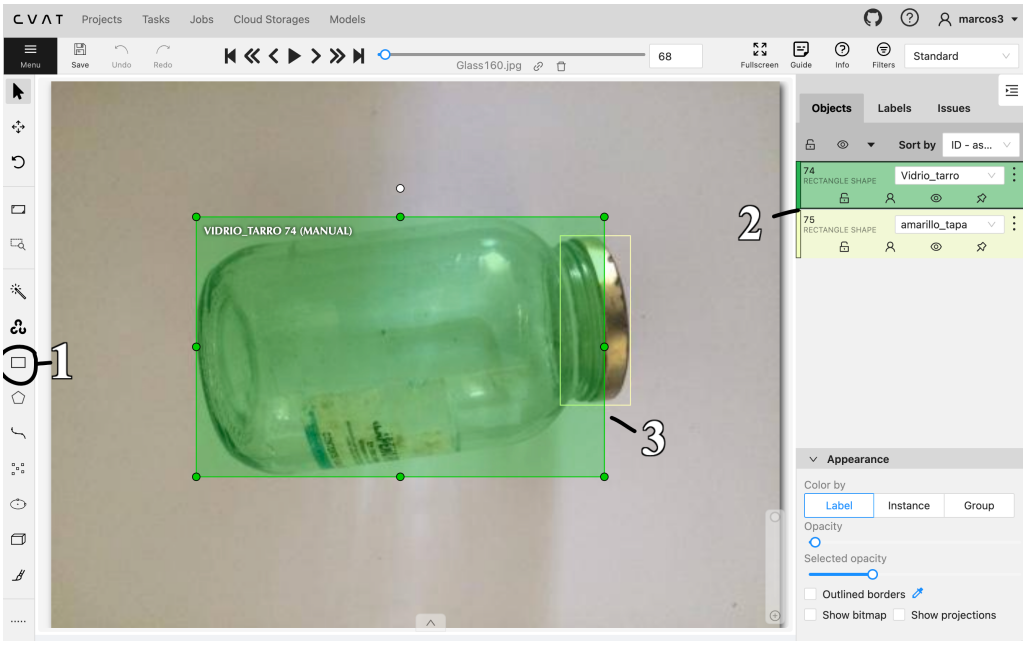


Figura 18: Ejemplo de dibujo de *Bounding boxes* con CVAT.AI

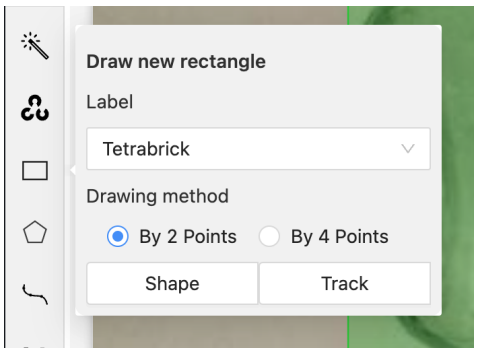


Figura 19: Herramienta rectángulo en CVAT.AI

4.2.3. Exportación de las anotaciones

Tras terminar de etiquetar todas las imágenes del proyecto, solamente falta exportarlas en el formato que se necesite. Para ello, se vuelve a la página del proyecto y se selecciona el icono con tres puntos al lado del proyecto, para después hacer *click* sobre "*Export dataset*".

Finalmente se elige el formato en el que se quieren exportar las anotaciones. Para YOLOv8 el formato es: `YOLO 1.1`, sin embargo para llevar a cabo la distribución estratificada de las anotaciones de las clases, se necesitará también exportarlo en formato `COCO 1.0`. Esto último será explicado en el apartado 5.1 de la memoria, donde se explica cómo se ha llevado a cabo la división del *dataset* para entrenamiento, validación y test de la red.

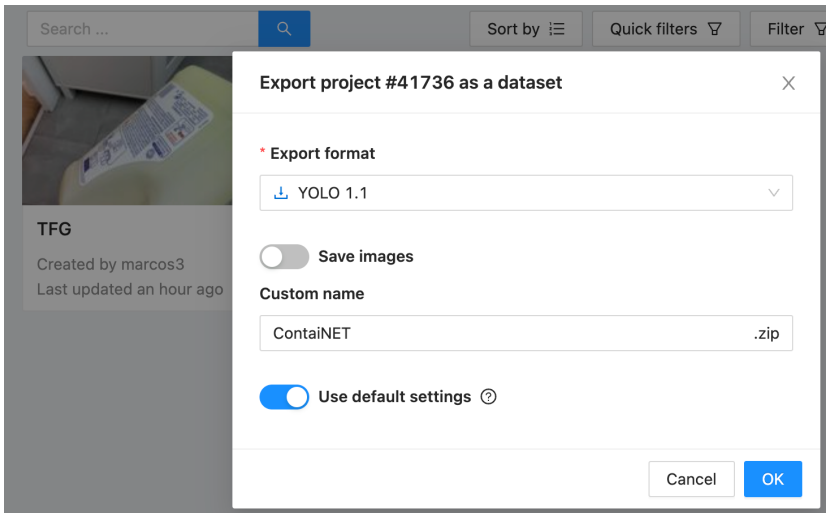


Figura 20: Ejemplo de exportación de las anotaciones en CVAT.AI

4.3. Hardware

Los únicos elementos físicos utilizados durante el desarrollo de este trabajo son:

- **MacBook Pro 14" 2023:** Con chip M2 utilizado como único ordenador destinado a organizar el *dataset* además de entrenar, validar y testear las redes neuronales. Se utiliza la CPU en vez de la GPU porque YOLOv8 es nueva y todavía no está adaptado para este chip. Sin embargo, el entrenamiento con la CPU de macOS no se ve perjudicado comparado con las GPUs de otros equipos.
- **iPhone XR 128GB:** Para tomar fotografías de envases que conforman parte del *dataset*.

5. Descripción del método experimental

En este apartado se va a explicar el criterio que se he seguido para la división de ContaiNET en diferentes subgrupos para el entrenamiento, validación y test de la red, así como se expondrán los experimentos llevados a cabo para entrenar un detector de objetos basado en la versión nano de YOLOv8.

Además, a modo de puntualización, los experimentos explicados más adelante no han sido los únicos llevados a cabo durante la preparación de este TFG. Antes de llevar a cabo estos experimentos con la base de datos al completo, se hicieron una serie de pruebas. Estas utilizaron menor número de muestras y presentaban fallos conceptuales debido a estar todavía aprendiendo a entrenar una red neuronal. Los resultados de esas pruebas, aunque no son relevantes, han servido para fijar ciertos aspectos del Trabajo de Fin de Grado como el número de clases finales que se ha decidido utilizar. En un principio se fijaron hasta 36 clases diferentes a detectar, que finalmente resultaron en la fusión de varias de ellas debido a que la muestra de imágenes necesarias para llevar a cabo un entrenamiento decente sería muy alta y la anotación de estas consumiría la mayor parte del tiempo del proyecto.

5.1. División de la base de datos en subsets

A la hora de dividir un base de datos para entrenar una red neuronal hay que tener en cuenta si esta está equilibrada o no. Como se puede ver en el Cuadro 2, ContaiNET no tiene el mismo número de instancias para cada clase, por lo que no se debe hacer una división aleatoria de los datos. Si se dividieran de manera aleatoria, podría suceder que la mayor parte de las instancias de vidrio para cosméticos aparezcan en la división de test y entonces la red nunca sea entrenada o validada con ejemplos de cosméticos, lo que sesgaría los resultados. La distribución de los datos debe ser entonces estratificada.

Una distribución estratificada de los datos se refiere a cuando todas las clases se dividen en la misma proporción para entrenamiento, validación y test [20]. Para el presente TFG se ha decidido que la proporción sea 70/20/10 (*train/val/test*). Esto significa que de cada clase, un 70 % de las instancias irá para el entrenamiento de la red, un 20 % para la validación y un 10 % para testear el rendimiento con una muestra no utilizada durante el entrenamiento.

Para la división de los datos en diferentes subsets, se ha utilizado un fichero de Python al que se le introducen las anotaciones exportadas en formato COCO (un único archivo con extensión `.json` que contiene todas las anotaciones de todas las imágenes) y los porcentajes deseados para las subcategorías de la base de datos. Este código entonces realiza una distribución estratificada de las anotaciones de cada clase. El código utilizado se encuentra adjuntado en el Anexo I de esta memoria, que ha sido obtenido de un repositorio de Github que permite su uso gratuito [21].

5.2. Entrenamientos con YOLOv8n.pt

En este apartado se van a llevar a cabo entrenamientos de entrenamiento de la red neuronal preeentrenada YOLOv8 nano ya que es la que más rápido realiza las detecciones y es menos exigente con el equipo informático a la hora de entrenar.

Estos entrenamientos se van a diferenciar en el número de épocas que duran, a fin de determinar cual el tiempo de entrenamiento necesario para no obtener un sistema poco entrenado (*underfitted*) pero tampoco sobre entrenado (*overfitted*).

Los experimentos son los siguientes:

- **Experimento 1:** Entrenamiento durante 50 épocas
- **Experimento 2:** Entrenamiento durante 100 épocas
- **Experimento 3:** Entrenamiento durante 200 épocas

El proceso seguido para entrenar las redes con la base de datos de ContaiNET se explica a continuación:

1. Primero se han determinado los hiperparámetros para el entrenamiento:
    - `epochs:` Se ha cambiado para cada experimento. Se ha empezado con 50, luego 100 y finalmente 200.
    - `patience:` Para el primer experimento se ha puesto una paciencia de 10 epochs. Para el segundo y el tercero se ha dejado en 20.
    - `batch:` Se ha seleccionado que se ajuste automáticamente (-1), lo que ha resultado en una *batch size* de 16.
    - `lr0:` Se ha seleccionado un *learning rate* inicial de 0.01, ya que es considerado un optimizador del entrenamiento en la mayoría de los casos.
    - `imgsz:` Debido a que las imágenes descargadas de TrashNET ya están escaladas a 512 píxeles y estas conforman cerca de la mitad de ContaiNET, se ha elegido este valor para que YOLOv8 solo tenga que escalar el resto de imágenes de la base de datos y también para que el entrenamiento no sea muy largo.
2. Se procede al entrenamiento de la red neuronal combinada con validación tras cada época para que el entrenamiento sea más eficiente. Esto también permite realizar un seguimiento más detallado mientras se espera a que el entrenamiento termine. En esta etapa, YOLOv8 genera imágenes en la carpeta de resultados con tandas de imágenes que usa para el entrenamiento, a las que aplica técnicas de *data augmentation* para sacar más partido a la base de datos. Se puede ver un ejemplo en la Figura 21.

3. Tras acabarse el número de épocas establecido, se obtienen dos archivos con los pesos. Uno con los pesos de la última época (`last.pt`) y otro con los pesos que se considera que proporcionan un mejor rendimiento (`best.pt`). En la mayoría de los casos de entrenamiento durante un número de épocas bajo, la última época es la que genera los mejores resultados, por lo que ambos archivos terminan siendo iguales.

4. Finalmente, para finalizar el experimento y obtener métricas del rendimiento de la red ya entrenada con las imágenes destinadas al test. Se utiliza el modo de validación (*val*) de YOLO, pero se utiliza el argumento `split = 'test'` para especificar que se use el subset de test.

5. Además, para un ejemplo más visual del rendimiento de la red entrenada y validada en imágenes individuales, se utiliza el modo de predicción (*predict*) para obtener una imagen con las cajas delimitadoras predichas con un grado de confianza mayor al umbral que se quiera, en este caso se ha decidido que sea mayor a 0.7.

Figura 21: Tanda de imágenes creada por YOLOv8 para el entrenamiento

6. Resultados

6.1. Resultados del entrenamiento durante 50 épocas

El proceso de entrenamiento y validación de la red durante 50 épocas duró un total de 17 horas y 4 minutos.

En la Figura 22 se observa la curva Precision-Recall tras la validación con imágenes utilizadas durante el entrenamiento de la red neuronal y en la Figura 23, funciones que describen las mejoras durante el entrenamiento y validación:

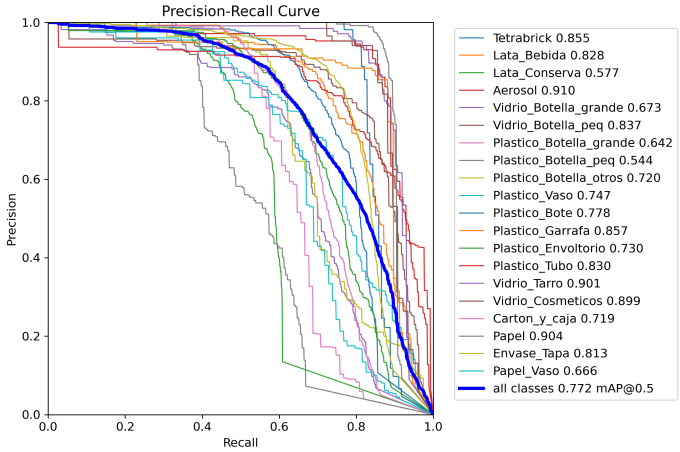


Figura 22: Curva P-R de la validación para un entrenamiento de 50 épocas.

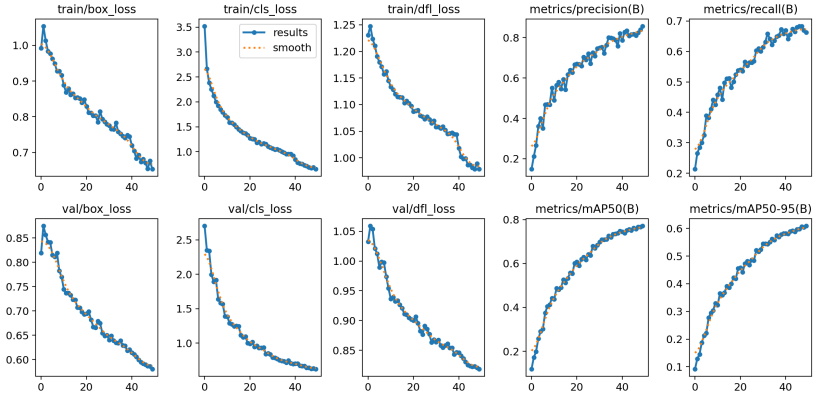


Figura 23: Métricas del entrenamiento y validación durante 50 épocas

Las funciones de pérdidas en 23 son descendientes lo cual es bueno. En cuanto a las de precisión, *recall* y mAP, también es positivo que sean ascendientes. La precisión media de todas las clases (0.772) es un valor bastante bueno teniendo en cuenta que se ha entrenado solo durante 50 épocas.

En las Cuadros 3 y 4 se recogen los resultados después de entrenar la red y después del test, respectivamente:

Nombre de clase	**Precisión**	**Recall**	**mAP50**
Tetrabrick	0.918	0.813	0.855
Lata de bebida	0.855	0.700	0.828
Lata de conserva	0.849	0.466	0.582
Aerosol	0.908	0.850	0.912
Botella de plástico grande (bebida)	0.779	0.569	0.642
Botella de plástico pequeña (bebida)	0.975	0.338	0.544
Botella de plástico (otros usos)	0.646	0.641	0.720
Garrafa de plástico	0.877	0.845	0.857
Bote de plástico	0.929	0.603	0.778
Tubo de plástico	0.848	0.706	0.832
Envoltorio de plástico	0.886	0.541	0.731
Vaso de plástico	0.672	0.722	0.747
Botella de vidrio grande	0.565	0.689	0.674
Botella de vidrio pequeña	0.859	0.727	0.841
Tarro de vidrio	0.953	0.818	0.902
Vidrio para cosméticos	0.964	0.737	0.896
Cajas y cartón	0.881	0.569	0.720
Papel	0.970	0.850	0.904
Vaso de papel	0.889	0.447	0.663
Tapa de envase	0.937	0.635	0.813
Media total de las clases	**0.858**	**0.663**	**0.772**

Cuadro 3: Resultados del primer entrenamiento y validación durante 50 épocas

Aquí se puede ver en más detalle que aunque la precisión media de las clases es bastante alta (0.772), hay clases como las botellas de plástico pequeñas, latas de conserva y vasos de papel que no tienen precisiones medias intraclase tan buenas.

Nombre de clase	**Precisión**	**Recall**	**mAP50**
Tetrabrick	0.670	0.605	0.685
Lata de bebida	0.795	0.659	0.772
Lata de conserva	0.794	0.891	0.921
Aerosol	0.844	0.852	0.863
Botella de plástico grande (bebida)	0.686	0.383	0.421
Botella de plástico pequeña (bebida)	0.897	0.420	0.563
Botella de plástico (otros usos)	0.654	0.719	0.762
Garrafa de plástico	0.729	0.603	0.659
Bote de plástico	0.873	0.592	0.739
Tubo de plástico	0.833	0.818	0.887
Envoltorio de plástico	0.826	0.600	0.745
Vaso de plástico	0.495	0.637	0.600
Botella de vidrio grande	0.544	0.756	0.732
Botella de vidrio pequeña	0.839	0.800	0.837
Tarro de vidrio	0.908	0.778	0.855
Vidrio para cosméticos	0.912	0.828	0.910
Cajas y cartón	0.848	0.640	0.741
Papel	0.948	0.879	0.907
Vaso de papel	0.551	0.366	0.410
Tapa de envase	0.903	0.606	0.761
Media total de las clases	**0.777**	**0.672**	**0.739**

Cuadro 4: Resultados del test del primer experimento

Tras el test, lo natural es que los resultados bajen un poco debido a que se está evaluando el modelo con una serie de imágenes que nunca ha sido vista por la red neuronal. Se puede apreciar en la precisión y mAP50 total de las clases que son menores durante la validación (de 0.858 a 0.777 y de 0.772 a 0.739, respectivamente). Igualmente, los resultados del test siguen resultando buenos considerando el tiempo de entrenamiento.

En la Figura 24 se observa la matriz de confusión tras el test que parece correcta como comentamos en el apartado que explicaba como funciona esta métrica. En la Figura 25, las cajas predichas por la red con una de las imágenes de test.

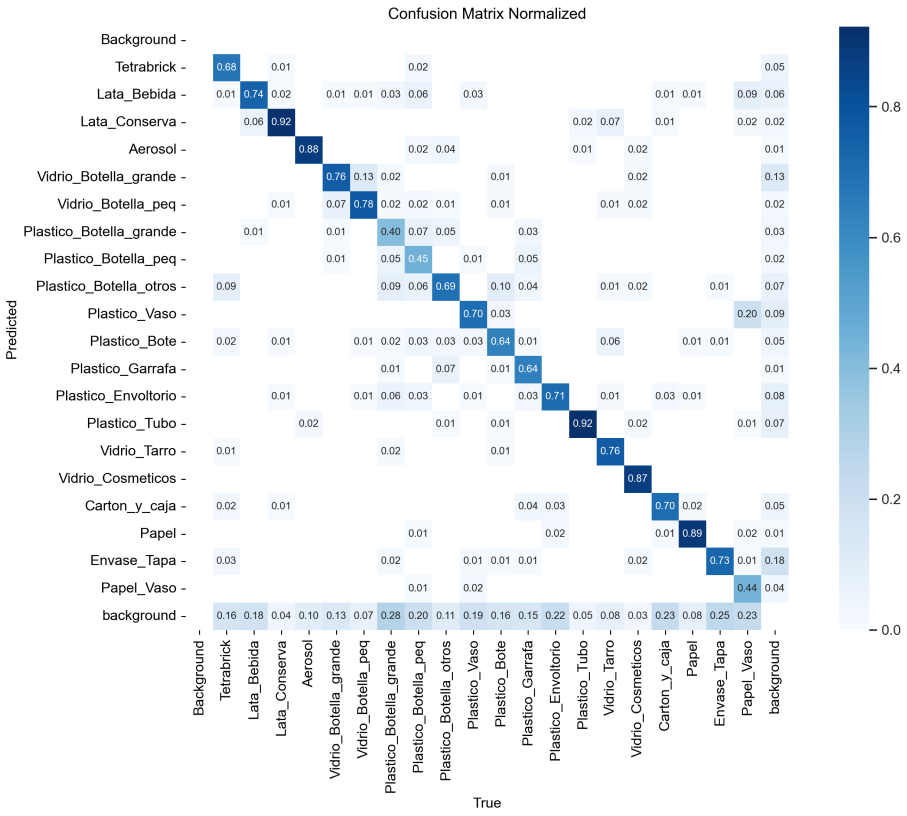


Figura 24: Matriz de confusión tras el test para un entrenamiento de 50 épocas

Figura 25: Detección para una de las imágenes de test tras 50 épocas

6.2. Resultados del entrenamiento durante 100 épocas

El proceso de entrenamiento y validación de la red durante 50 épocas duró un total de 37 horas y 6 minutos.

En la Figura 26 se observa la curva Precision-Recall tras la validación con imágenes utilizadas durante el entrenamiento de la red neuronal y en la Figura 27, funciones que describen las mejoras durante el segundo entrenamiento y validación:

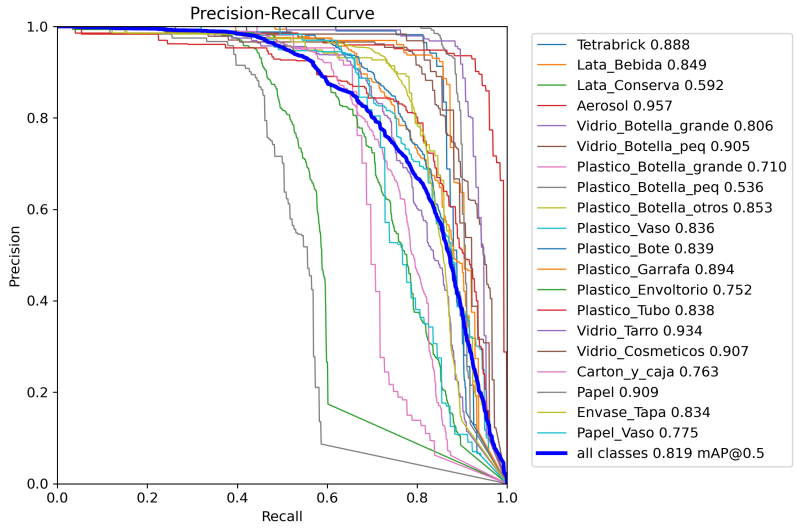


Figura 26: Curva P-R de la validación para un entrenamiento de 100 épocas

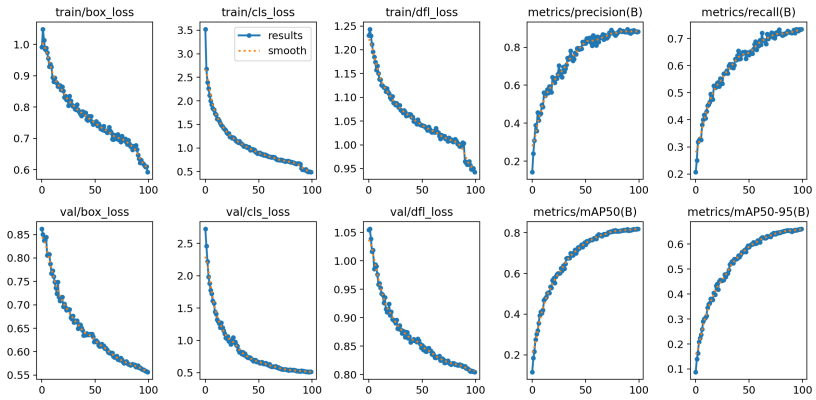


Figura 27: Métricas del entrenamiento y validación durante 100 épocas.

En este caso parece que la precisión media de las clases ha subido con respecto al experimento anterior (de 0.772 a 0.819). Además, al que en el experimento anterior, parece que el entrenamiento ha ido correctamente ya que en la Figura 27 se puede ver que las funciones de pérdidas son descendientes, mientras las de precisión, *recall* y mAP crecen.

En las Cuadros 5 y 6 se recogen los resultados después de entrenar la red y después del test, respectivamente:

Nombre de clase	**Precisión**	**Recall**	**mAP50**
Tetrabrick	0.934	0.839	0.889
Lata de bebida	0.877	0.709	0.849
Lata de conserva	0.885	0.460	0.586
Aerosol	0.892	0.945	0.957
Botella de plástico grande (bebida)	0.845	0.667	0.711
Botella de plástico pequeña (bebida)	0.914	0.413	0.537
Botella de plástico (otros usos)	0.896	0.773	0.852
Garrafa de plástico	0.886	0.861	0.894
Bote de plástico	0.935	0.660	0.838
Tubo de plástico	0.833	0.767	0.841
Envoltorio de plástico	0.849	0.619	0.753
Vaso de plástico	0.709	0.815	0.837
Botella de vidrio grande	0.735	0.751	0.806
Botella de vidrio pequeña	0.899	0.810	0.907
Tarro de vidrio	0.968	0.881	0.933
Vidrio para cosméticos	0.956	0.816	0.913
Cajas y cartón	0.837	0.658	0.763
Papel	0.935	0.877	0.909
Vaso de papel	0.942	0.634	0.775
Tapa de envase	0.938	0.720	0.834
Media total de las clases	**0.883**	**0.734**	**0.819**

Cuadro 5: Resultados del segundo entrenamiento y validación durante 100 épocas

Tras 100 épocas la mayoría de clases parecen haber mejorado sus resultados, lo cual ha subido la media total de las clases. Sin embargo, en el caso de las botellas de plástico pequeñas, los resultados actuales son peores que antes (de 0.544 a 0.537).

Nombre de clase	**Precisión**	**Recall**	**mAP50**
Tetrabrick	0.773	0.580	0.689
Lata de bebida	0.858	0.687	0.801
Lata de conserva	0.842	0.927	0.941
Aerosol	0.822	0.921	0.902
Botella de plástico grande (bebida)	0.633	0.531	0.517
Botella de plástico pequeña (bebida)	0.804	0.494	0.594
Botella de plástico (otros usos)	0.861	0.818	0.844
Garrafa de plástico	0.861	0.718	0.758
Bote de plástico	0.902	0.629	0.813
Tubo de plástico	0.936	0.838	0.926
Envoltorio de plástico	0.836	0.638	0.762
Vaso de plástico	0.673	0.594	0.646
Botella de vidrio grande	0.764	0.794	0.817
Botella de vidrio pequeña	0.898	0.836	0.910
Tarro de vidrio	0.913	0.854	0.915
Vidrio para cosméticos	0.922	0.857	0.913
Cajas y cartón	0.824	0.678	0.767
Papel	0.938	0.903	0.922
Vaso de papel	0.775	0.365	0.496
Tapa de envase	0.914	0.647	0.787
Media total de las clases	**0.839**	**0.715**	**0.786**

Cuadro 6: Resultados del test del segundo experimento

Como era de esperar, los resultados han vuelto a bajar en el test con respecto a los de la validación. Sin embargo, estos siguen siendo mejores que los del experimento anterior. El caso de la botella de plástico pequeña que parecía haber empeorado durante el último entrenamiento, obtiene en la fase de test mejores resultados todavía (0.594).

En la Figura 28 se observa la matriz de confusión tras el test con imágenes no

utilizadas durante el entrenamiento y en la Figura 25, las cajas predichas por la red con una de las imágenes de test.

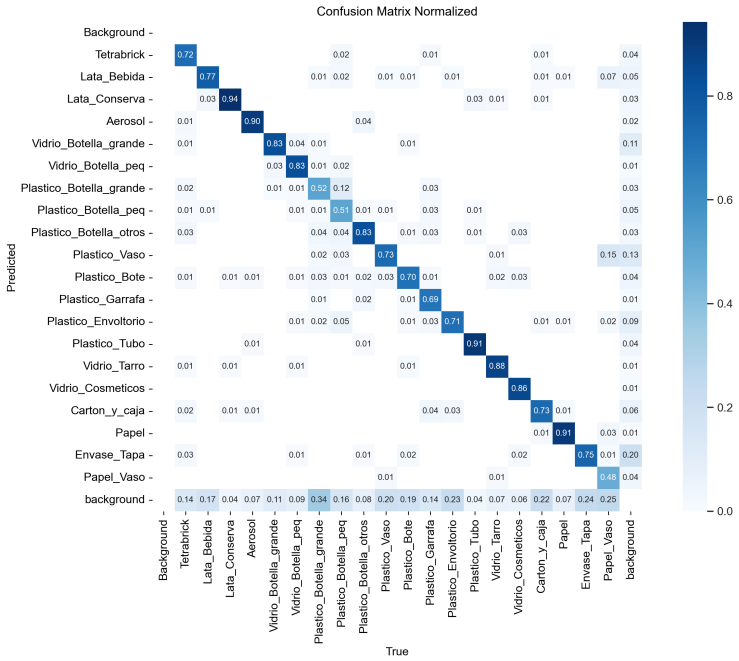


Figura 28: Matriz de confusión tras el test para un entrenamiento de 100 épocas

Figura 29: Detección para una de las imágenes de test tras 100 épocas

6.3. Resultados del entrenamiento durante 200 épocas

Este entrenamiento no ha dado resultados ya que el parámetro `patience = 20` ha cortado el entrenamiento cuando ha visto que el entrenamiento no seguía mejorando.

7. Discusión

En este apartado se van a comparar los resultados de los tres experimentos, así como comentar posibles razones por las que estos se puedan haber visto afectados.

	Precisión	**Recall**	**mAP50**
Experimento 1	0.777	0.672	0.739
Experimento 2	0.839	0.715	0.786

Cuadro 7: Resultados totales del test de los Experimentos 1 y 2

Los resultados para el primer experimento, entrenando durante 50 épocas, aunque con espacio para mejorar, parecen indicar que la red ha sido entrenada correctamente. La curva PR del entrenamiento (Figura 22) muestra como la tendencia general del modelo se aproxima a la ideal pero las individuales para cada clase tienen resultados dispares. En algunos casos como el del papel, se consigue una precisión muy alta que se mantiene con sensibilidades de hasta 0.8, sin embargo en otros caso como las latas de conserva o las botellas de plástico pequeñas, la curva está muy alejada de la ideal.

Tras el test se ve que aunque bajan un poco los resultados medios, se mantienen altos. Para dar por finalizado el test, se puede ver en la Figura 25 un ejemplo visual de predicción del modelo sobre una imagen en la que la mayoría de los elementos han sido identificados, y aquellos que han sido identificados, lo han sido correctamente. Se puede considerar que este experimento aunque todavía puede mejorar, da resultados competitivos.

En cuanto al segundo experimento, inicialmente parece que la red se beneficia de entrenarse durante un periodo de tiempo más largo, ya que el hiperparámetro *patience* no produce ningún corte forzoso del entrenamiento. Como se ha mencionado anteriormente, aunque la mAP global de este segundo entrenamiento es mejor que en el caso del primer experimento, se ha producido una bajada de la precisión para algunas clases como las botellas de plástico pequeñas que han pasado de 0.544 mPA50 después de 50 épocas a 0.537 tras 100 épocas. Aunque esta bajada no sea muy pronunciada a simple vista, hay que tener en cuenta que el tiempo de entrenamiento ha sido del doble y aún así no solo no ha mejorado sino que ha decrecido.

Los resultados del test del segundo experimento vuelven a traer consigo una pequeña bajada de en cuanto al rendimiento de la validación del entrenamiento pero esto ya hemos establecido que es natural. Los valores obtenidos confirman lo que se intuía al evaluar el entrenamiento durante 100 épocas, y es que en efecto el modelo se beneficia en su mayor parte de ser entrenado durante más tiempo. En el caso particular anteriormente mencionado de las botellas de plástico pequeñas, parece ser que tras el test,

su mAP mejora con respecto a la validación del entrenamiento. Esto, aunque puede ser positivo, también podría ser un indicador de un sesgo en la muestra de test, en la que las imágenes de esta clase no son tan variadas como en el caso de la muestra del entrenamiento y validación. Para finalizar el segundo experimento, se ve que la imagen con las predicciones del modelo (Figura 29) consigue identificar correctamente más envases que las predicciones del experimento anterior (botes de plástico del frente y de la derecha), sin embargo los índices de confianza son menores en algunos casos en comparación con este primer experimento y una de las detecciones que antes había sido capaz de predecir correctamente, ya no es detectada (tarro de vidrio de la izquierda).

Finalmente, durante el entrenamiento del tercer experimento se han producido dos paradas forzosas debido a que el entrenamiento no mejoraba de manera significante después de 20 épocas. Las razones para esto podrían indicar que no es conveniente entrenar la red durante tanto tiempo con una base de datos de estas características ya que esto puede ocasionar *overfitting*. Para entrenar la red durante más tiempo sería interesante aumentar la base de datos con muchas más imágenes de envases. El proceso de entrenamiento ha sido cortado además tras la época 121, lo que indica que los mejores resultados no han variado considerablemente desde alrededor de la época 100, como es en el caso del experimento 2.

Es importante también conocer las limitaciones en cuanto a las detecciones incorrectas de clases de envases como se puede ver en la matrices de confusión de los dos primeros experimentos en las Figuras 24 y 28. Durante el etiquetado de las imágenes se ha detectado que existen envases que pertenecen a clases diferentes que sin embargo son muy parecidos en cuanto a apariencia. Este es el caso de algunos envases de plástico que en su diseño buscan asemejarse lo más posible a envases de vidrio, como algunas botellas de plástico de aceite o botes de plástico de mayonesa que parecen de vidrio a simple vista. Estos errores en la detección son altamente complicados de evitar para una red neuronal para detección de objetos actual incluso si se utilizara una base de datos más extensa.

8. Conclusiones y trabajos futuros:

8.1. Conclusiones:

Para concluir el presente TFG se puede afirmar que las **redes neuronales** demuestran ser de gran utilidad en problemas de detección de envases de diferentes formas y materiales, dada la complejidad de estos. Además, en el caso particular que se ha estudiado, se concluye que para la base de datos ContaiNET, el número ideal para el entrenamiento de la red ronda las 100 épocas, ya que entrenarla durante menos ciclos da un rendimiento menor y entrenarla durante un tiempo mayor termina siendo contraproducente para el tamaño de la muestra.

Los resultados obtenidos se consideran buenos, ya que con 100 épocas se ha conseguido rozar la marca de 0.8 (mAP50). Sin embargo, estos se podrían mejorar para obtener resultados más precisos, con una base de datos de imágenes más grandes, por ejemplo, como se ha sugerido en el apartado de discusiones. De cualquier manera, se ha aprendido a elaborar una base de datos con sus correspondientes anotaciones, pudiendo aprender de posibles errores cometidos durante la confección de esta.

Asimismo, este proyecto ha servido para comprender mucho mejor la situación actual en la que nos encontramos en material de redes neuronales para la detección de objetos, introduciéndonos de cabeza en el entrenamiento de una de los modelos más recientes hasta la fecha.

Es importante destacar que la documentación disponible hasta la fecha sobre redes neuronales y trabajos similares es muy limitada. Sobre todo cuando hablamos de Yolov8 de Ultralytics, que fue lanzada a principios de 2023 y su documentación se ha ido ampliando incluso durante el desarrollo de este TFG. También se ha notado una gran escasez en cuanto a trabajos similares en el ámbito de redes neuronales para la detección de objetos, siendo los enfocados hacia la diferenciación de envases prácticamente inexistentes.

8.2. Trabajos futuros:

En este aparado se valoran posibles mejoras que pueden ser aplicadas al desarrollo de este proyecto o similares en el futuro, a la vez que se consideran brevemente posibles aplicaciones que se le puede dar a un producto como este en el mercado actual:

8.2.1. Posibles mejoras:

Las mejoras en cuanto a este proyecto están divididas en dos partes: las mejoras directas que se podrían implementar para continuar y mejorar los resultados de los entrenamientos de las redes de YOLOv8, y las posibles mejoras destinadas a ampliar el

proyecto hacia aplicaciones específicas que requieran combinar tecnologías para mejorar los resultados:

- **Mejoras directas:**

  Utilización de modelos preentrenados superiores de YOLOv8 como YOLOv8x.pt que mejoran considerablemente en cuanto a precisión, aunque son más demandantes en cuanto a tiempo y equipo informático.

  Aumentar la base de datos con al menos 1000 imágenes por clase y mayor número de clases para poder realizar una detección de envases más específica, que permita a la red a encontrar los patrones intraclase más fácilmente.

- **Mejoras para ampliar:**

  Como se ha mencionado al final de la discusión de los resultados, es importante conocer las limitaciones de las redes neuronales para la detección de objetos en imágenes, ya que en algunos casos estas no van a poder hacer buenas detecciones entre envases parecidos pero de materiales diferentes. Al menos no sin integrar otros tipos de sensores que puedan complementar aquella información que el detector de objetos no puede extraer de imágenes y vídeos. Estos pueden sensores de densidad que permiten diferenciar mejor entre materiales diferentes.

8.2.2. Posibles aplicaciones:

Las posibles aplicaciones para este tipo de software son muy variadas.

Como se ha mencionado en el apartado anterior, si un detector de objetos de estas características es combinado con otro tipo de sensores que permitan una detección más precisa al usar la información de estos, este modelo tiene la versatilidad de poder ser utilizado en grandes plantas de reciclaje donde complementar los algoritmos ya presentes para la detección de basura mal diferenciada.

Otra posible aplicación es su integración en las conocidas máquinas de *Reverse Vending*, como se ha mencionado en la motivación de este TFG. Estas se podrían beneficiar de software de visión por computador para reducir las limitaciones de uso que tienen en la actualidad. En el Anexo III de esta memoria se encuentra una propuesta de *Business Plan* en inglés para una de estas máquinas que integra redes neuronales, así como una serie de planos de una versión más simple de la máquina enfocada a la detección y almacenamiento de botellas de vidrio grandes, medianas y pequeñas. Estos documentos fueron entregados en su momento como trabajos de clase durante el presente curso académico 2022/2023 y sirvieron como el germen de la idea de este Trabajo de Fin de Grado.

Referencias

[1] M. McClure. *Everything you should know about single-use plastic*. 2021. URL: https://www.greenpeace.org/africa/en/blogs/14052/everything-you-should-know-about-single-use-plastic/#:~:text=Disposable%5C%20plastic%5C%20items%5C%20don%5C%27t,sources%5C%20and%5C%20even%5C%20our%5C%20food. [Consultado en: 10/04/2023].

[2] ONU. *Objetivos de Desarrollo Sostenible*. 2021. URL: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. [Consultado en: 10/04/2023].

[3] OECD. *Measuring distance to the SDG targets: Country profile - Spain*. URL: https://www.oecd.org/wise/measuring-distance-to-the-SDG-targets-country-profile-Spain.pdf.

[4] Dansk Retursystem A/S. *Sustainability of the Danish Return System*. URL: https://danskretursystem.dk/en/sustainability/. [Consultado en: 12/04/2023].

[5] MIT Sloan School of Management. *Machine Learning Explained*. Ideas Made to Matter. URL: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained.

[6] Aprende Machine Learning. *Modelos de Detección de Objetos*. URL: https://www.aprendemachinelearning.com/modelos-de-deteccion-de-objetos. [Consultado en: 05/07/2023].

[7] R. Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[8] K. Sambasivarao. "Non-maximum Suppresion (NMS)". En: *Towards Data Science* (2019). URL: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c. [Consultado en: 14/07/2023].

[9] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". En: (2015), págs. 91-99. DOI: 10.1109/ICCV.2015.16. URL: https://arxiv.org/pdf/1506.01497.pdf.

[10] Alec Radford, Luke Metz y Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". En: *ArXiv preprint arXiv:1506.02640* (2015).

[11] Ultralytics. *YOLOv8*. 2023. URL: https://docs.ultralytics.com/. [Consultado en: 27/03/2023].

[12] Glenn Jocher. *True Meaning of Argument imgsz?* 2023. URL: https://github.com/ultralytics/ultralytics/issues/2546.

[13] S. Rozada Raneros. *Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning*. 2021. URL: https://uvadoc.uva.es/bitstream/handle/10324/45359/TFM-G1316.pdf.

[14] E. Rico Guardiola. *Diseño, implementación y evaluación de técnicas genéricas de detección de C. elegans mediante redes neuronales convolucionales*. Riunet. 2022. URL: https://riunet.upv.es/bitstream/handle/10251/187031/Rico%20-%20Diseno%20implementacion%20y%20evaluacion%20de%20tecnicas%20genericas%20de%20deteccion%20de%20C%20elegans%20mediant....pdf?sequence=1&isAllowed=y.

[15] G. O. Thung y J. Yang. *Classification of Trash for Recyclability Status*. Standford University, 2016. URL: https://cs229.stanford.edu/proj2016/report/ThungYang-ClassificationOfTrashForRecyclabilityStatus-report.pdf. [Consultado en: 09/04/2023].

[16] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[17] User: RangeKing. *Brief Summary of YOLOv8 Model Structure*. 2023. URL: https://github.com/ultralytics/ultralytics/issues/189.

[18] Ultralytics. *YOLOv8 Object Detection Datasets Overview*. 2023. URL: https://docs.ultralytics.com/datasets/detect/. [Consultado en: 24/05/2023].

[19] S. Hossain. "Dual Focal Loss to address class imbalance in semantic segmentation". En: *Neurocomputing* 462 (2021), págs. 69-87. URL: https://www.sciencedirect.com/science/article/abs/pii/S0925231221011310.

[20] A. Igareta. *Stratified Sampling: You May Have Been Splitting Your Dataset All Wrong*. Towards Data Science. 2021. URL: https://towardsdatascience.com/stratified-sampling-you-may-have-been-splitting-your-dataset-all-wrong-8cfdd0d32502.

[21] PyLabel Project. *Dataset Splitting Repository in Github*. GitHub repository. 2021. URL: https://github.com/pylabel-project/samples/blob/main/dataset_splitting.ipynb.

Anexo I:

En este anexo a la memoria se pueden encontrar los códigos utilizados:

- Código de Python para descargar imágenes de Google imágenes a partir de palabras clave.

```python
from simple_image_download import simple_image_download as simp

response = simp.simple_image_download

keywords = ["glass jar HD", "paper wrapper HD", "small plastic water bottle HD"] #Declare the keywords of the prompts for download

for kw in keywords:
response().download(kw, 50) #Download the number of images specified for every keyword.
```

- Código de Python para dividir la base de datos en subsets para entrenamiento, validación y test con una distribución estratificada de las clases.

```python
import os, zipfile
import logging
from pylabel import importer

#Specify path to the .json file
path_to_annotations = "/Users/marcos/Desktop/train.json"
#Specify the path to the images
path_to_images = "/Users/marcos/Desktop/Data_completa/images"

#Import the dataset into the pylable schema
dataset = importer.ImportCoco(path_to_annotations, path_to_images=path_to_images, name="ContaiNET")
dataset.df.head(5)

#This gives back the number of images,classes and instances per class
print(f"Number of images: {dataset.analyze.num_images}")
print(f"Number of classes: {dataset.analyze.num_classes}")
print(f"Classes:{dataset.analyze.classes}")
print(f"Class counts:\n{dataset.analyze.class_counts}")

#This splits the dataset in a Stratified distribution:
dataset.splitter.StratifiedGroupShuffleSplit(train_pct=.7, val_pct=.2, test_pct=.1, batch_size=1)

#This saves the split dataset into different folders for train, val and test and provides a .yaml file with the classes.
dataset.export.ExportToYoloV5(output_path='/Users/marcos/Desktop/Data_completa/model_training/labels',yaml_file='dataset.yaml', copy_images=True, use_splits=True)
```

- Entrenamiento, validación y test con YOLOv8

```
from ultralytics import YOLO

#First Experiment
model = YOLO('yolov8n.yaml').load('yolov8n.pt') #Load a pretrained
    model and transfer its weights

model.train(data= '/Users/marcos/Desktop/TrabajoFindeGrado/code/data.
    yaml', epochs=50, imgsz=512, batch=-1, patience=10, val=True)

metrics = model.val(data= '/Users/marcos/Desktop/TrabajoFindeGrado/
    code/data.yaml', imgsz= 512) # validate the model's performance
metrics.box.map50  # map50

model = YOLO('runs/detect/val/best.pt') #Selecting the weights after
    training
results = model.val(data= '/Users/marcos/Desktop/TrabajoFindeGrado/
    code/data.yaml', imgsz= 512, split='test') #test the model

#Second Experiment
model = YOLO('yolov8n.yaml').load('yolov8n.pt') #Load a pretrained
    model and transfer its weights

model.train(data= '/Users/marcos/Desktop/TrabajoFindeGrado/code/data.
    yaml', epochs=100, imgsz=512, batch=-1, patience=20, val=True)

metrics = model.val(data= '/Users/marcos/Desktop/TrabajoFindeGrado/
    code/data.yaml', imgsz= 512) # validate the model's performance
metrics.box.map50  # map50

model = YOLO('runs/detect/val/best.pt') #Selecting the weights after
    training
results = model.val(data= '/Users/marcos/Desktop/TrabajoFindeGrado/
    code/data.yaml', imgsz= 512, split='test') #test the model

#Third Experiment
model = YOLO('yolov8n.yaml').load('yolov8n.pt') #Load a pretrained
    model and transfer its weights

model.train(data= '/Users/marcos/Desktop/TrabajoFindeGrado/code/data.
    yaml', epochs=200, imgsz=512, batch=-1, patience=20, val=True)

metrics = model.val(data= '/Users/marcos/Desktop/TrabajoFindeGrado/
    code/data.yaml', imgsz= 512) # validate the model's performance
metrics.box.map50  # map50

model = YOLO('runs/detect/val/best.pt') #Selecting the weights after
    training
results = model.val(data= '/Users/marcos/Desktop/TrabajoFindeGrado/
    code/data.yaml', imgsz= 512, split='test') #test the model
```

- Archivo `data.yaml`

```yaml
path: /Users/marcos/Desktop/TrabajoFindeGrado/code/data #dataset root directory
train: images/train # train images (relative to "path")
val: images/val     # validation images (relative to "path")
test: images/test   # test images (relative to "path")

# Classes
names:
  0: Background
  1: Tetrabrick
  2: Lata_Bebida
  3: Lata_Conserva
  4: Aerosol
  5: Vidrio_Botella_grande
  6: Vidrio_Botella_peq
  7: Plastico_Botella_grande
  8: Plastico_Botella_peq
  9: Plastico_Botella_otros
  10: Plastico_Vaso
  11: Plastico_Bote
  12: Plastico_Garrafa
  13: Plastico_Envoltorio
  14: Plastico_Tubo
  15: Vidrio_Tarro
  16: Vidrio_Cosmeticos
  17: Carton_y_caja
  18: Papel
  19: Envase_Tapa
  20: Papel_Vaso
```

Anexo II:

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

Cuadro 8: Grado de relación del TFG con los ODS.

De los objetivos con alto grado de relación con este TFG, las metas específicas que los relacionan son:

- ODS 12: Metas 12.4 y 12.5
- ODS 13: Metas 13.2 y 13.3
- ODS 14: Meta 14.1

Anexo III: Business Plan

Los contenidos del presente anexo no son en sí parte de la memoria de este TFG, sino proyectos de asignaturas cursadas durante el Grado de Ingeniería Electrónica Industrial y Automática que han servido como idea precursora para el desarrollo de este proyecto.

Se adjuntan en este anexo a modo de primer borrador de una posible aplicación de las redes neuronales entrenadas para este TFG.

Business Plan: DvuelB - Machine for the repurpose of containers for a circular economy.

Table of Contents:

1. Introduction of the business idea:

The situation the world is in nowadays regarding single-use containers should not come through as a surprise. Even though these containers are often branded as recyclable, they are actually designed to be discarded after one use. The alarming truth is that around 90 % of all single-use plastic containers that are generated have never been recycled [1]. This is in part because of poorly organized recycling systems that blame and shame the average citizen for not recycling correctly when they are in fact not help nor encouraged to do so.

The U.N. has proposed a series of goals to be met by the end of the decade known as the **Sustainable Development Goals** that aim to minimize the effects of climate change among other things. The three goals related to waste management and pollution reduction are goals number 12, 13 and 14 ("Responsible Consumption and Production", "Climate Action" and "Life below water" respectively). The most recent report from 2022 shows that even though Spain is making an effort to meet the sustainability targets, these efforts are not enough [3].

However, some countries like Denmark and Sweden have decided to support the integration of systems that encourage their citizens to be involved in the recycling of waste containers from everyday life activities, by rewarding them immediately after their correct disposal of this waste. The **Danish Return System** is a perfect example of this implementation, where the government applies special taxes to various types of bottles and cans that can be refunded by depositing these back into the custody of supermarkets and grocery stores that sell them. In this way, the Danish government has been able to recycle up to 92 % of all bottles and cans in circulation [4].

Finally, Spain has decided to jump on board along these European countries to try to keep in circulation the materials used in the manufacturing of plastic containers, glass bottles and cans. One of the many laws and one of the most recent ones is **RD 1055/2022** which states that distributors of packaged products are required to participate in deposit, return and refund systems for single-use packaging, as established in agreements with producer responsibility organizations **(Art. 43.b)** [22].

This recent development in the Spanish law and the current growth of the need to implement return systems like the ones seen in other places of the world, contribute to the birthing of this business idea. From **DvuelB**, we are determined to enter a market not yet developed in Spain, to make recycling a more interactive practice that involves customers by encouraging them to recycle but also fulfills a need for every establishment or company that generates these containers, as the law now states that they ought to participate in their deposit, return and refund system. All of this without taking into account the economic resources that these businesses can save by recycling used containers to produce their own productive materials and even reducing their carbon footprint by 49 % when using recycled TEP compared to non-recycled.

2. DvuelB - The product:

The product in question being offered to the market by **DvuelB** is a reverse vending machine equipped with a camera (for computer vision) that is capable or recognizing a wide range of different containers (plastic containers, cans and glass containers) in order to handle them as required and stored them awaiting for their proper treatment. For simplicity reasons, throughout the present document there will be a distinction between "client" and "user". The former will refer to the business or government that purchases the machine and the latter, to the person that makes use of the machine as an individual separated from the client.

The machine will be programmed to meet the specific requirements of the client. These requirements can be adjusting the range of containers accepted by the machine programmed for a specific business, as some of them will only want to accept the containers that they directly produce or with the same material components that they use in their production processes. This part of the design corresponds to the "software" portion of the product. Apart from this, there is also the "hardware" portion, which involves the design of the body of the machine and its electronic components.

2.1 Software:

As mentioned earlier in this business plan, the machine will work via computer vision. A camera (*In-Sight 2000-130C Mini*) will scan the container as a whole and communicate with a neural network trained specifically with a data base that contains the elements to be detected. If the container introduced matches any of the ones present in the data base, then it will be accepted by the machine and will store the amount of money accumulated from all the containers accepted during the operation. In the case the container is not supported by the data base, the machine will reject it, asking the user to remove it from the detection area.

The amount of money that corresponds to the type of container will also be chosen by the client. For example, common values would be 10 cents for small plastic bottles and cans, 15 cents for small glass bottles, 20 cents for big plastic bottles and 30 cents for big glass bottles. Once the container disposed is accepted by the computer vision camera, a screen on the front of the machine will display the money accumulated from returning containers and these ones would be stored according to their composition.

The screen that allows the interaction between the user and the machine will not only be able to display the total amount accumulated but also a menu for the user to choose when to start and end the operation, as well as the option to donate the money to a charity or exchange it for voucher for their next purchase. This voucher will also be able to be exchanged for cash inside the establishment of the client. To promote inclusivity, the machine will have accessibility features that allow voice commands for the visually impaired as well as instructions in braille.

2.2 Hardware:

The machine will integrate different electronic components:

The body of the machine can be adapted to the needs of the client and may undergo design changes to fit a specific aesthetic. However, the main electronic components will remain constant.

The screen that will serve as a interface between the user and the machine will be placed always in the front of the machine, next to the receipt printer and the opening to deposit the waste containers. In this opening, there is conveyor belt that transports the containers inside the machine if they have been approved by the camera. The camera is situated in the interior of the machine pointing to the conveyor belt next to the opening. In this way, when the container is introduced, the camera can quickly scan it and determined whether it is a valid container or not. When waste containers are accepted, the conveyor belt will transport them to the back where they will be separated into groups depending on their material. Glass containers will be stored first. However, cans and plastic containers will continue on the belt to be shredded into small pieces, maximizing storing space.

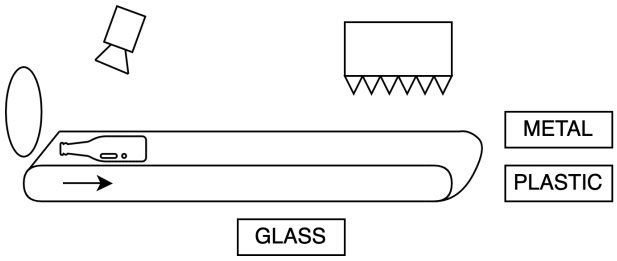


Figura 30: Schematics of the interior of the machine

2.3 Overall summary of the DvuelB machine:

In conclusion, the DvuelB machine is a reverse vending machine that is able to detect a wide variety of waste containers and store them according to their materials in a relatively small space compared to conventional dumpsters. Moreover, it is a highly flexible product in terms of personalizing it for the client (from the exterior design, to the data base of containers, the amount for the cash reward offered to the user as well as the size of the machine or storage capacity of it).

However, the main and most basic principle is the same for them all. A computer vision camera connected to a neural network that is capable of recognizing waste containers deposited by a user, and later dispose them according to the material of the container in different storage units, where they will await to be recycled (plastic, metal and glass) or set to undergo disinfection processes to be reused (sometimes glass).

3. Target market:

This product is very important and also vital in today's world. Now more than ever it is necessary to invest in a circular economy. By law [22], producers, distributors and sellers of containers are required to be responsible for the containers they have sold to the public. This product offers these businesses a solution to prove their commitment to the environment as well as to reduce their production expenses by recovering raw materials to produce new containers.

As DvuelB is a machine that can be personalized to the needs of our clients, that opens up our market to different types of possible buyers. Some of the clients that our business is going to target are:

- **Big supermarkets and small grocery stores:** These are the main sellers of containers to the public. As seen in other countries like Denmark and Germany, these are great possible clients for a business idea like this one of a reverse vending machine.
- **Waste management companies:** Their mission is to collect waste. Our reverse vending machine could be an interesting alternative to their current methods and could become the solution for areas that are difficult to access with big garbage trucks.
- **Public administration:** Both regional and local governments can seek to implement reverse vending machines in certain areas to promote recycling between their citizens. DvuelB could be implemented near ecoparks or even serve as a substitute to conventional dumpsters. In the context of Spain, these machines could be very useful even during national or regional festivities. After these festivities end, the streets are usually filled with empty cans and bottles, which could be avoided by setting reverse vending machines that encourage citizens to clean after themselves in order to get a reward.
- **Packaging companies:** By law these companies are also required to be responsible of the one-use containers that they generate. They may also be more interested in our product as this would be an easy alternative to obtain back raw materials that they can use to produce more packaging. Recycled TEP does not lose quality compared to non-recycled TEP and also reduces CO_2 emissions by 49 %, which benefits them.
- **Non-profit organizations:** These organizations can be interested in buying our product to set them in public spaces or events to promote recycling and sustainability.

4. Marketing plan:

Taking into account that in the context of Spain, reverse vending machines are not a product that is highly demanded nor offered, the first thing a company like **DvuelB Green S.L.** should do is to start promoting the benefits and unique advantages of purchasing a reverse vending machine and implementing it in a business.

In order to promote the DvuelB machine, a series of demonstrations will be done in supermarkets and other businesses that sell packaged goods. In this demonstrations, a DvuelB machine will be taken to the establishment to show in real time how it would work, pointing out the benefits that it can bring to their business as well as their need to be responsible of the waste containers sold by them, which is now mandatory by law and this machine could serve towards meeting that requirement.

Another marketing strategy to reach other possible clients other than grocery stores and supermarkets, would be to send information out to the sustainability and environment departments of local administrations, to let them know of our product and specific real life examples of where and when they should implement the reverse vending machine in their community to maximize their effect and raise awareness among their citizens.

To try to reach higher levels of the Spanish administration, from DvuelB S.L. we are going to present our product to public contests from the *Ministerio para la transición ecológica y el reto demográfico*. From there, we could also get hold of local administrations and grants that could help our business grow.

It is very important that when we approach any possible client, they understand that our product is flexible enough to meet their specific needs in terms of design and variety of supported containers. We offer a personalized experience for any of our clients and that of course will be reflected in the final price, being able to reduce this one if they purchase a larger amount of machines.

5. Personnel and Legal form:

The company **DvuelB Green S.L.** will be founded by four initial members: one electronic and automation engineer, one industrial design engineer, one mechanical engineer and one business and administration graduate. Each of these members contributes with 25.000€ at the beginning plus the additional investment of a business angel that will be investing 20.000€ in additional paid-in capital.

The total capital of DvuelB Green S.L. will ascend then to 120.000€ . Each share will cost 15 € , adding up to a total of 8000 shares owned by the shareholders.

Once the name of the company is decided (DvuelB Green S.L.) and the share capital is higher than the minimum required, the next step will be to draft and sign the articles of association of the company (*estatutos*), that contain the rights and obligation of the partners, the company's purpose, share capital and other vital aspects for the company. These articles must be signed before a notary public and entered into a public deed.

After the public deed is signed, the company will have to be registered in the *Registro Mercantil* in the province where the company is based, which in this case it will be set in Murcia, Spain.

In terms of employees, all four founding members will be working full time during the first years of the economic activity of the company and there will be no other full time workers in the company for this period. The profit for the shareholders will be the minimum during the first year, pending reevaluation after our business starts to mature a little more. Once the activity of the business starts to show improvement and it's reaching more clients and generating profit, more employees will be hired to answer to this growth in demand and allow the shareholders to step back and focus on improving the product and making administrative decisions.

6. Financial and economic plan:

At the beginning of the economic activity of DvuelB Green S.L., an initial investment must be made to be able to start producing and selling DvuelB machines. Taking into account the assets (both current and non-current) needed to do so, the following table has been made:

Initial Investment	
Concept	**Amount (€)**
Non-Current Assets	**86500**
I. Intangible Assets	5700
Software Licences	5700
II. Tangible Assets (Property, land and equipment)	80800
CNC Machine	25000
Computer equipment	8000
Office	10000
Warehouse	30000
Furniture	7800
Current Assets	**45000**
II. Inventories	45000
Raw Materials	45000
TOTAL ASSETS	**131500**

Cuadro 9: Initial investment needed to start our activity

With our initial capital we cannot afford to start our activity, so we would need to ask for external help to finance our activity. The difference between the starting capital of the company and the total assets needed is only of 11500, so the loan could be of around this value. However, it would be more beneficial to ask for a larger loan to be paid in the longer term, allowing more flexibility in case our company faces other short term liabilities.

Below in table 10, there are the orientative inflows and outflows for out company during a 10 year period with discount rate of 5 %. Considering an initial investment of 131500 € and that during the first few years of our activity we will no be selling

as many DvuelB machines, the operating inflows then will be lower than once the company has been up and running for a few years. Also, we are not taking into account any possible extraordinary inflows to consider scenarios not so ideal. However, we are considering additional investments in years 3 and 6 to account for the possible upgrade of some elements in our business.

	Inflows		**Outflows**					
Year	**Op.**	**Ext.**	**Op.**	**Ext.**	**Investment**	**Project c.f.**	**Disc. c.f.**	**Acc. disc. c.f.**
0					131500	-131500	-131500	-131500
1	70500	-	59200	-		11300	10762	-120738
2	75270	-	59200	-		16070	14576	-106162
3	110330	-	70250	-	9000	31080	26848	-79314
4	150030	-	132000	-		18030	14833	-64481
5	187400	-	132000	-		55400	43407	-21073
6	187400	-	132000	-	9000	46400	34624	13551
7	187400	-	132000	-		55400	39372	52923
8	187400	-	132000	-		55400	37497	90420
9	187400	-	132000	-		55400	35711	126131
10	187400	-	132000	-		55400	34011	160142

Cuadro 10: Estimated Inflows and Outflows for a 10 year period and 5 % discount rate

NOTE: Op.: Operational // Ext.: Extraordinary // c.f.: Cash flow // Disc.: Discounted // Acc.: Accumulated.

According with these estimations, it would seem that the initial investment could be obtained back around year 6. With this estimate, we obtain a profitability index of 1.22 (PI > 0) and a I.R.R. resulting in a value of 20.27 %. These are good indicators for our company as that means that there is still room for the discount rate to change without harming significantly our business.

The above inflows and outflows have been calculated by estimating the sales and costs of the activity of our company with the following information (which is subjected to change as it is purely orientative. It also should be noted that after the first five years, it has been considered that the activity will stabilize to a fixed operating inflow and outflow as part of our estimations (Tables 11 and 12)).

ANNUAL INCOME					
Concept	**Year 1**	**Year 2**	**Year 3**	**Year 4**	**Years 5-10**
Units sold	4	6	11	16	22
Base price	22000	20000	18000	14000	12000
Discount	-17500	-44730	-87670	-73970	-76600
TOTAL INCOME	**70500**	**75270**	**110330**	**150030**	**187400**

Cuadro 11: Estimated Annual Income for a 10 year period.

ANNUAL EXPENSES				
Concept	**Year 1**	**Year 2**	**Year 3**	**Years 4 - 10**
Supplies	**46955**	**47580**	**56254**	**102220**
Electronic components	46000	46500	55000	100000
Office supplies	100	100	200	330
Manufacturing supplies	855	980	1054	1890
Personnel Expenses	**7200**	**7200**	**10800**	**14400**
Salaries	7200	7200	10800	14400
External Expenses	**4706**	**3737**	**3794**	**11986**
Water	140	150	240	318
Power	1030	1040	1980	2300
Internet	324	324	324	324
Insurance	1000	1000	1200	2000
Miscellaneous	2212	1223	50	6244
Taxes	**339**	**683**	**1202**	**4194**
TOTAL	**59200**	**59200**	**70250**	**132000**

Cuadro 12: Estimated Annual Expenses for a 10 year period.

Bibliography

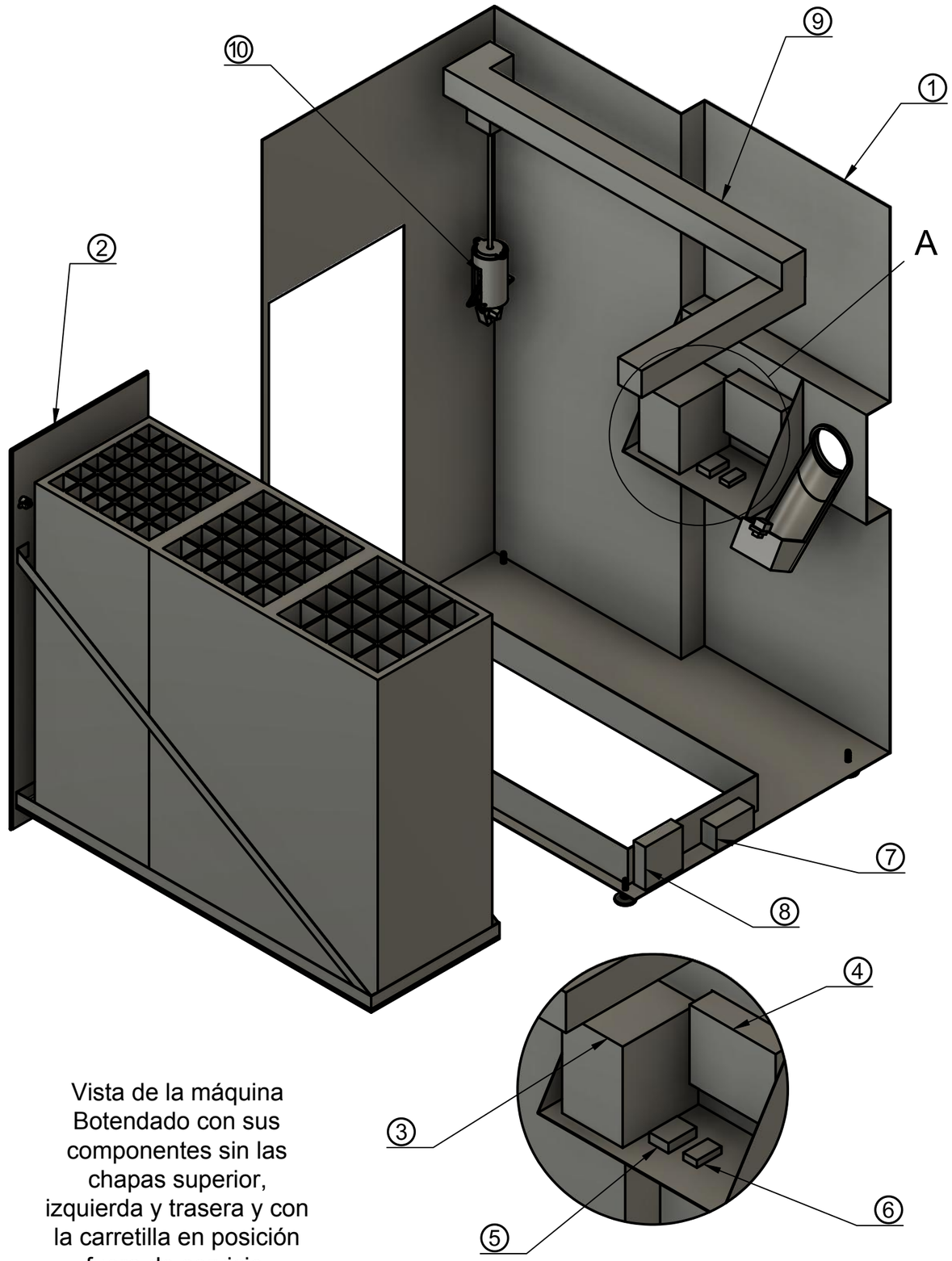
[1] M. McClure. *Everything you should know about single-use plastic*. 2021. URL: https://www.greenpeace.org/africa/en/blogs/14052/everything-you-should-know-about-single-use-plastic/#:~:text=Disposable%5C%20plastic%5C%20items%5C%20don%5C%27t,sources%5C%20and%5C%20even%5C%20our%5C%20food. [Consultado en: 10/04/2023].

[2] ONU. *Objetivos de Desarrollo Sostenible*. 2021. URL: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. [Consultado en: 10/04/2023].

[3] OECD. *Measuring distance to the SDG targets: Country profile - Spain*. URL: https://www.oecd.org/wise/measuring-distance-to-the-SDG-targets-country-profile-Spain.pdf.

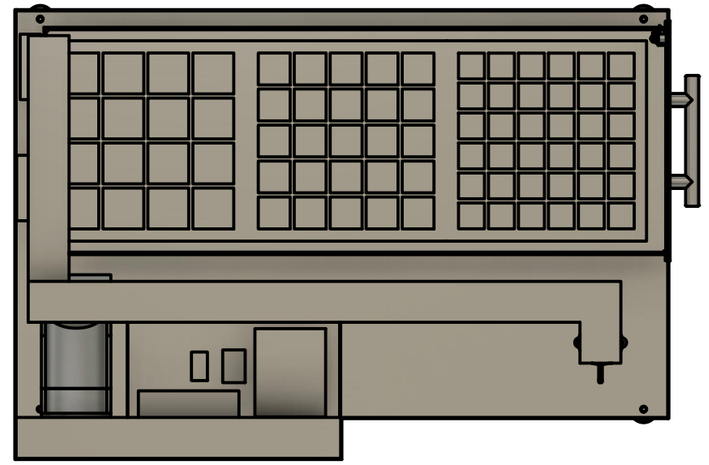
[4] Dansk Retursystem A/S. *Sustainability of the Danish Return System*. URL: https://danskretursystem.dk/en/sustainability/. [Consultado en: 12/04/2023].

[22] Spanish Law. *Real Decreto 1055/2022, de 27 de diciembre, por el que se regula la gestión de envases y residuos de envases.* [BOE núm. 313, de 29 de diciembre de 2022.]



Vista de la máquina Botendado con sus componentes sin las chapas superior, izquierda y trasera y con la carretilla en posición fuera de servicio.

A (1:10)

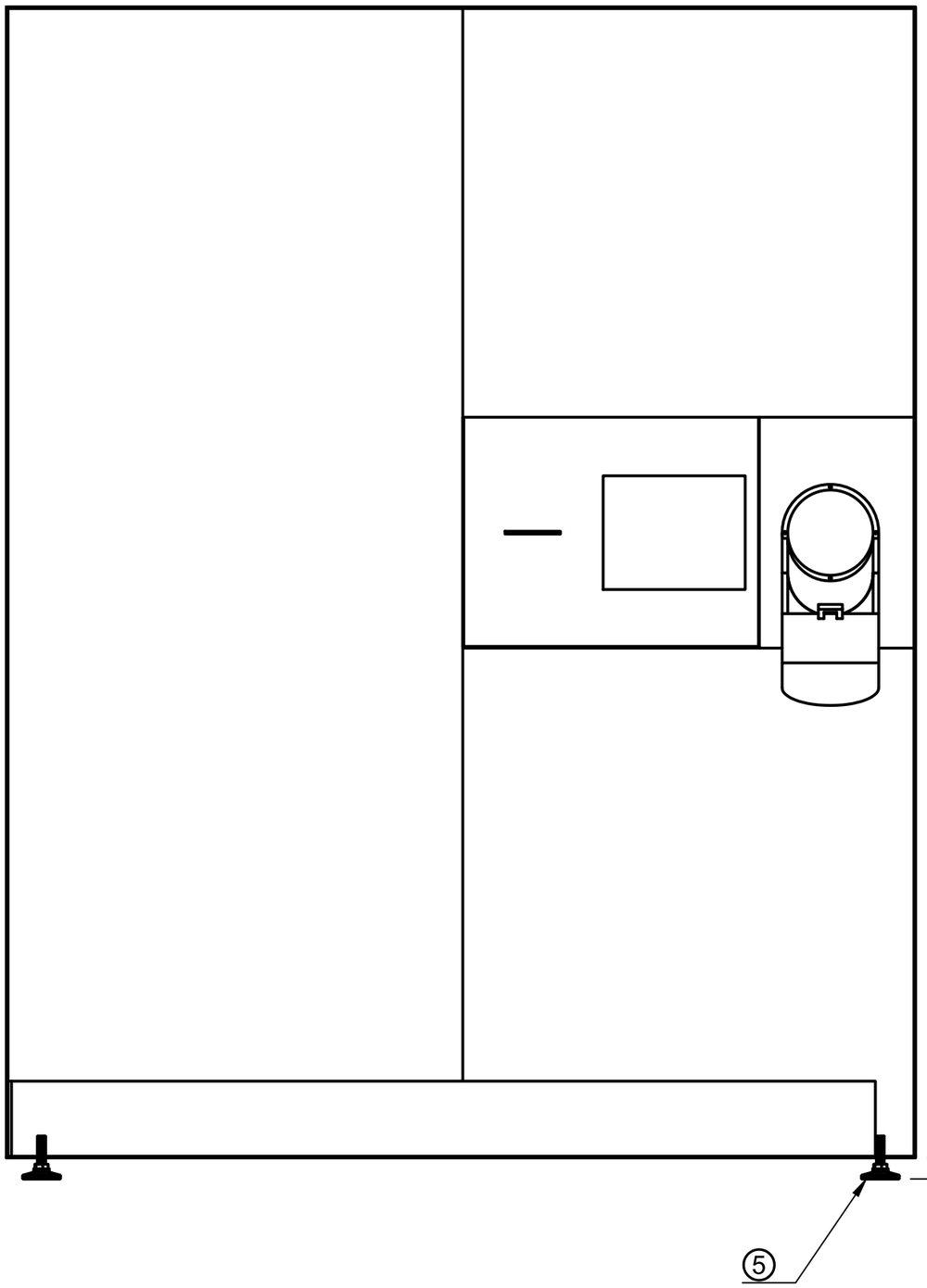


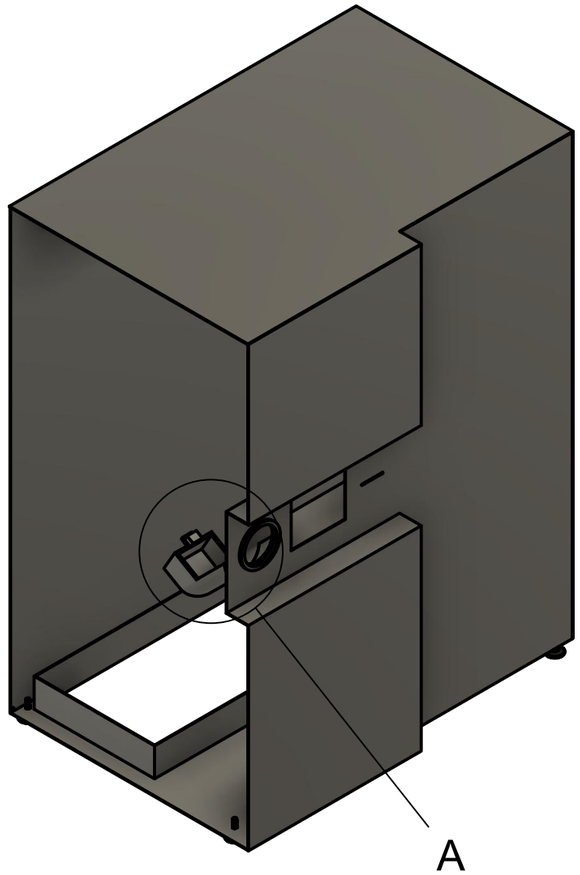
Vista de la máquina Botendado desde arriba sin la chapa superior con la carretilla en posición de servicio.

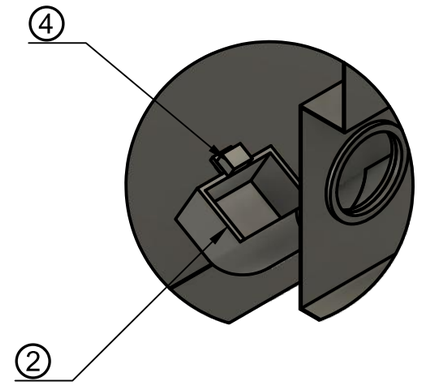
Lista de piezas

Elemento	Cantidad	Nombre	Plano correspondiente
1	1	Estructura de Botendado	11
2	1	Carretilla	12
3	1	Impresora de tíquets	Material comprado
4	1	Pantalla táctil	Material comprado
5	1	Arduino UNO	Material comprado
6	1	Convertidor	Material comprado
7	1	Convertidor	Material comprado
8	1	Fuente de alimentación	Material comprado
9	1	Robot cartesiano 3 ejes	Material comprado
10	1	Pinza robótica	Material comprado

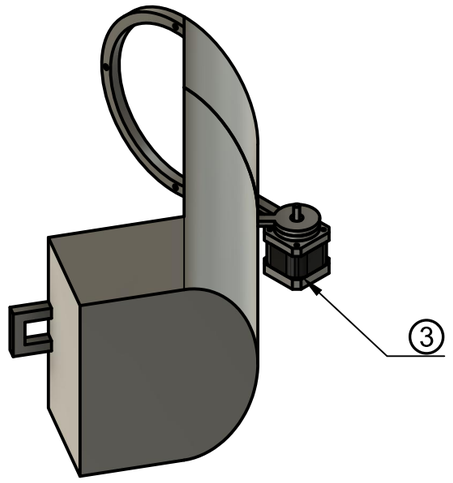
Dept.: Diseño	Technical reference: bot	Created by: Botendado	Approved by: Mercadona
		Document type: Plano de conjunto	Document status: APPROVED
1:15		Title: Botendado	DWG No.: 1
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 1/24





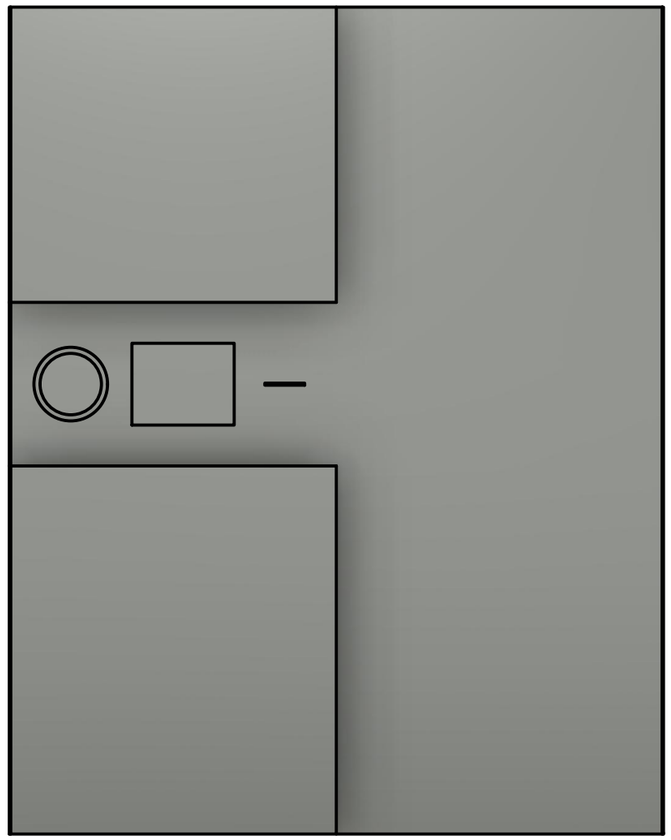


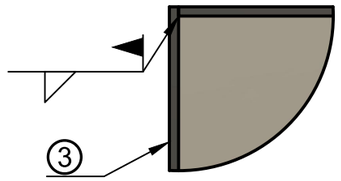
A (1:10)



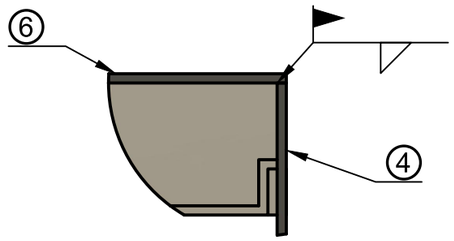
Lista de piezas			
Elemento	**Cantidad**	**Nombre**	**Plano correspondiente**
1	1	Estructura vacía	11.1
2	1	Mecanismo de aceptación	11.2
3	1	Motor paso a paso	Material comprado
4	1	Cámara de visión artifcial	Material comprado
5	4	Patas de altura regulable	Material comprado

Dept. Diseño	Technical reference est_bot	Created by Botendado	Approved by Mercadona
		Document type Plano de conjunto	Document status APPROVED
1:10		Title Estructura de Botendado	DWG No. 11
			Rev. - / Date of issue 20/03/2035 / Sheet 2/24

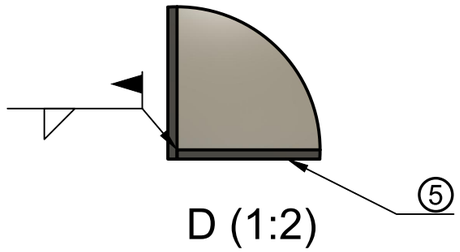




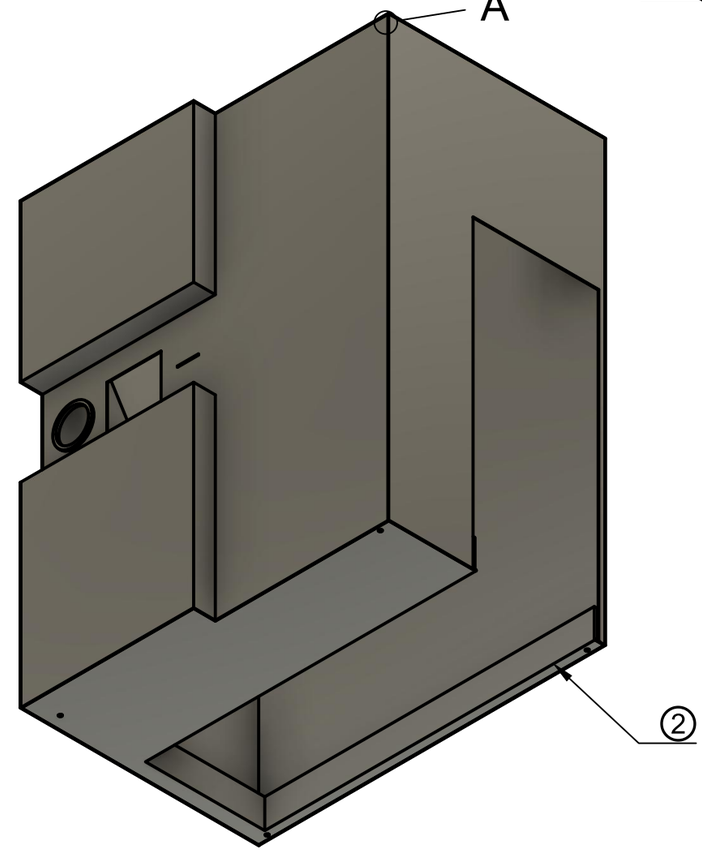
B (1:2)

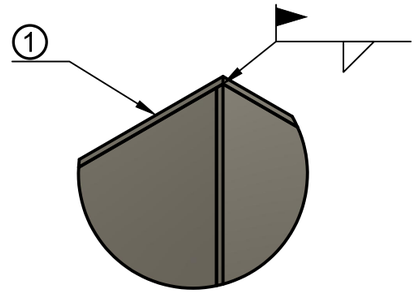


C (1:2)

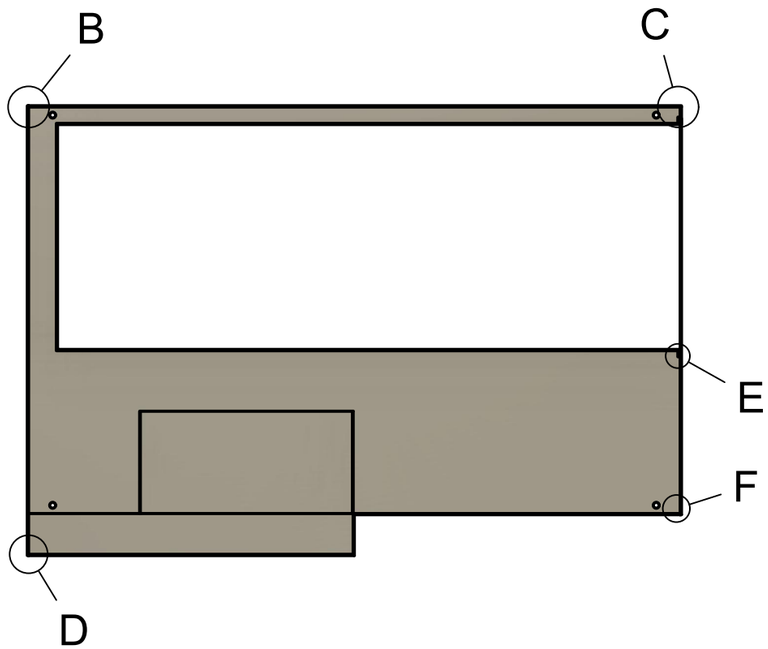


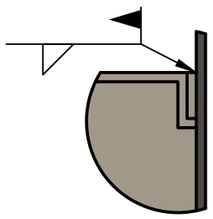
D (1:2)



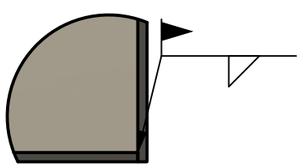


A (1:2)





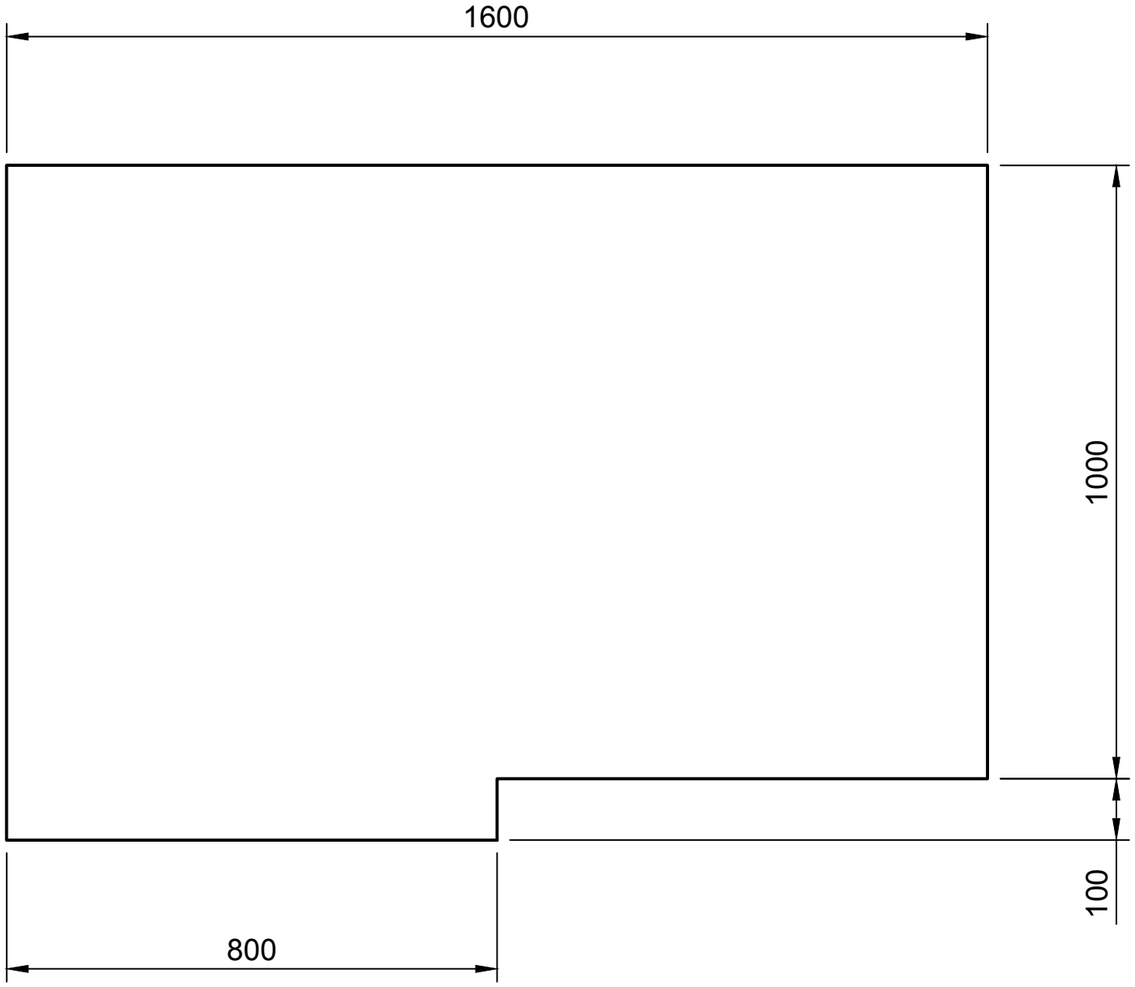
E (1:2)

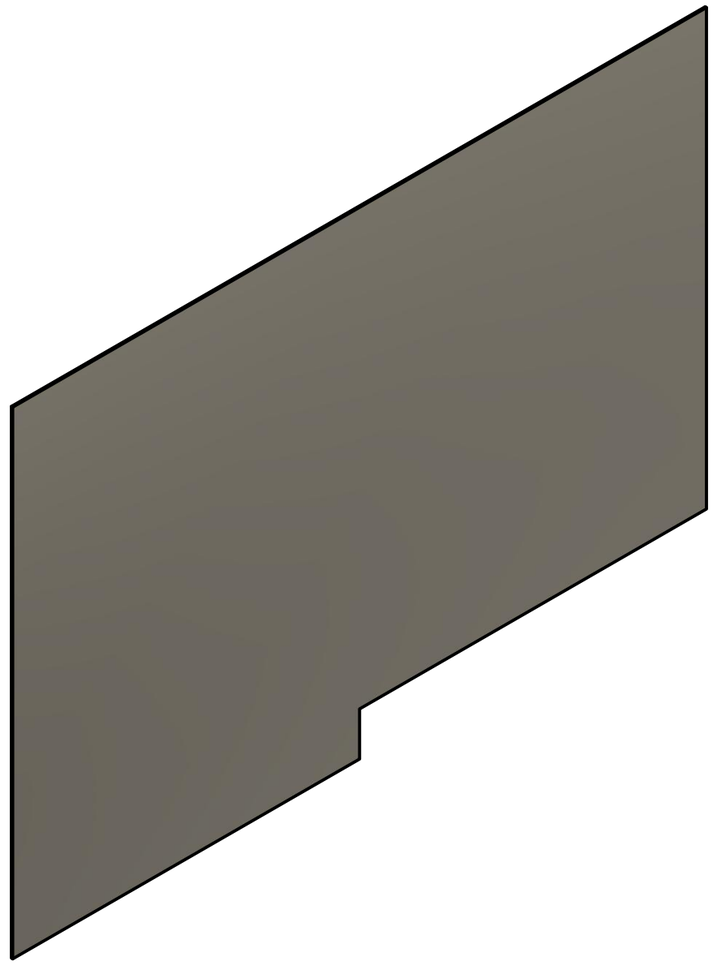


F (1:2)

Lista de piezas			
Elemento	**Cantidad**	**Nombre**	**Plano correspondiente**
1	1	Chapa superior	11.1.1
2	1	Chapa inferior	11.1.2
3	1	Chapa izquierda	11.1.3
4	1	Chapa derecha	11.1.4
5	1	Chapa frontal	11.1.5
6	1	Chapa trasera	11.1.6

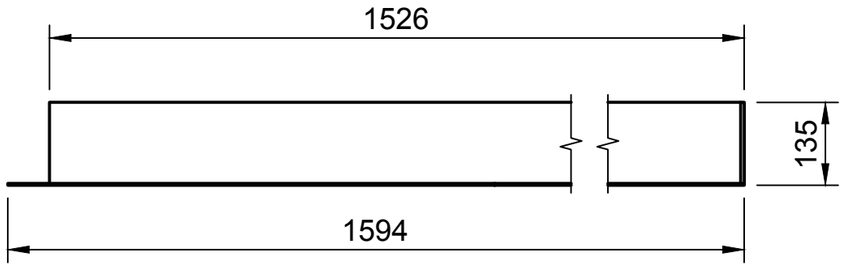
Dept. Diseño	Technical reference est_vac	Created by Botendado	Approved by Mercadona
		Document type Plano de conjunto	Document status APPROVED
1:15		Title Estructura vacía	DWG No. 11.1
			Rev. - / Date of issue 20/03/2035 / Sheet 3/24

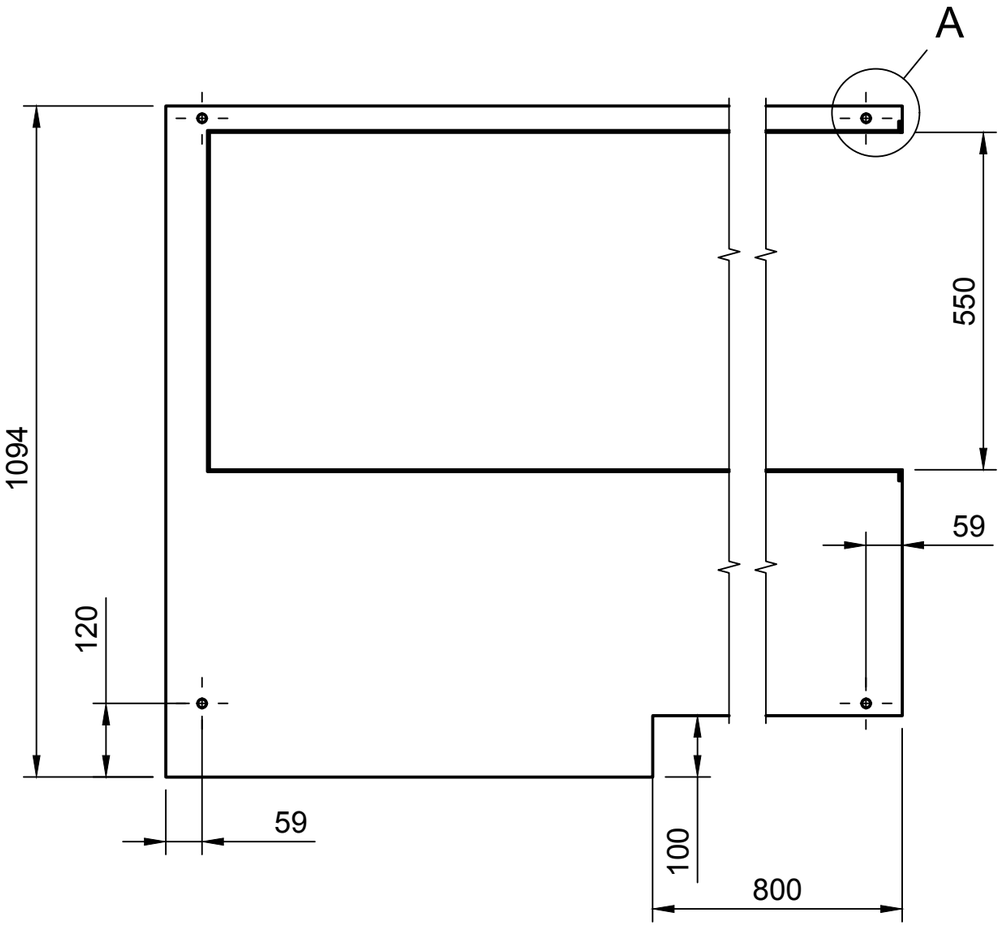


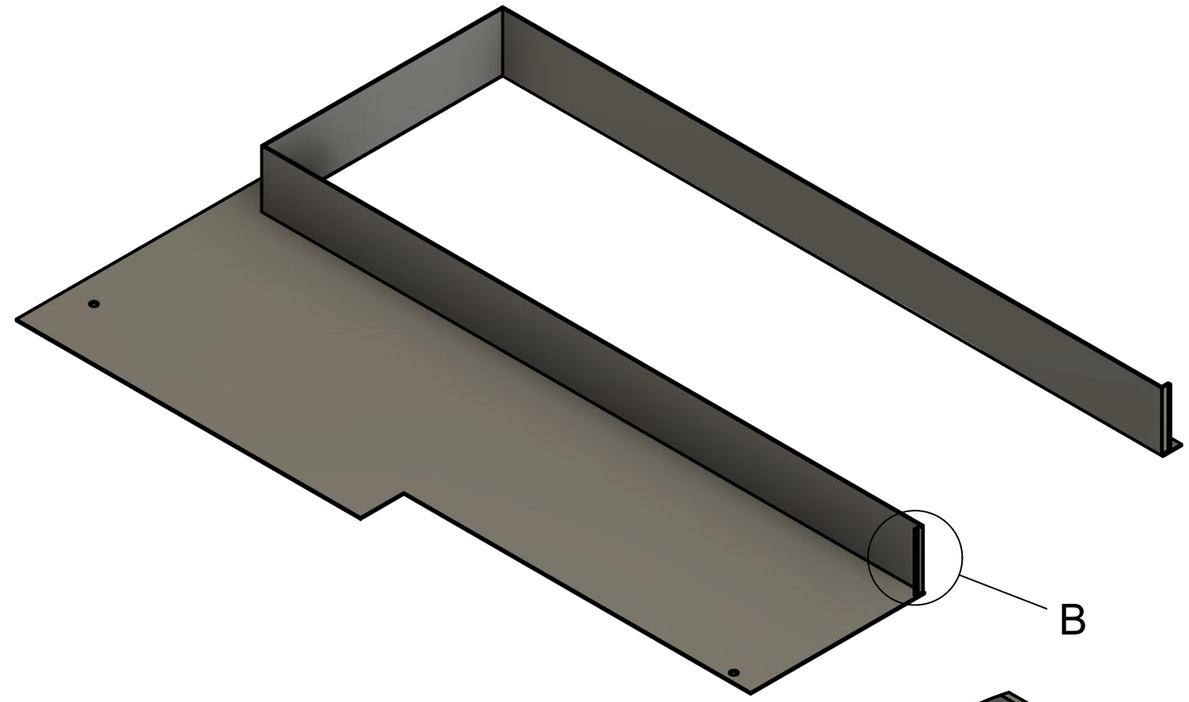


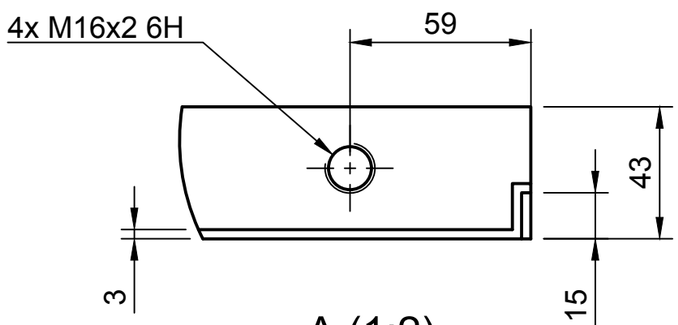
El grosor de la plancha de acero es igual a 3mm.

Dept.	Technical reference	Created by	Approved by
Diseño	cha_ac_sup	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:10		Title: Chapa Superior	DWG No.: 11.1.1
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 4/24

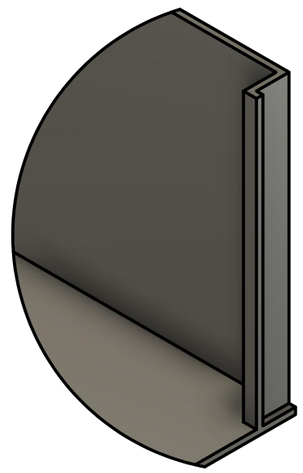








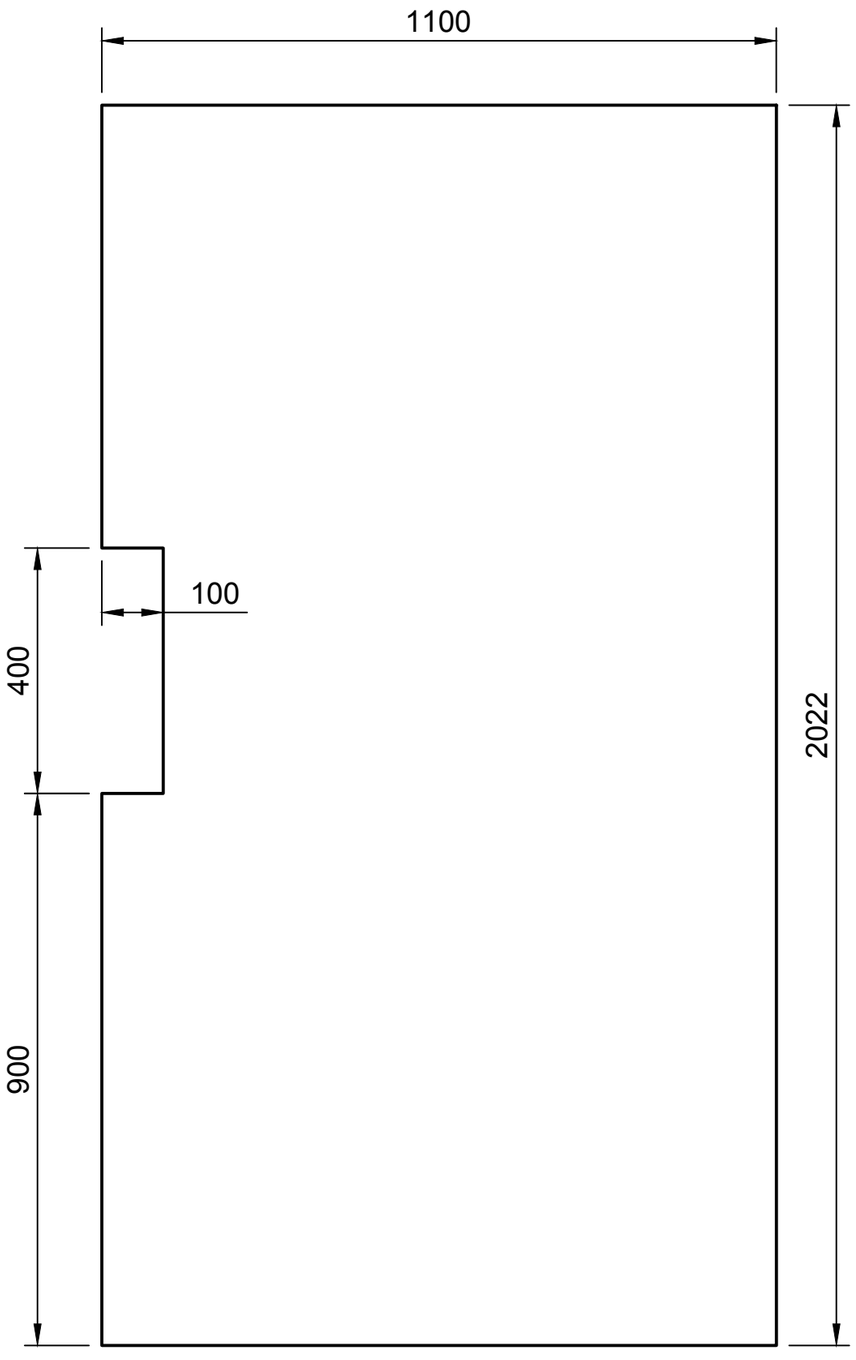
A (1:2)

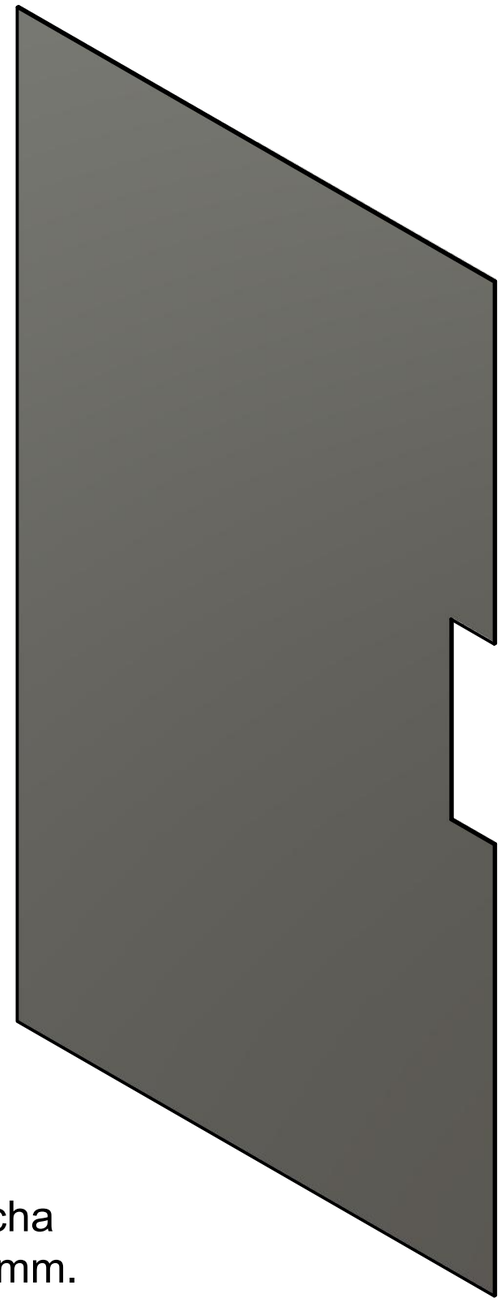


B (1:2)

El grosor de la plancha de acero es igual a 3mm

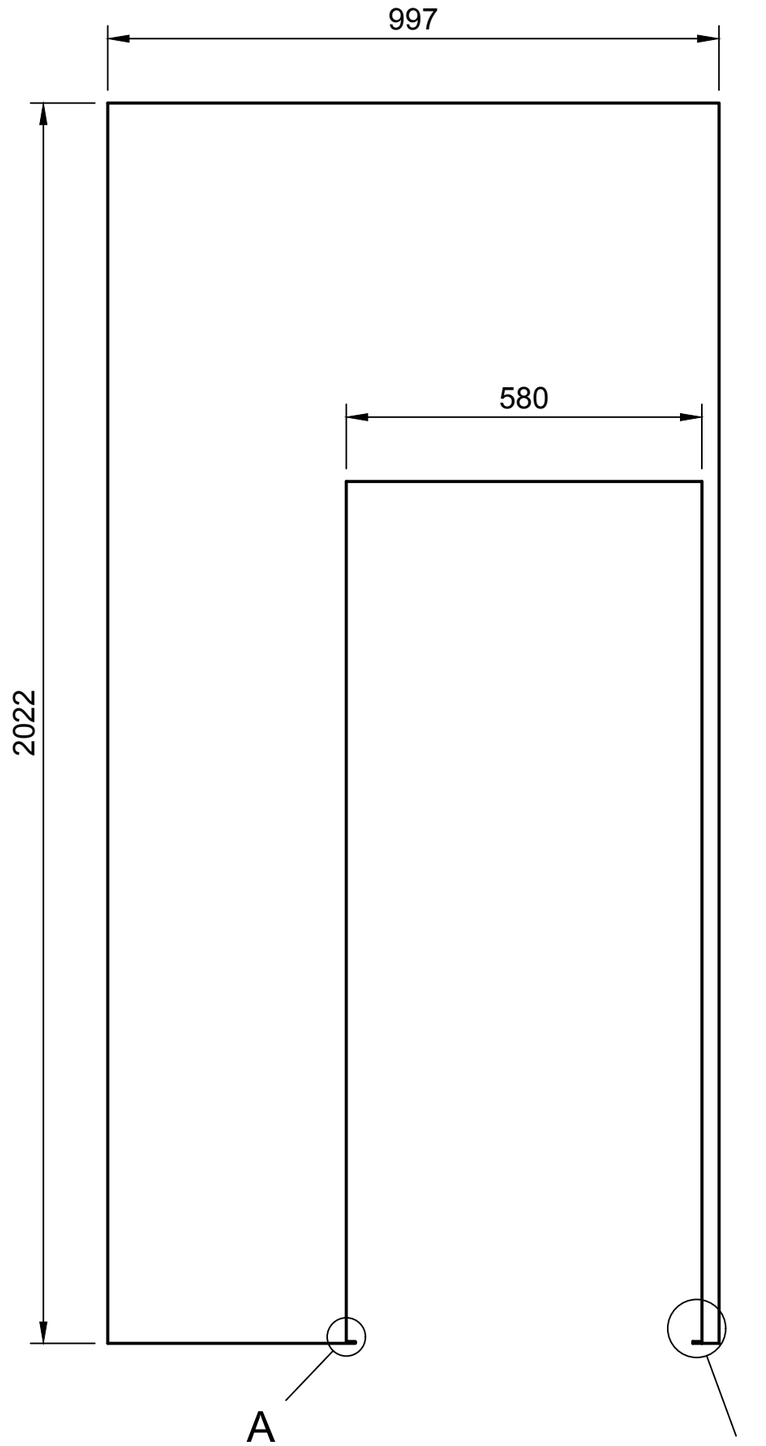
Dept.	Technical reference	Created by	Approved by
Diseño	cha_ac_inf	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:10		Title: Chapa Inferior	DWG No.: 11.1.2
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 5/24

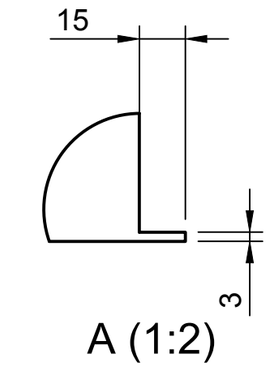




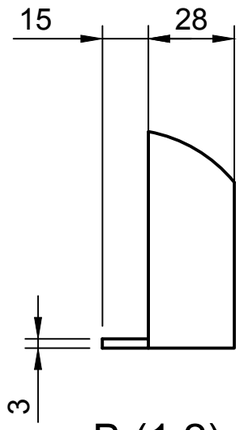
El grosor de la plancha
de acero es igual a 3mm.

Dept.	Technical reference	Created by	Approved by
Diseño	cha_ac_izq	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:10		Title: Chapa izquierda	DWG No.: 11.1.3
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 6/24

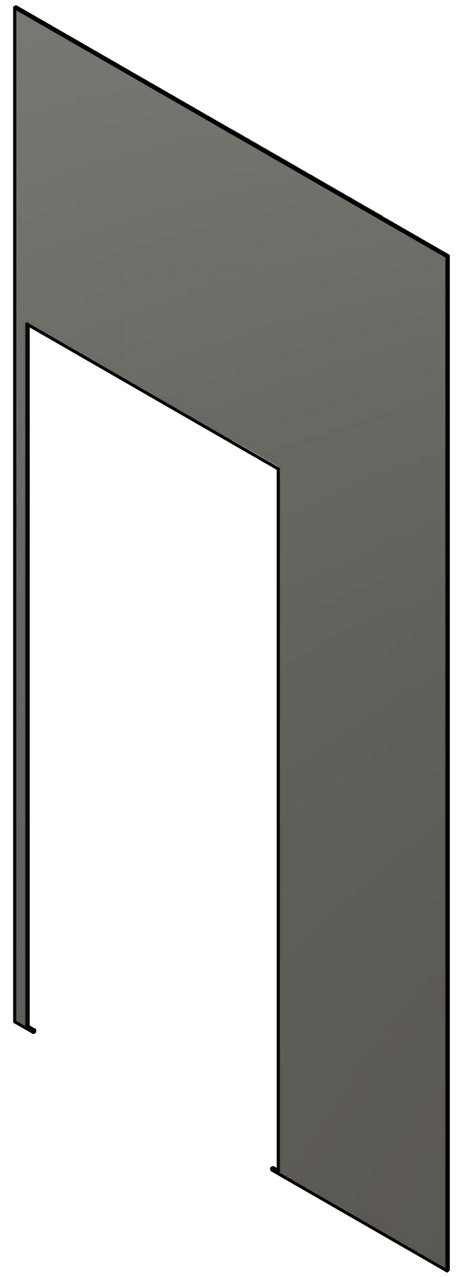




A (1:2)



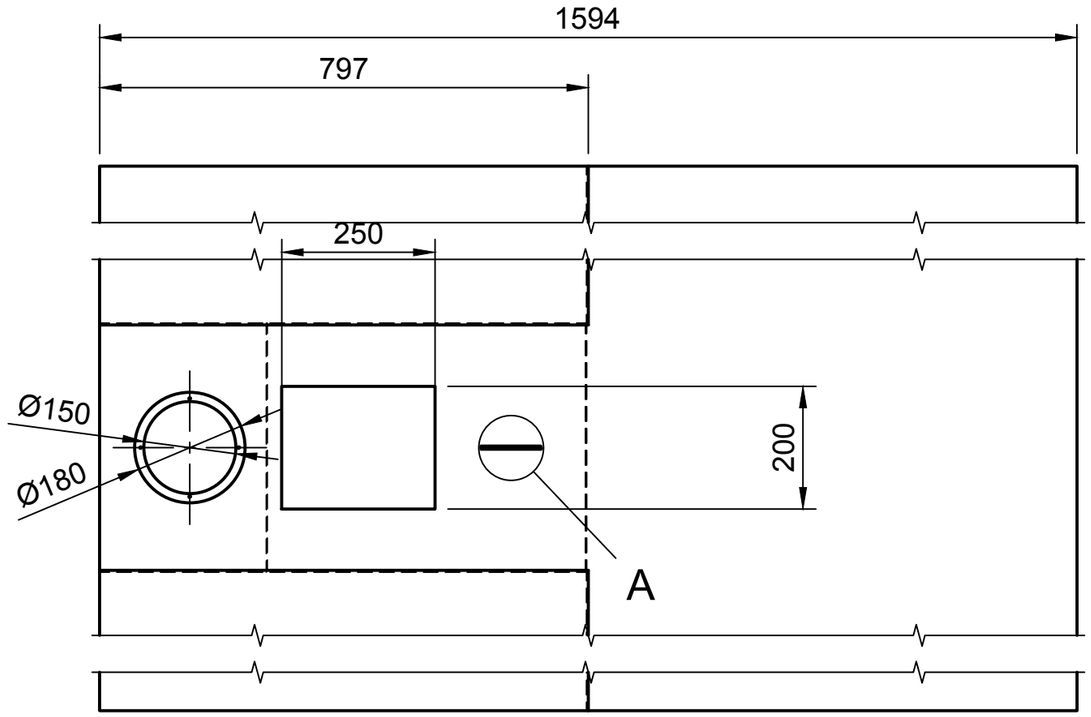
B (1:2)

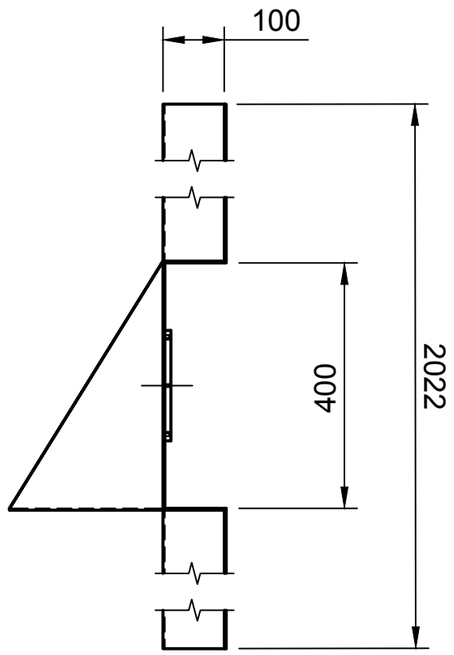


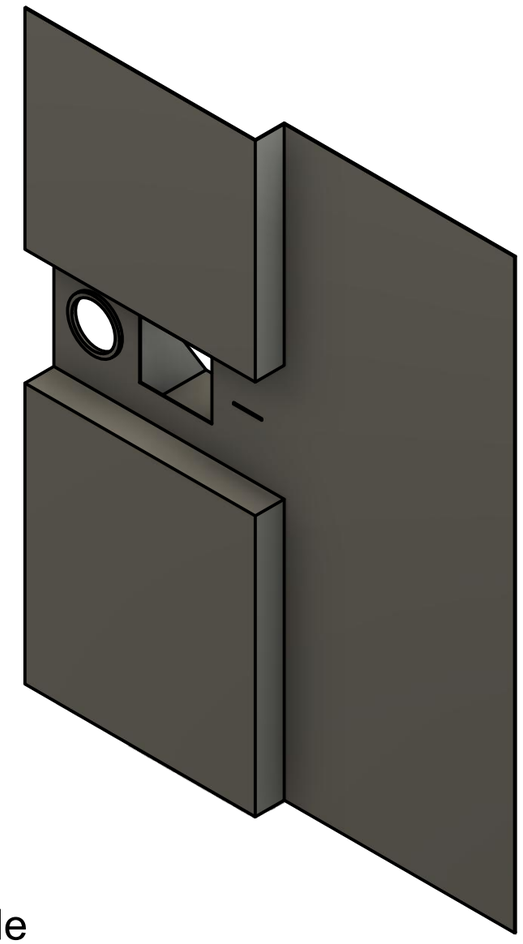
El grosor de la plancha
de acero es igual a 3mm

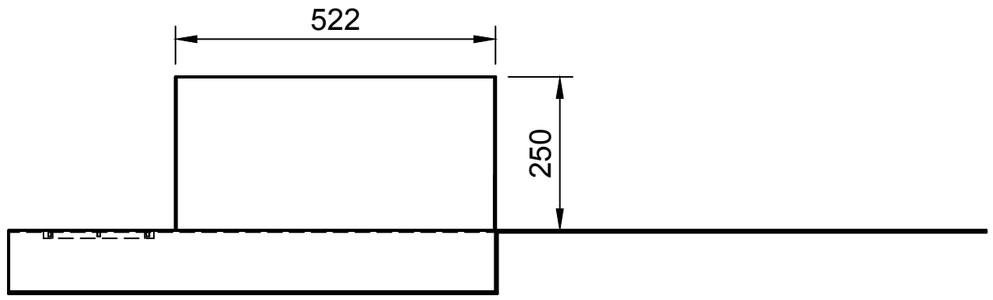
Dept.	Technical reference	Created by	Approved by
Diseño	cha_ac_der	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:10		Title: Chapa Derecha	DWG No.: 11.1.4

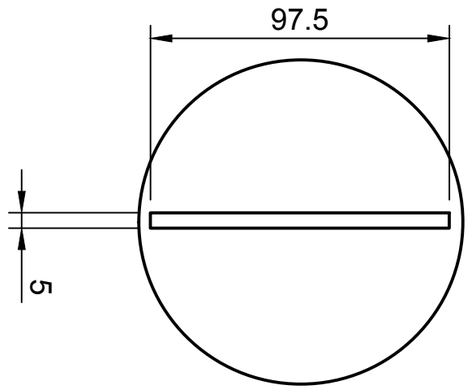
Rev.	Date of issue	Sheet
-	20/03/2035	7/24







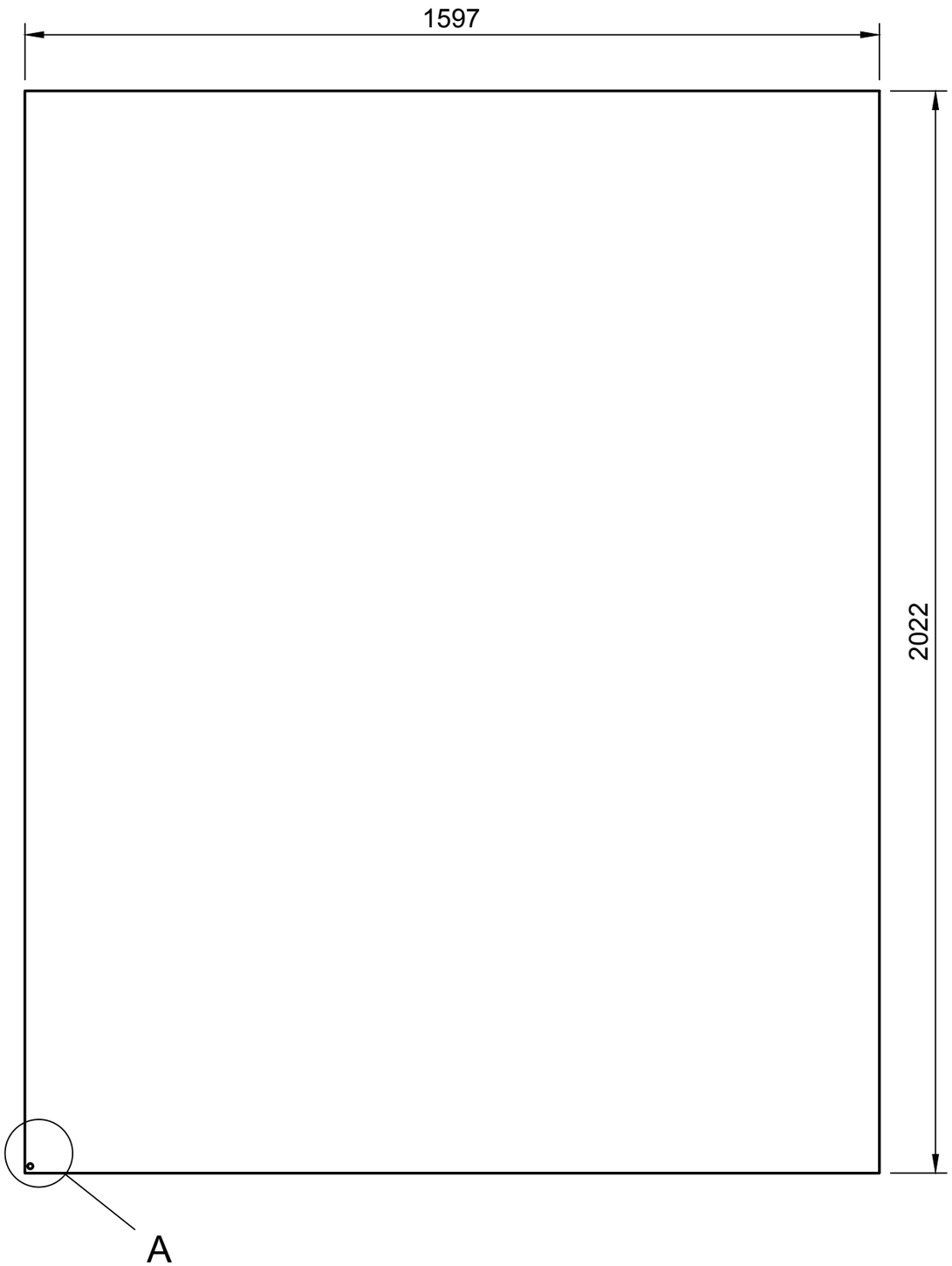


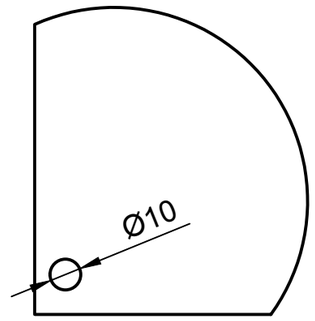


A (1:2)

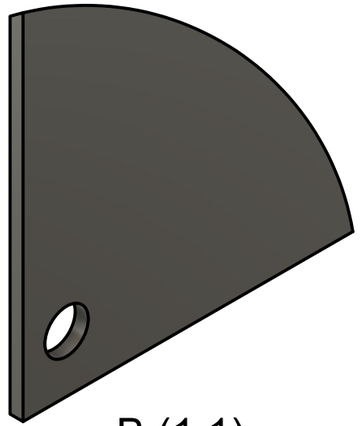
El grosor de la plancha de acero es igual a 3mm

Dept.	Technical reference	Created by	Approved by
Diseño	cha_ac_fro	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:10		Title: Chapa Frontal	DWG No.: 11.1.5
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 8/24

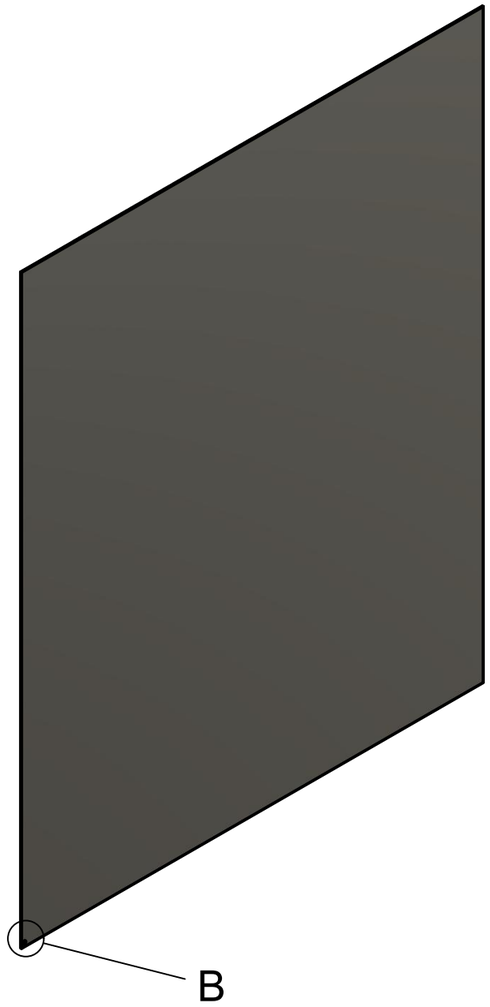




A (1:2)

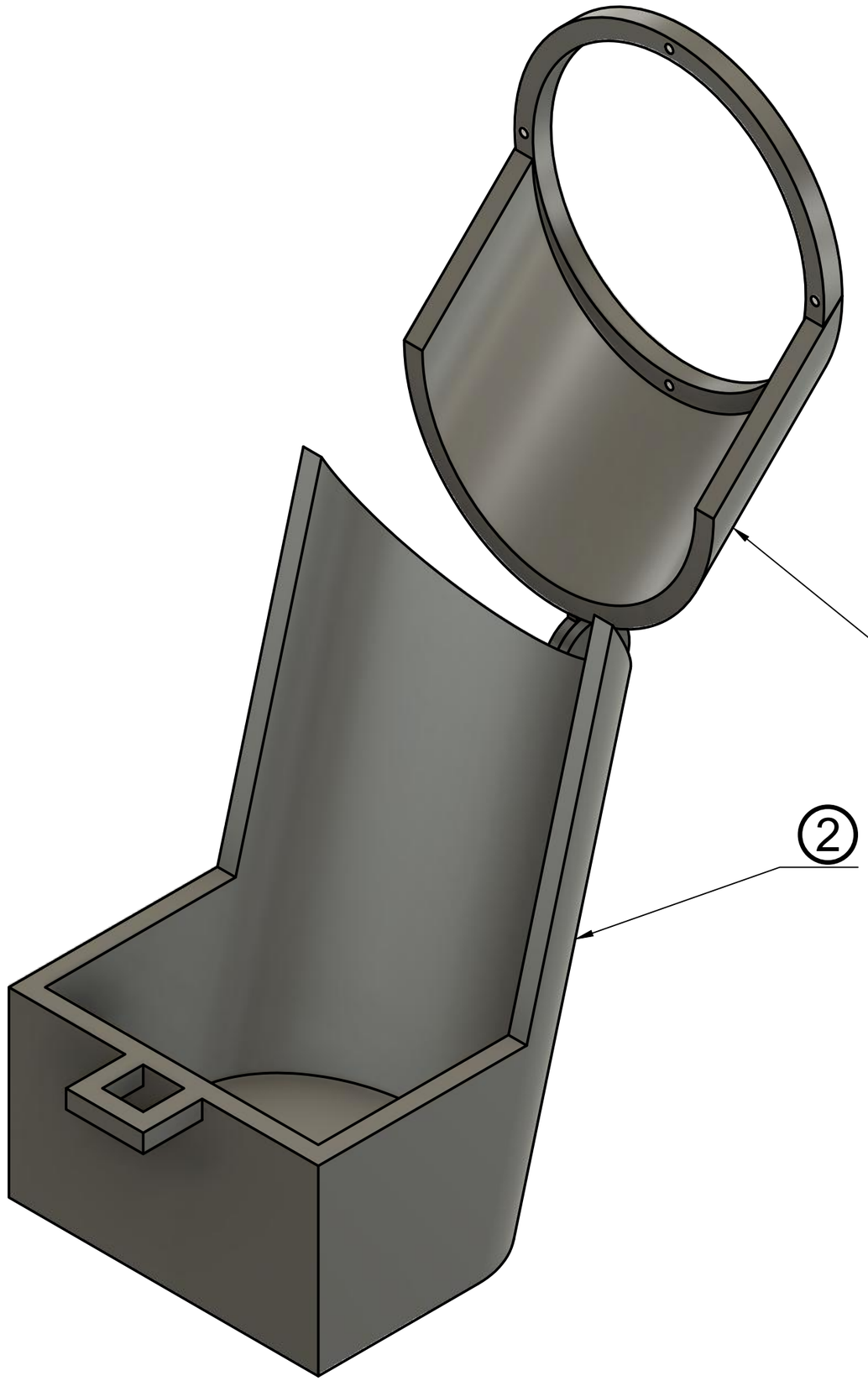


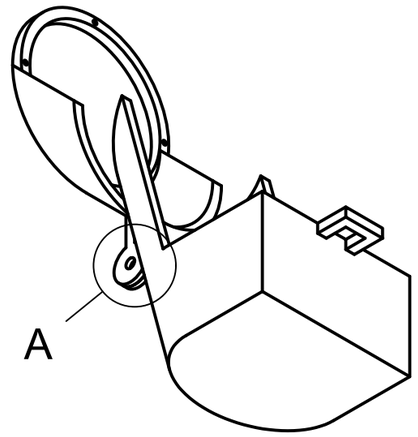
B (1:1)



El grosor de la plancha de acero es igual a 3mm

Dept.	Technical reference	Created by	Approved by
Diseño	cha_ac_tra	Botendado	Mercadona
1:10		Document type: Plano de pieza	Document status: APPROVED
		Title: Chapa Trasera	DWG No.: 11.1.6
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 9/24





1:5

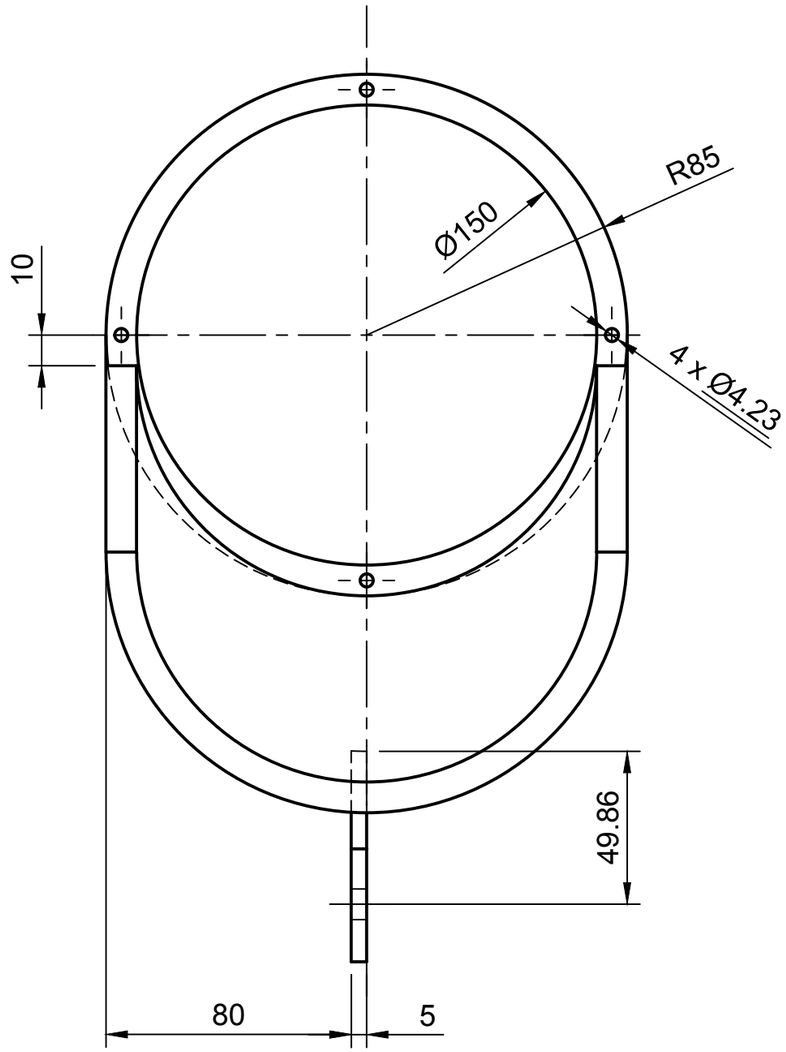
A (1:2)

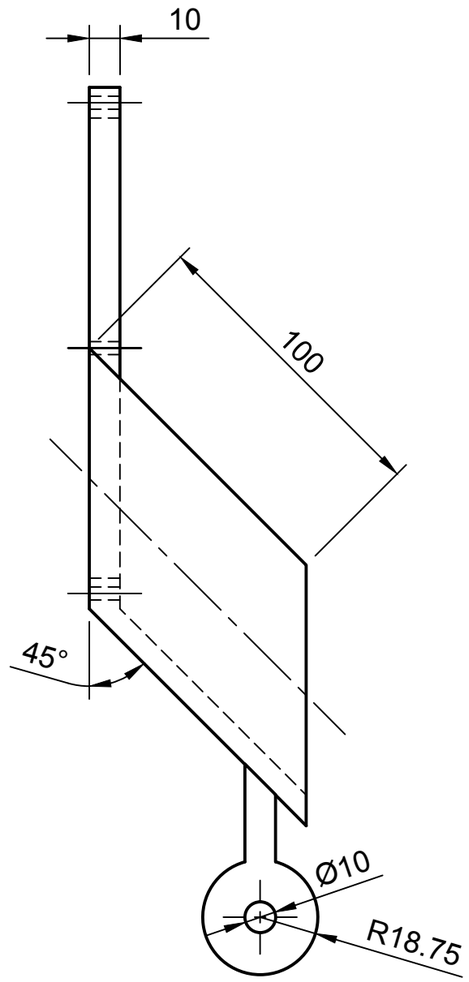
Los mecanismos inferior y superior se conectan uniendo sus secciones circulares de acuerdo la posición su posición relativa en el corte A.

Lista de piezas

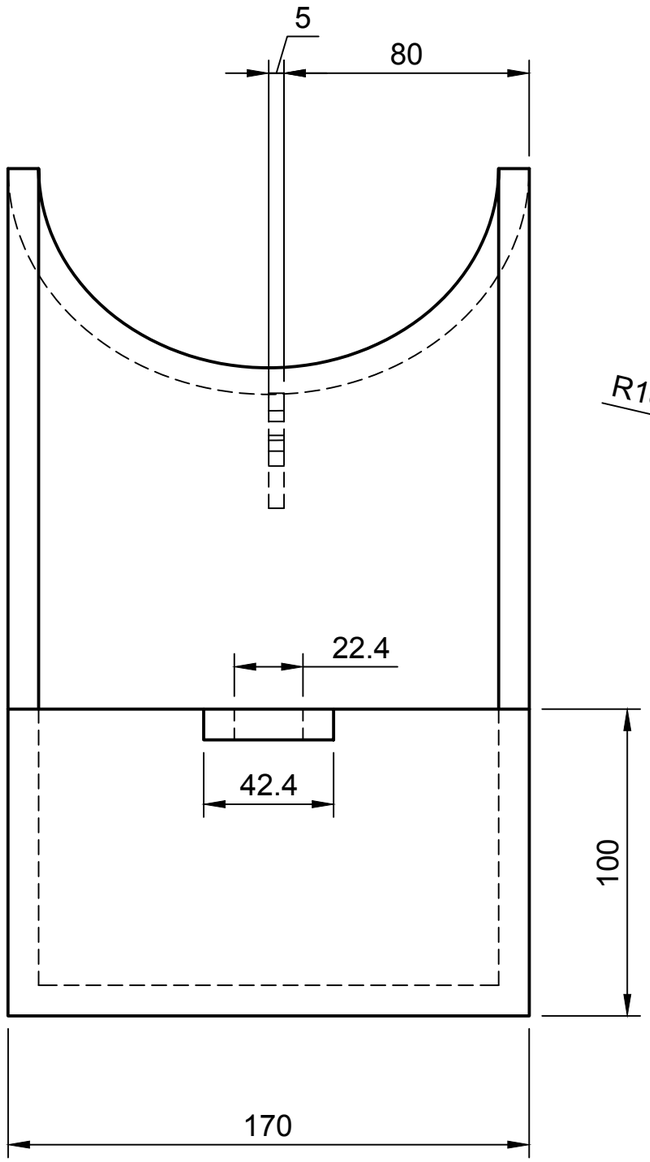
Elemento	Cantidad	Nombre	Plano correspondiente
1	1	Acople fijo superior	11.2.1
2	1	Sección giratoria inferior	11.2.2

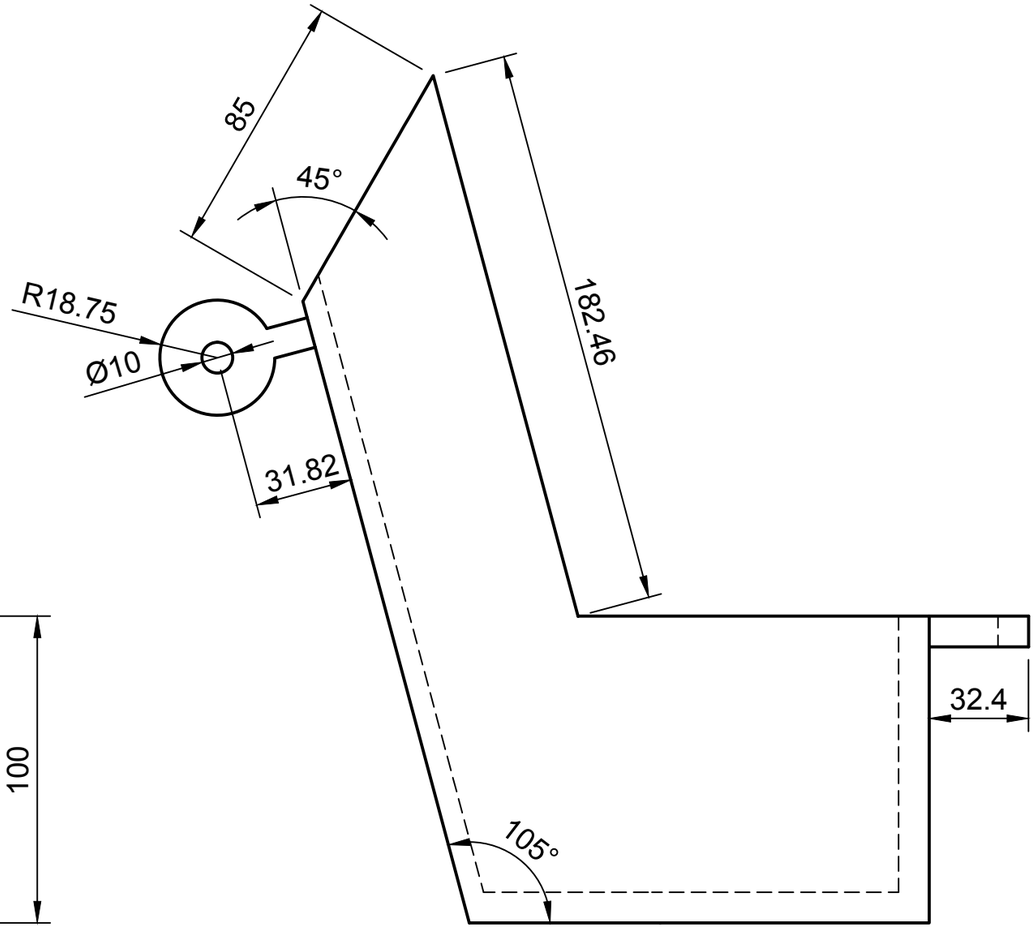
Dept. Diseño	Technical reference mec_ace	Created by Botendado	Approved by Mercadona
1:2		Document type Plano de conjunto	Document status APPROVED
		Title Mecanismo de aceptación	DWG No. 11.2
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 10/24

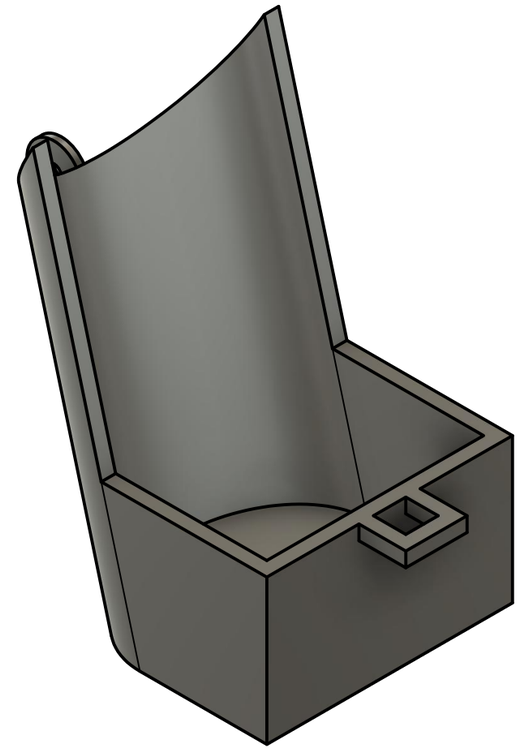


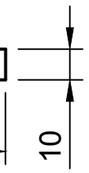


Dept.	Technical reference	Created by	Approved by
Diseño	aco_fij	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:2		Title: Acople fijo superior	DWG No.: 11.2.1
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 11/24

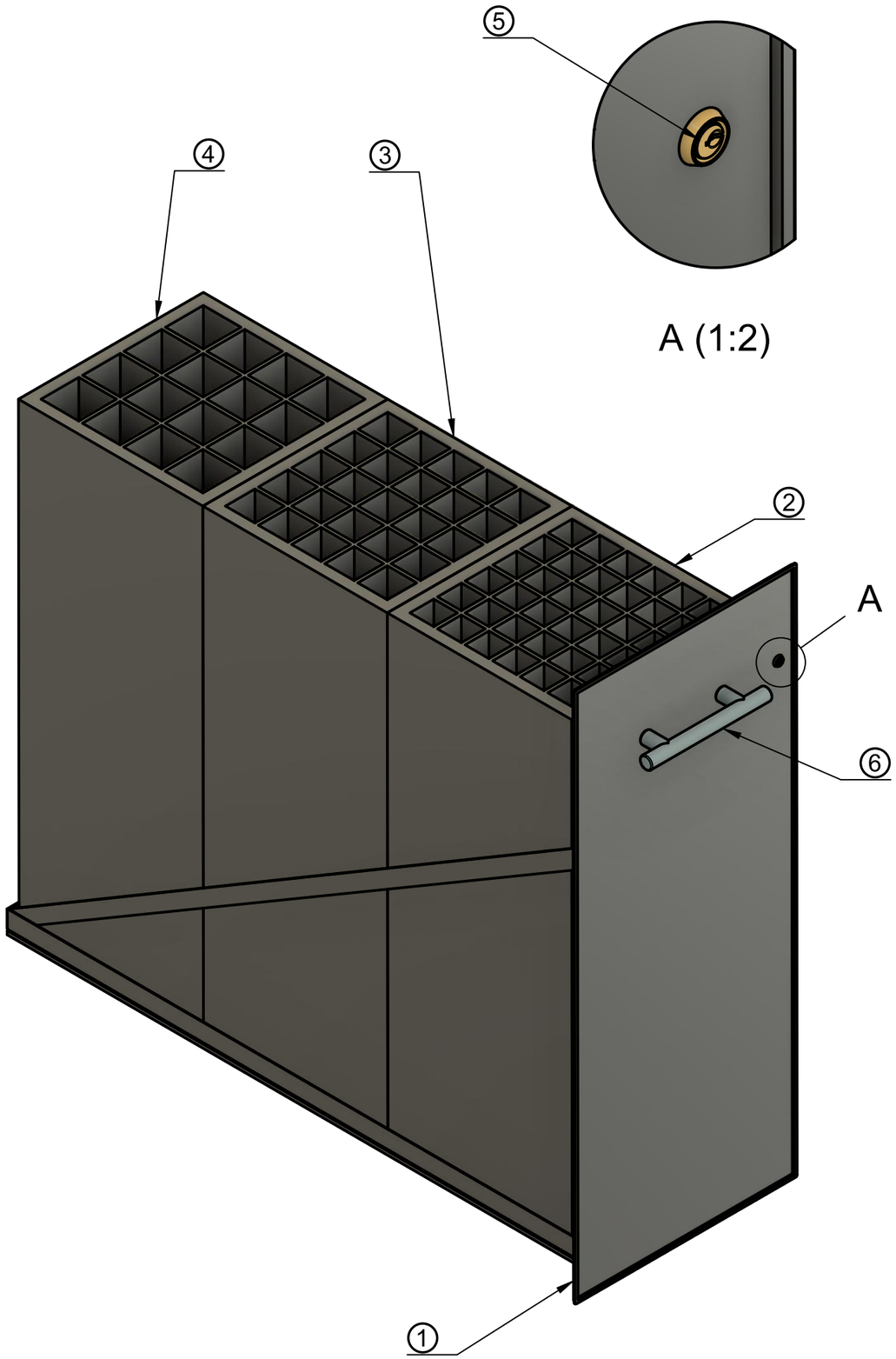




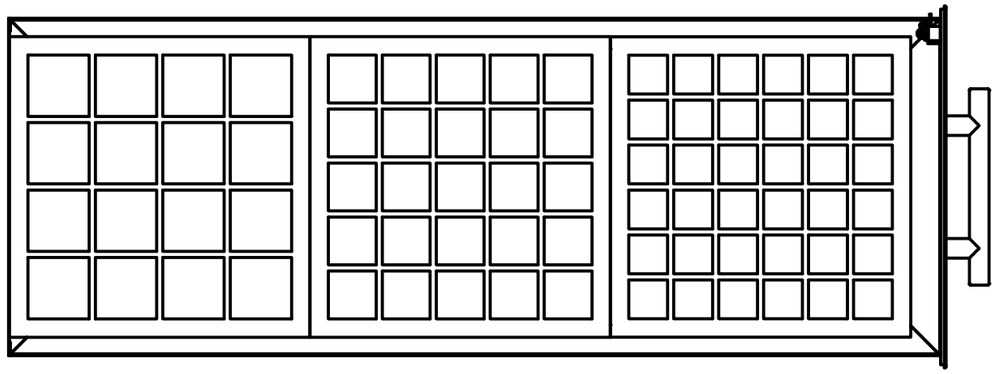




Dept.	Technical reference	Created by	Approved by
Diseño	sec_gir	Botendado	Mercadona
1:2		Document type: Plano de pieza	Document status: APPROVED
		Title: Sección giratoria inferior	DWG No.: 11.2.2
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 12/24



A (1:2)

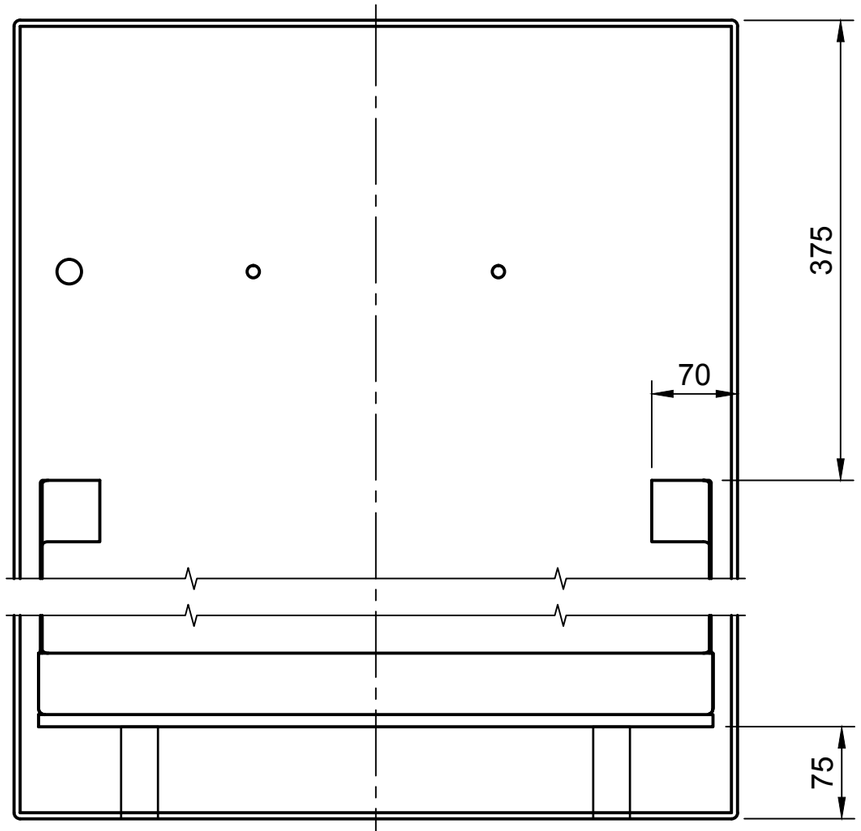


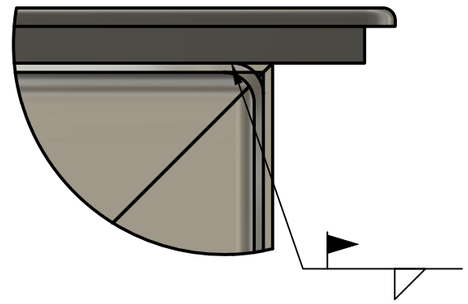
Posición y orden de los contenedores en la carretilla.

Lista de piezas

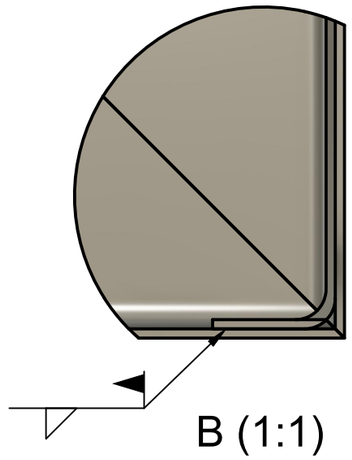
Elemento	Cantidad	Nombre	Plano correspondiente
1	1	Estructura de la carretilla	12.1
2	1	Contenedor botellas peq.	12.2
3	1	Contenedor botellas med.	12.3
4	1	Contenedor botellas grande	12.4
5	1	Cerradura autoenrroscable	Material comprado
6	1	Tirador de acero	Material comprado

Dept.: Diseño	Technical reference: car	Created by: Botendado	Approved by: Mercadona
1:10		Document type: Plano de conjunto	Document status: APPROVED
		Title: Carretilla	DWG No.: 12
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 13/24

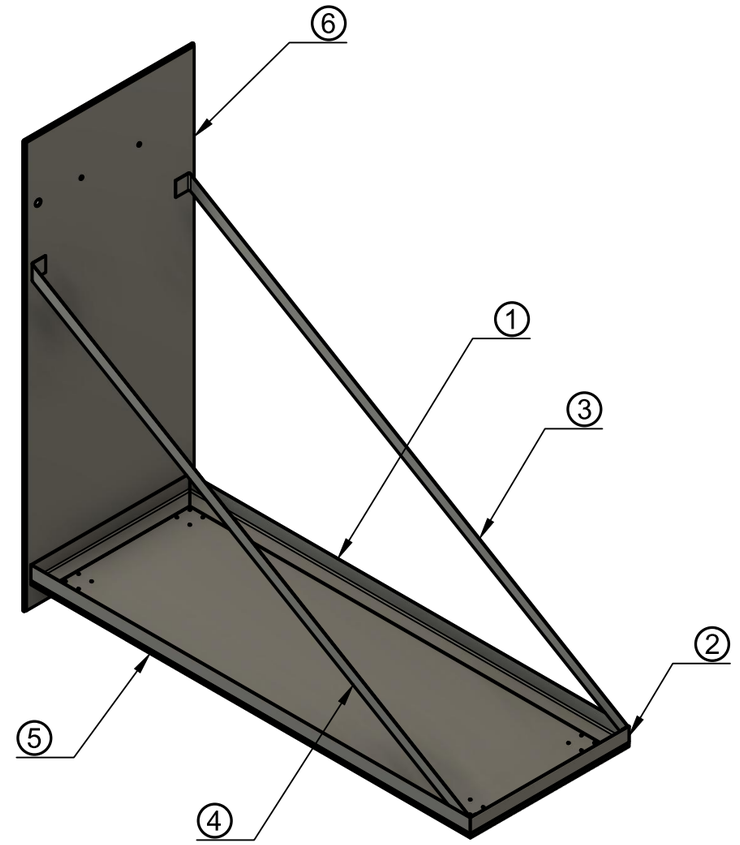


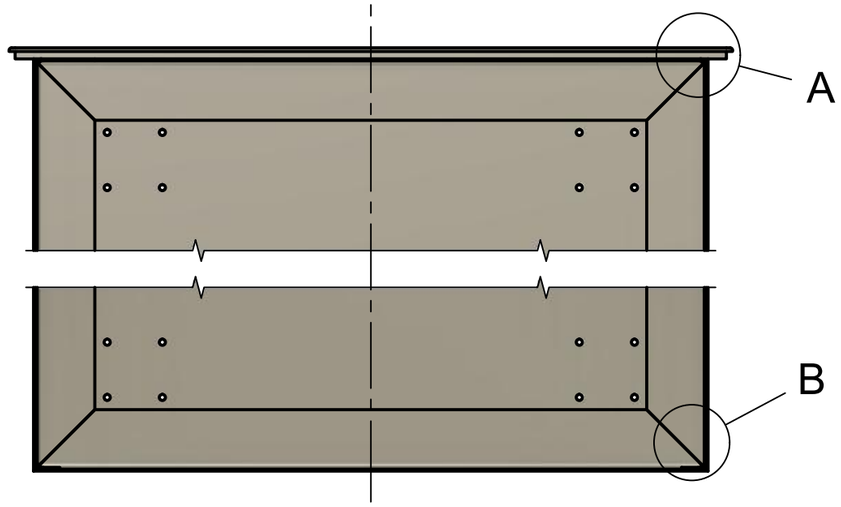


A (1:1)



B (1:1)

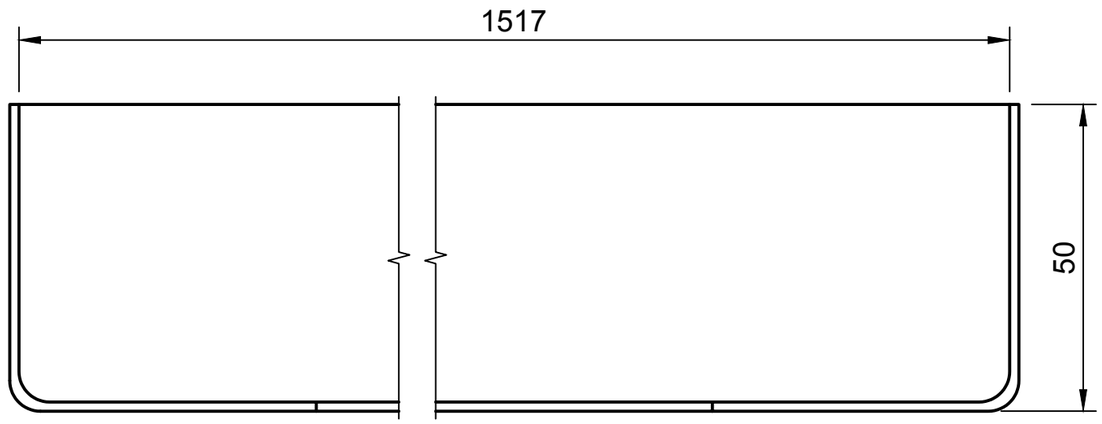


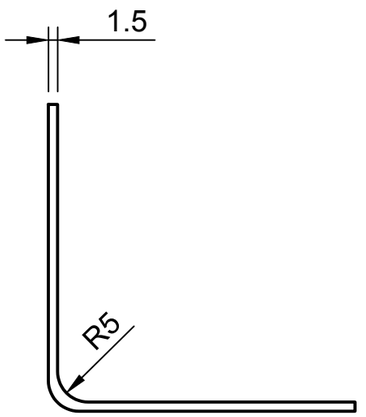


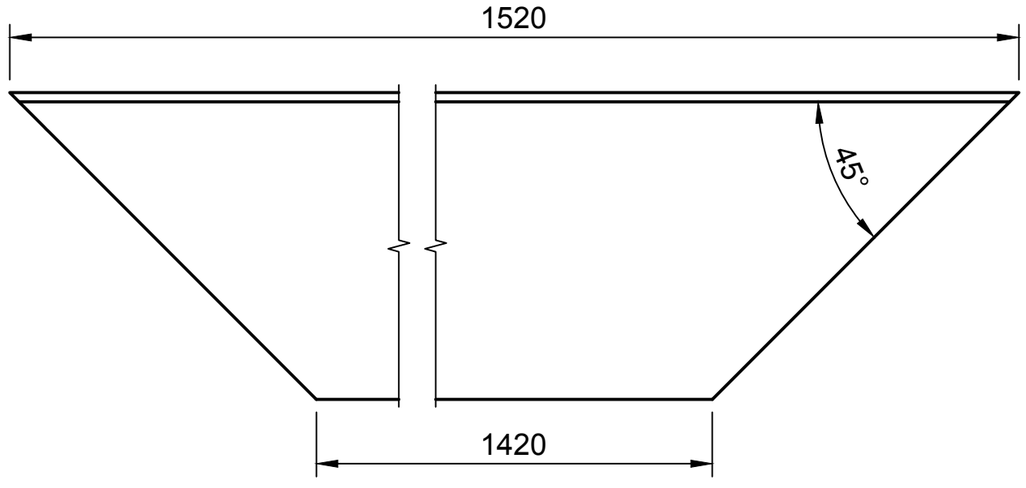
Lista de piezas

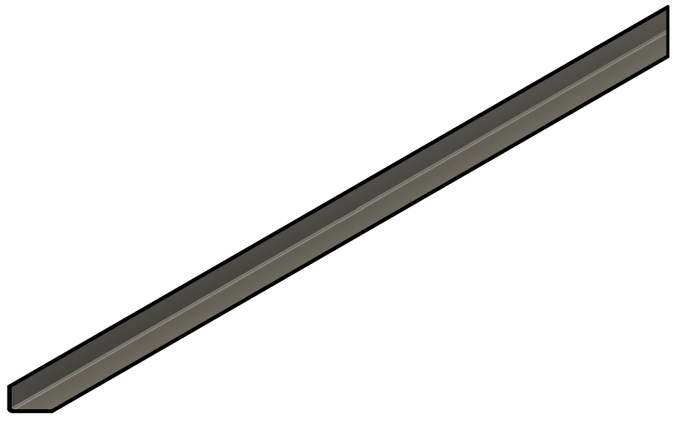
Elemento	Cantidad	Nombre	Plano correspondiente
1	2	Perfil lateral largo	12.1.1
2	2	Perfil lateral corto	12.1.2
3	1	Perfil diagonal izquierdo	12.1.3
4	1	Perfil diagonal derecho	12.1.4
5	1	Base de la carretilla	12.1.5
6	1	Puerta de la carretilla	12.1.6

Dept. Diseño	Technical reference est_car	Created by Botendado	Approved by Mercadona
		Document type Plano de conjunto	Document status APPROVED
1:5		Title Estructura de la carretilla	DWG No. 12.1
			Rev. - / Date of issue 20/03/2035 / Sheet 14/24

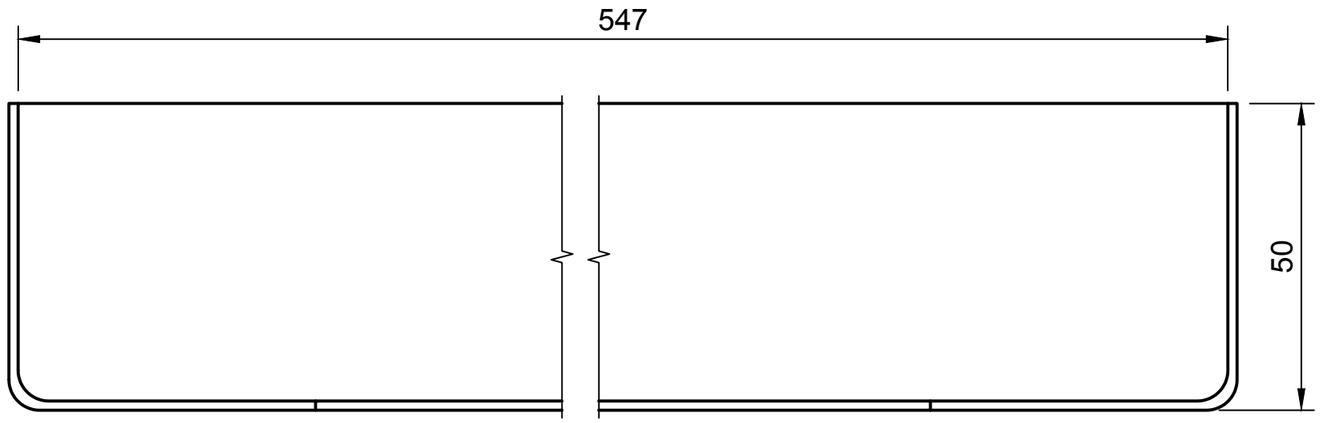


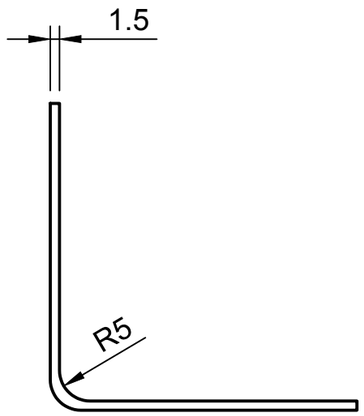


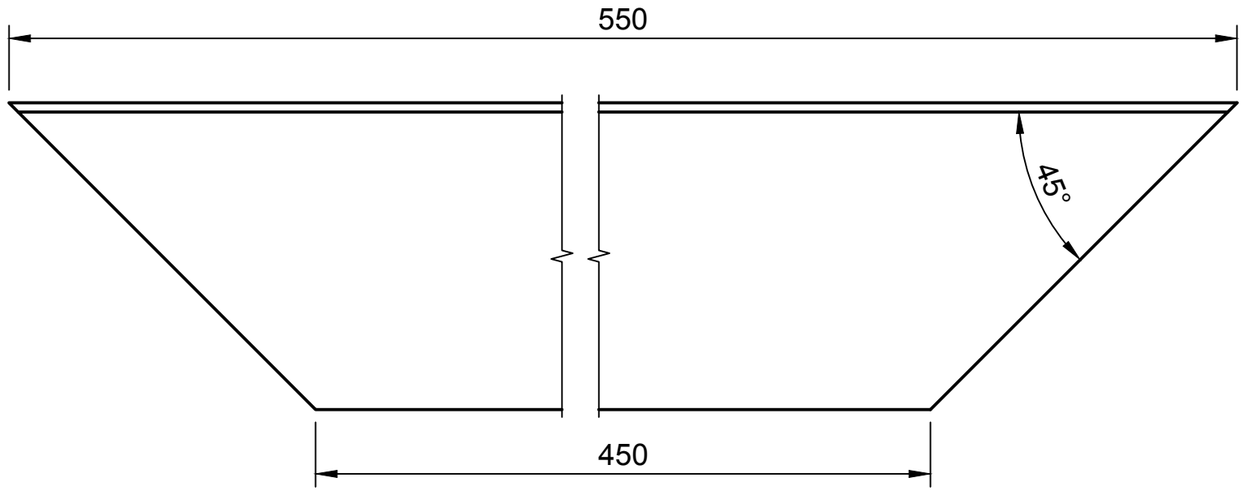


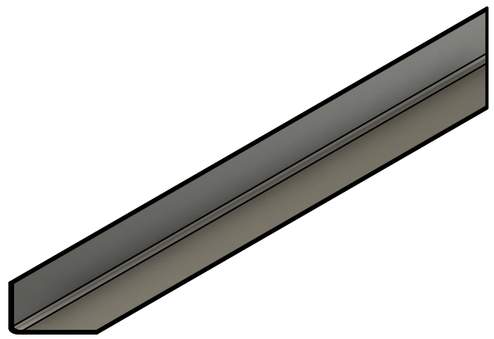


Dept.	Technical reference	Created by	Approved by
Diseño	per_lat_lar	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:1		Title: Perfil lateral largo	DWG No.: 12.1.1
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 15/24

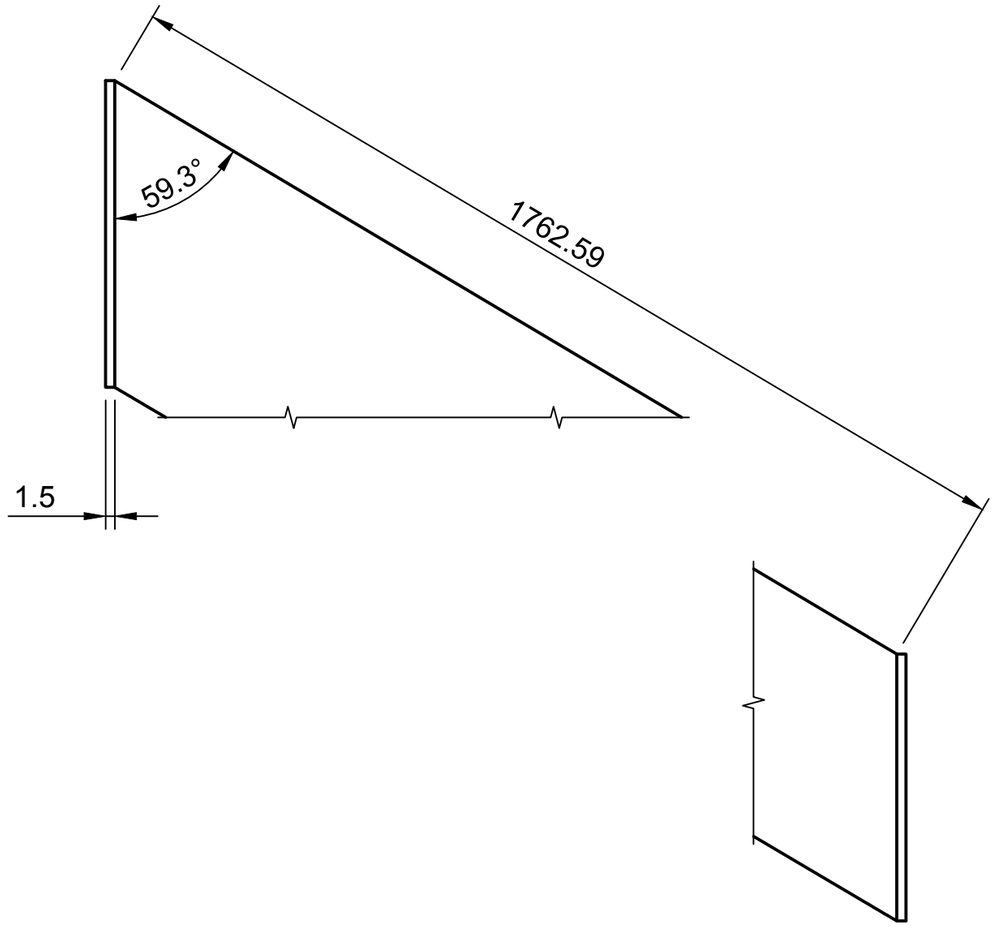


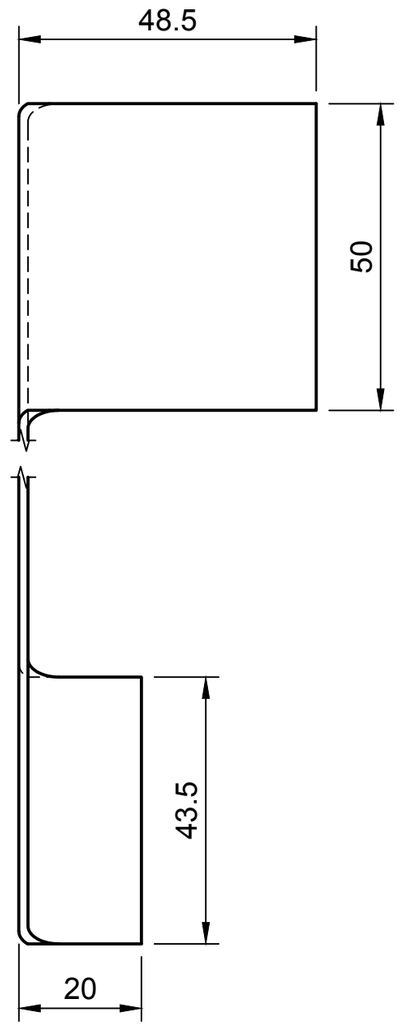


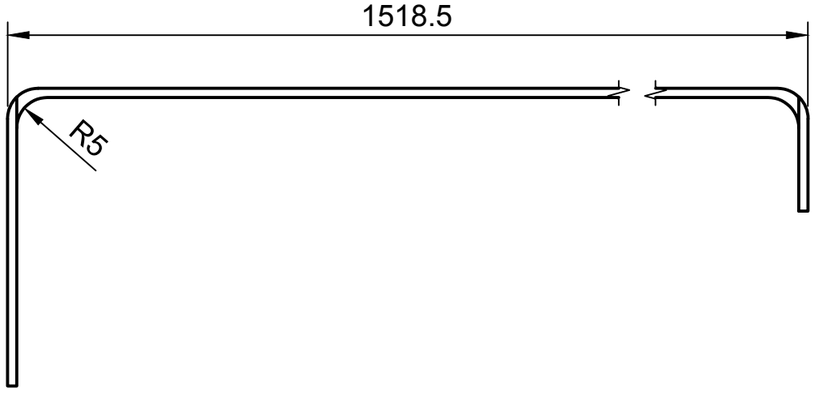




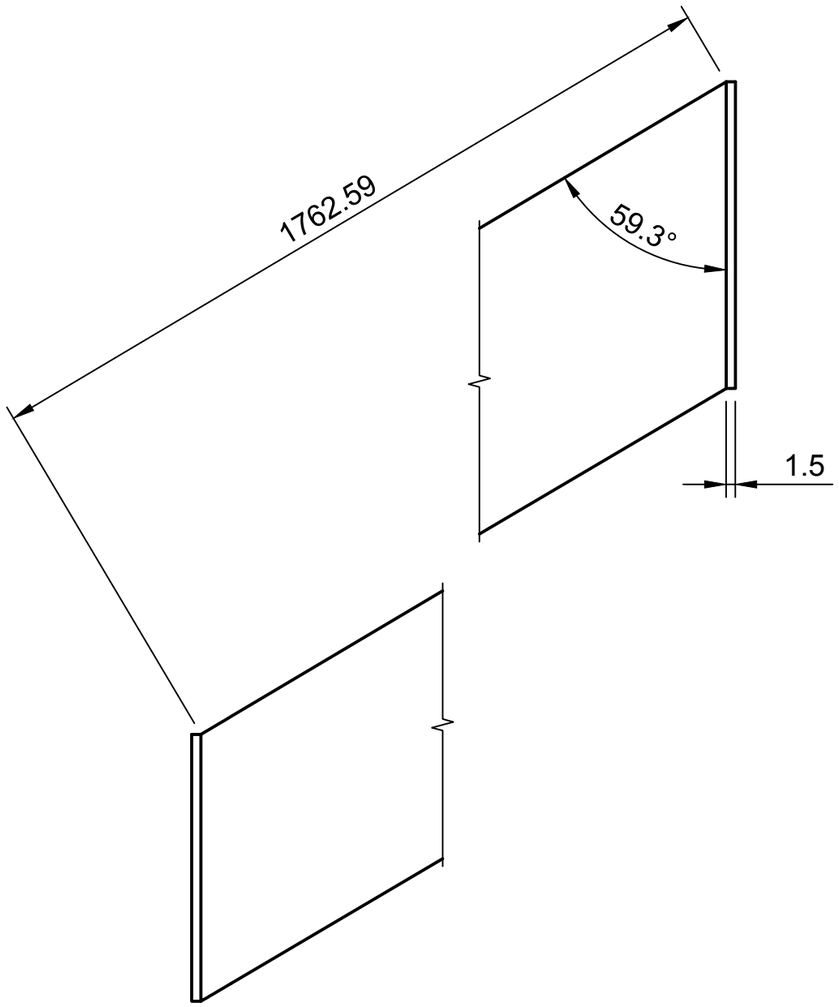
Dept.	Technical reference	Created by	Approved by
Diseño	per_lat_cor	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:1		Title: Perfil lateral corto	DWG No.: 12.1.2
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 16/24

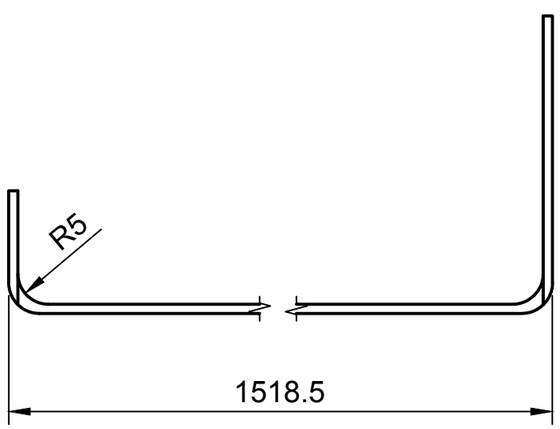


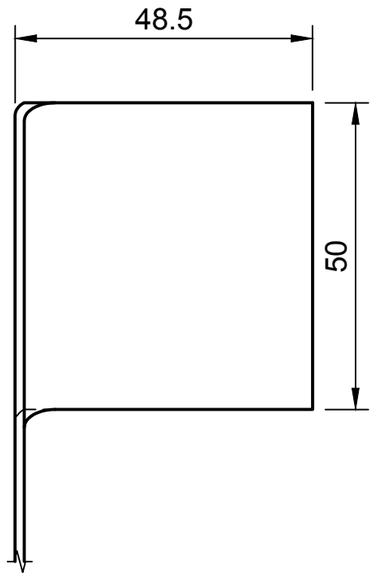


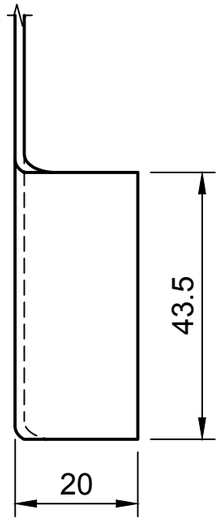


Dept.	Technical reference	Created by	Approved by
Diseño	per_dia_izq	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:1		Title: Perfil diagonal izquierdo	DWG No.: 12.1.3
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 17/24

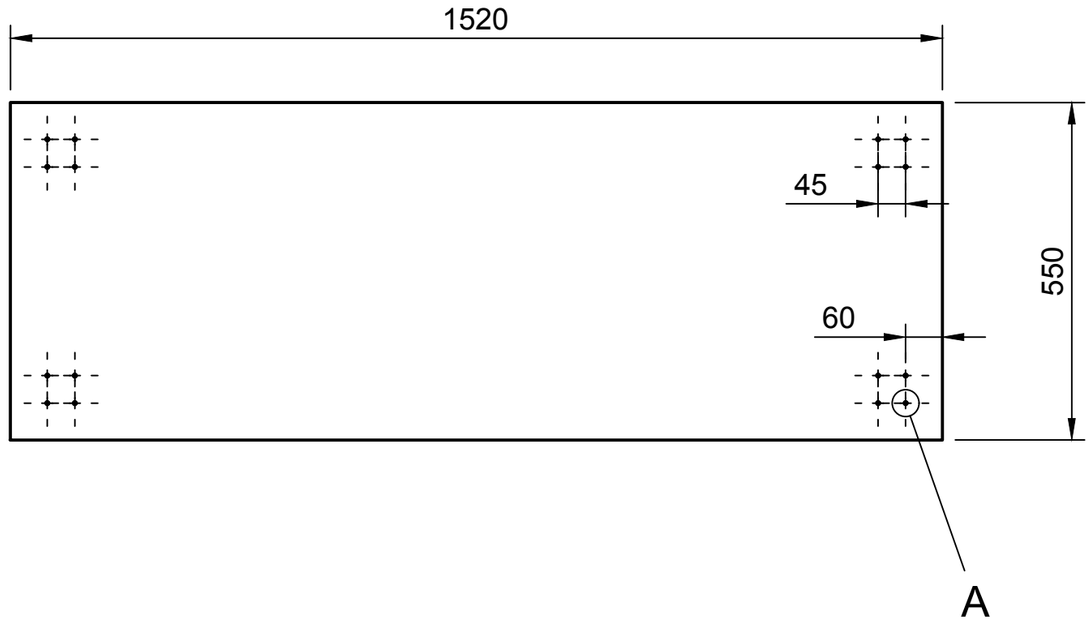


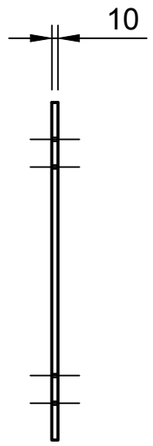


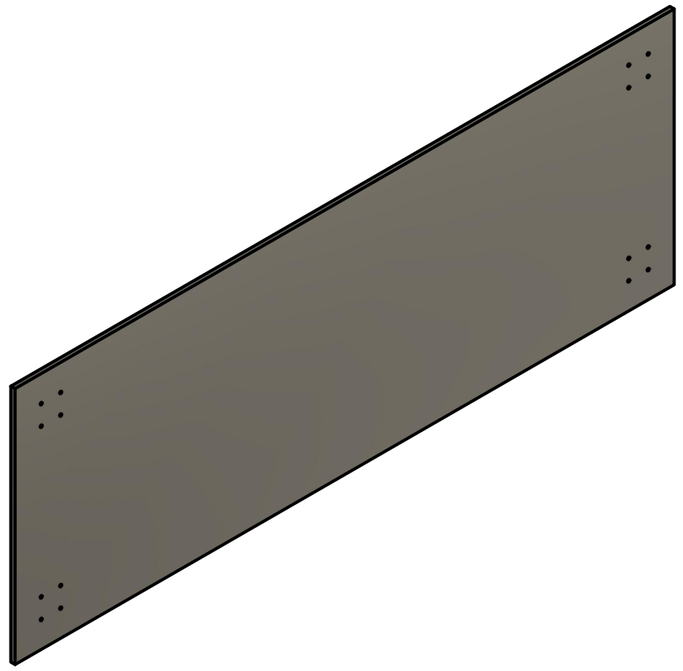




Dept.	Technical reference	Created by	Approved by
Diseño	per_dia_der	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:1		Title: Perfil diagonal derecho	DWG No.: 12.1.4
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 18/24

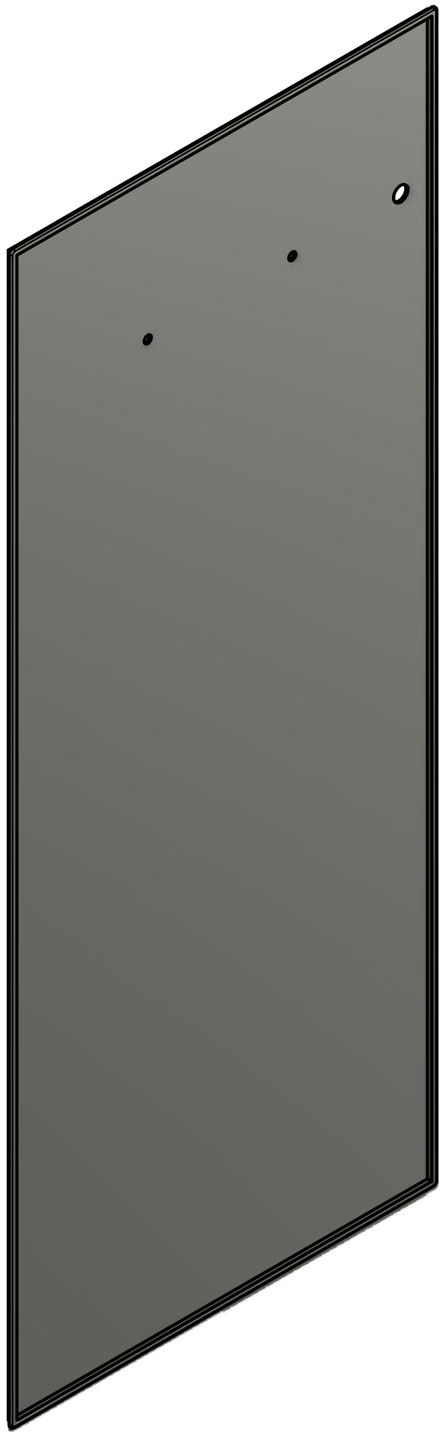


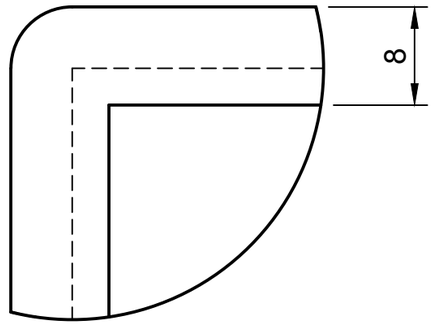




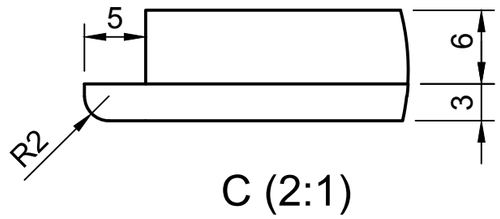
A (2:1)

Dept.	Technical reference	Created by	Approved by
Diseño	bas_car	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:10		Title: Base de la carretilla	DWG No.: 12.1.5
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 19/24

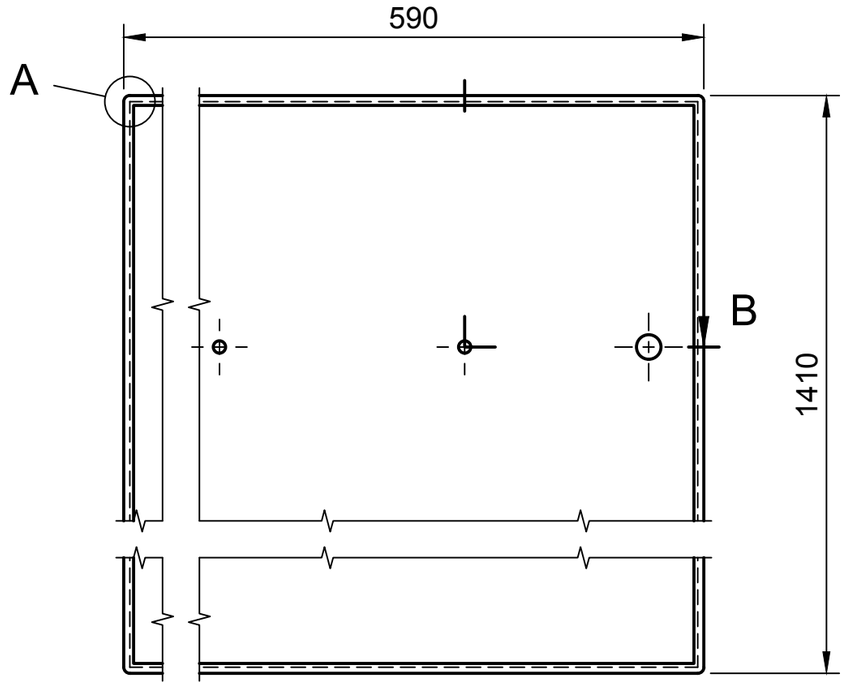


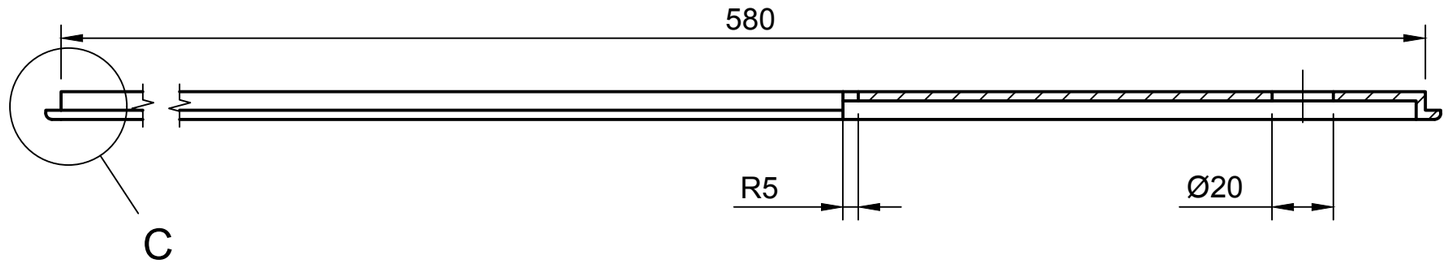


A (2:1)



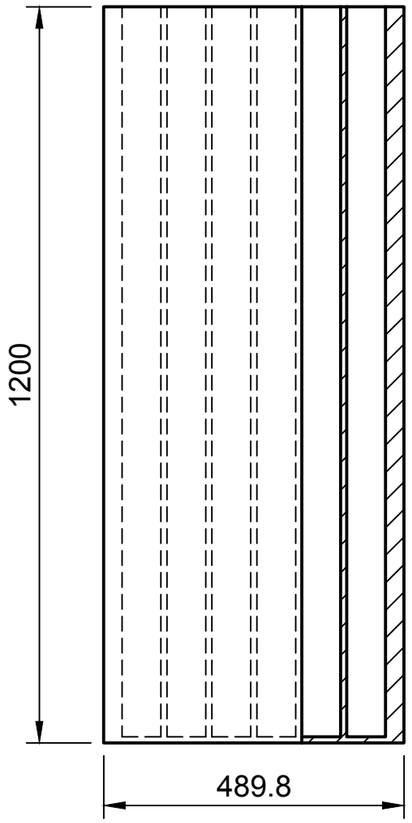
C (2:1)

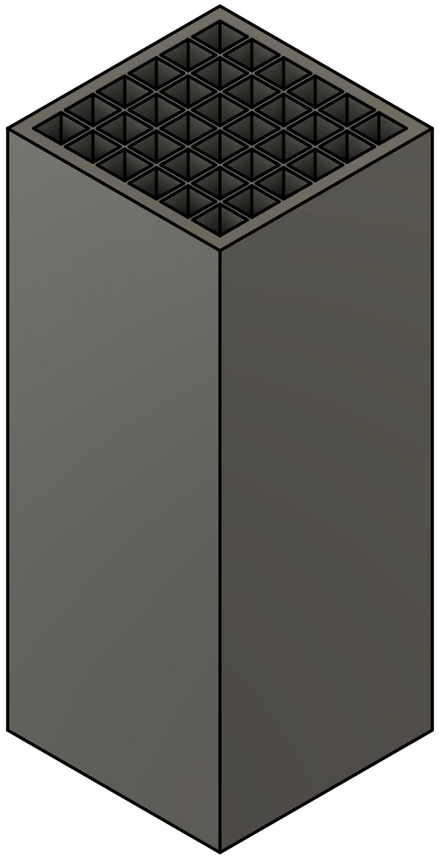


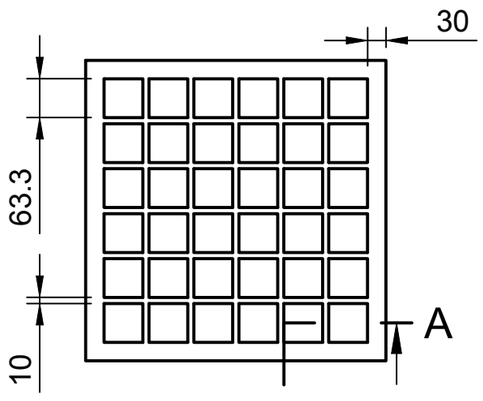


B-B (1:2)

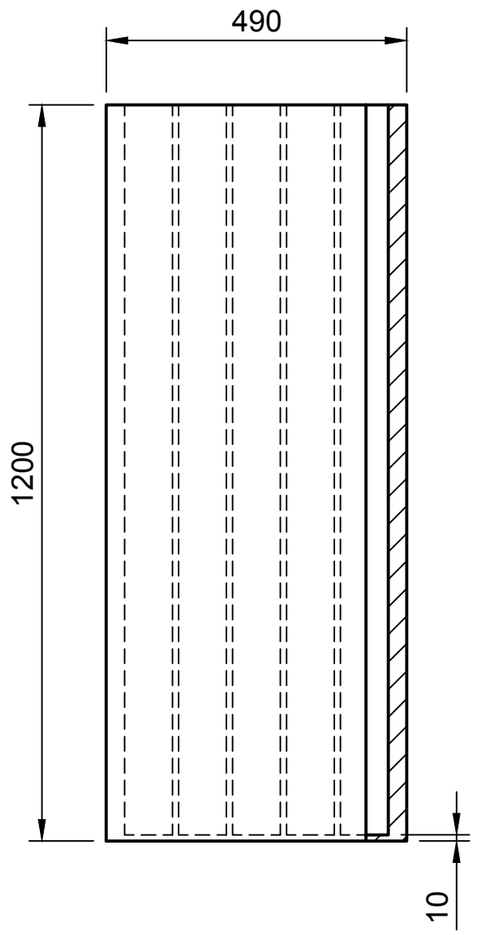
Dept.	Technical reference	Created by	Approved by
Diseño	pue_car	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:5		Title: Puerta de la carretilla	DWG No.: 12.1.6
			Rev.: - / Date of issue: 20/03/2036 / Sheet: 20/24

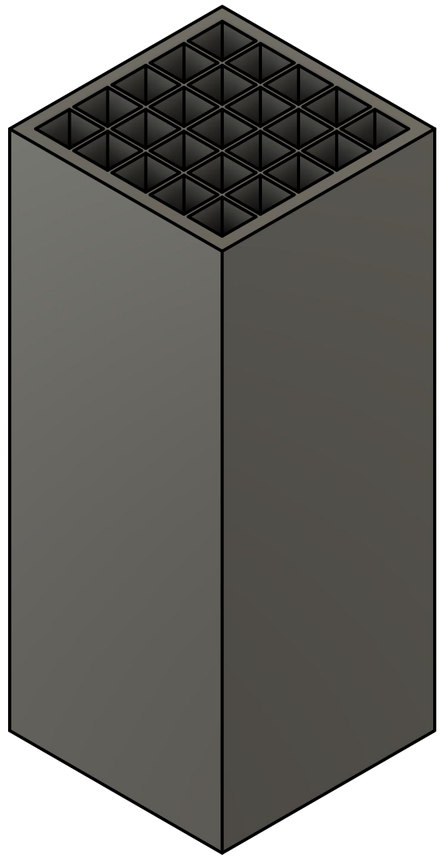


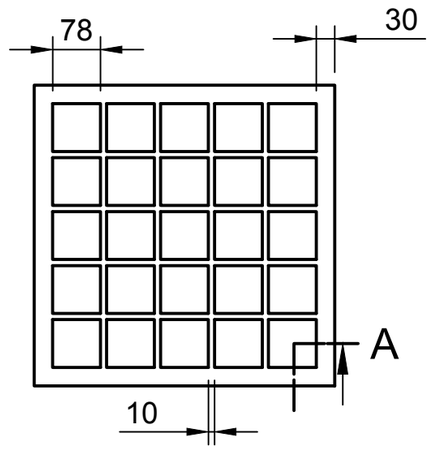




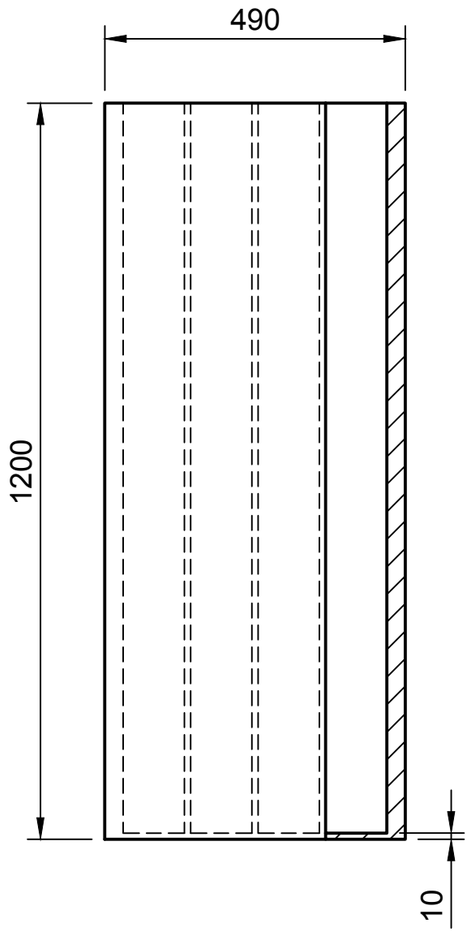
Dept.	Technical reference	Created by	Approved by
Diseño	con_bot_peq	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:10		Title: Contenedor botellas pequeñas	DWG No.: 12.2
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 21/24

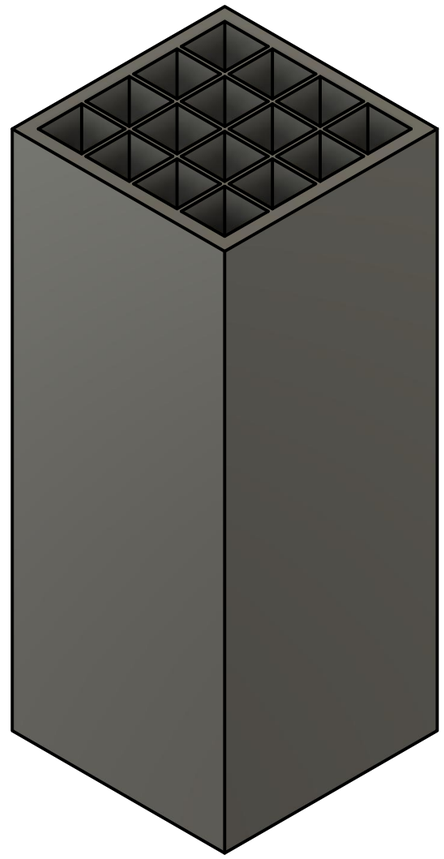


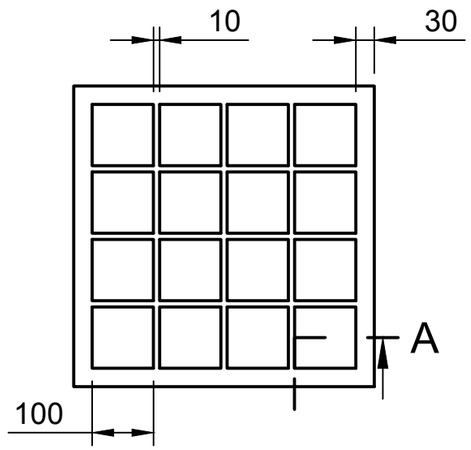




Dept.	Technical reference	Created by	Approved by
Diseño	con_bot_med	Botendado	Mercadona
1:10		Document type: Plano de pieza	Document status: APPROVED
		Title: Contenedor botellas medianas	DWG No.: 12.3
			Rev.: - / Date of issue: 20/03/2035 / Sheet: 22/24







Dept.	Technical reference	Created by	Approved by
Diseño	con_bot_gra	Botendado	Mercadona
		Document type: Plano de pieza	Document status: APPROVED
1:10		Title: Contenedor botellas grandes	DWG No.: 12.4

Rev.	Date of issue	Sheet
-	20/03/2035	23/24

Parte II

Pliego de Condiciones:

Índice:

1. Objeto:

El objeto del presente Trabajo de Fin de Grado es utilizar redes neuronales para la detección de diferentes tipos de envases. Además, se indagará en la documentación para la creación de una base de datos propia de envases reciclables con sus correspondientes anotaciones en el formato pertinente.

2. Condiciones de los Materiales:

2.1. Descripción

2.1.1. Equipo informático:

Para el desarrollo del presente proyecto solo se hará uso de material informático (hardware y software).

El hardware utilizado podrá ser todo aquél capaz de tener instalado la versión de Python 3.7 o superior, así como PyTorch 1.7 o superior. Este puede ser utilizado en los sistemas operativos de macOS, Windows y Linux. En el caso de macOS la versión de este no será inferior a macOS Catalina 10.15. Para Windows, la versión deberá ser no inferior a Windows 7. Finalmente, para Linux, su versión mínima no deberá ser menor a glibc v2.17. Es recomendable un ordenador con GPU, de lo contrario los tiempos de entrenamiento podrían aumentar considerablemente. El equipo deberá contar con una memoria RAM no inferior a 32 GB. Asimismo, la memoria de disco tampoco deberá ser menor a 32 GB, pues es necesaria para almacenar la base de datos de imágenes y los scripts de programación.

Las librerías necesarias serán las siguientes: Ultralytics, Os, PyLabel, OpenCV y NumPy.

Para la anotación de la base de datos de imágenes se podrá hacer uso de cualquier software online o local que permita exportar los ficheros de anotaciones en formato YOLO. Entre los recomendados se encuentran: CVAT.AI y Roboflow.

2.1.2. Datasets con muestra de imágenes:

Las imágenes que forman parte del dataset de este proyecto deberán estar anotadas en formato YOLO. De esta manera, por cada imagen utilizada en el entrenamiento, validación y test, debe existir un archivo .txt homónimo al archivo de imagen que contenga la información de las bounding boxes dibujadas durante el proceso de anotación.

Los archivos de texto con las anotaciones de cada imágen deberán seguir el siguiente formato para describir las bounding boxes: `class ; x ; y ; width ; height.`

- `class`: Este es un número entero que sirve para identificar la clase del objeto que está siendo etiquetado. Estos números enteros deberán ir desde cero (0) hasta el número de clases totales menos uno (nº clases - 1).

- `x y`: Coordenadas **x** e **y** del punto central de la bounding box en valores relativos en un rango de números reales desde 0 hasta 1.
- `width height`: **Ancho** y **alto** relativo de la bounding box con respecto al ancho y alto de la imágen.

Tanto las imágenes como las anotaciones deberán de estar dispuestas en tres carpetas: *train*, val y *test*. Las imágenes serán distribuidas entre estas tres carpetas según las proporciones que se decida antes del proceso de entrenamiento pero en ningún caso se puede dejar alguna de estas carpetas vacías. Dentro de cada carpeta se deberán disponer a su vez dos subcarpetas llamadas: *images* y *labels*. En la primera se copiaran los archivos de imagen y en la segunda, sus correspondientes anotaciones. Todas estas carpetas estarán contenidas a su vez en una carpeta llamada *data*, dentro del directorio *code*, donde también se deberán encontrar los archivos `main.py` y `data.yaml`.

El archivo `main.py` deberá contener el código para el entrenamiento, validación y test de la red neuronal.

El archivo `data.yaml` deberá contener la ruta global de la carpeta *data* donde se encuentra, así como las rutas relativas donde se encuentran los subsets de entrenamiento, validación y test. Además, contendrá los nombres de las clases enumerados empezando por el 0.

2.2. Control de Calidad:

La comprobación de los requisitos de hardware se llevará a cabo de la siguiente manera dependiendo del sistema operativo en el que se trabaje:

- En macOS: Se hará click en el icono con el logo de Apple en la esquina superior izquierda de la pantalla. Tras abrirse el menú desplegable, se seleccionará la opción "Acerca de este mac" y se comprobará que cumple los requisitos anteriormente mencionados.
- En Windows: Se presionarán las teclas "Windows + R" en el teclado y en la ventana "Ejecutar"se deberá escribir: "dxdiag".
- En Linux: Se deberá escribir "lshw" sobre la línea de comandos.

Para comprobar los requisitos de software, en un terminal se escribirá: `pip list`. Aparecerá una lista con las librerías instaladas y sus versiones, donde deberán aparecer las librerías mencionadas anteriormente.

Para comprobar la versión de Python, se escribirá en el terminal: `python −version`.

3. Condiciones de Ejecución del Proyecto:

3.1. Descripción:

3.1.1. Obtención de las imágenes:

El tamaño de las imágenes no será en ningún caso inferior a 512 × 512 píxeles. Estas no deberán estar pixeladas y deberán contener al menos un tipo de envase por imagen, con un máximo de 300 por imagen.

Se podrán utilizar diferentes medios para la obtención de estas imágenes. Estos pueden ser bases de datos que estén a la libre disposición del público, imágenes propias e imágenes con derechos reservados siempre que estas hayan sido debidamente cedidas para el desarrollo del proyecto en cuestión.

3.1.2. Etiquetado de las imágenes:

Todas las imágenes que forman parte de la base de datos deberán de estar correctamente anotadas, de forma que las cajas delimitadoras contengan la totalidad del objeto a detectar. Asimismo, las cajas deben de ser de aproximadamente el tamaño mínimo para que el objeto sea contenido en su totalidad, sin que esta sea excesivamente grande y abarque más área de la necesaria.

3.1.3. Entrenamiento y validación de la red:

Antes del entrenamiento de la red, se deberán fijar los parámetros de entrenamiento: `epochs`, `batch`, `device`, `patience` `val`, `imgsz`, y `data`. Los cuatro primeros dependerán de las necesidades del entrenamiento, mientras que `val` deberá ser `True`, `imgsz` no tendrá un valor inferior a `512` y `data` contendrá la ruta global del archivo `data.yaml`.

El entrenamiento de la red se deberá llevar a cabo con la porción de la base de datos contenida en la carpeta "*train*". Se cargarán también los pesos de la red preentrenada de YOLO en el modelo.

La validación, en cambio, con la porción de la base de datos contenida en la carpeta "*val*". El proceso de validación será llevado a cabo sobre el archivo `best.pt`, resultado del proceso de entrenamiento localizado en la carpeta de "*weights*".

3.2. Control de la ejecución:

Se deberá comprobar que ninguna de las imágenes seleccionadas para formar parte de la base de datos tenga tamaño inferior a 512×512 píxeles, esté pixelada, no disponga de los permisos para ser utilizada en caso de tener derechos de autor. De igual manera, se deberá comprobar que contenga al menos un envase y no más de 300. En caso de no cumplir alguno de estos requisitos, se deberá excluir dicha imagen de la base de datos.

Durante los procesos de entrenamiento y validación se deberá comprobar en la consola que las imágenes se cargan correctamente al inicio de cada uno de ellos. En caso contrario, se deberá buscar un código de error que explique el por qué de este fallo y remediarlo.

Tras el proceso de entrenamiento se debe comprobar que los pesos se han guardado en la carpeta "*weights*" dentro del directorio de los resultados del entrenamiento. Estos pesos serán los utilizados para la validación.

4. Prueba de Servicio:

El test final de la red se llevará a cabo con la porción de la base de datos contenida en la carpeta "*test*". El modelo que se utilizará para el test deberá ser el resultado del proceso de validación, es decir, el archivo "`best.pt`".

Los argumentos para el test estarán fijados según se considere mejor teniendo en cuenta las características del entrenamiento.

Para que el entrenamiento se considere satisfactorio, los resultados de test deberán obtener un valor para la mAP por clase no menor a 0.75, de lo contrario se considerará que la red no ha sido entrenada correctamente para esas clases.

Parte III

Presupuesto:

Índice:

1. Precios Elementales:

Debido a la naturaleza de este proyecto y que la mayoría de los recursos utilizados son virtuales y de licencia gratuita, los costes para llevarlo acabo radican en la amortización de los equipos informáticos utilizados y la mano de obra, en este caso de un ingeniero técnico industrial.

Ref.	**Unidad**	**Descripción**	**Precio (€)**
		Maquinaria	
e1	ud.	MacBook Pro Chip M2 14"	233,87
e2	ud.	iPhone XR 128GB	85,90
		M.O.D.	
h1	h	Ingeniero técnico industrial	17,00

Cuadro 13: Precios Elementales

El coste de la amortización del ordenador portátil y del teléfono móvil (utilizado en la obtención de parte de las imágenes que componen la base de datos), se ha calculado de la siguiente manera:

$$\text{Coste}_{(e1)} = \frac{3742{,}41 \cdot 0{,}5 \text{ años}}{8 \text{ años}} = 233{,}87 \text{ €} \tag{3}$$

$$\text{Coste}_{(e2)} = \frac{859{,}00 \cdot 0{,}5 \text{ años}}{5 \text{ años}} = 85{,}90 \text{ €} \tag{4}$$

2. Resumen del Presupuesto completo:

Para el computo de las horas trabajadas por el ingeniero técnico industrial durante el desarrollo de este proyecto se ha tomado como base el número de créditos matriculados en el TFG. De acuerdo a que este tiene un valor de 12 ECTS y cada crédito ECTS corresponde a 25 horas de trabajo, la suma final asciende a 300 horas trabajadas.

Ref.	**Unidad**	**Descripción**	**Precio**	**Cantidad**	**Total (€)**
Maquinaria					
e1	ud.	MacBook Pro Chip M2 14"	233,87	1	233,87
e2	ud.	iPhone XR 128GB	85,90	1	85,90
M.O.D.					
h1	h	Ingeniero técnico industrial	17	300	5100
Medios Auxiliares					
%		M.A. sobre los costes directos	1,5	5419,77	81,30
TOTAL (Presupuesto de ejecución material)					**5501,07**
Gastos Generales				6 %	330,06
Beneficio Industrial				13 %	715,14
Suma de G.G. y B.I.					**1045,20**
TOTAL (Presupuesto de ejecución por contrata					6546,27
I.V.A.				21 %	**1374,72**
PRESUPUESTO TOTAL					**7920,99**

Cuadro 14: Prespuesto Completo

El presupuesto total del presente proyecto (*Reconocimiento óptico de envases mediantes redes neuronales en un contexto de economía circular*) asciende a un total de SIETE MIL NOVECIENTOS VEINTE EUROS Y NOVENTA Y NUEVE CÉNTIMOS.