UNIVERSIDAD DE VALLADOLID
E.T.S.I DE TELECOMUNICACIÓN

TRABAJO FINAL DE GRADO
GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Desarrollo de un Modelo de Inteligencia Artificial para la Detección Automática de Productos en un Frigorífico

Autor: Mario Laustalet Fernández
Tutor: Alfonso Bahillo Martínez

5 de septiembre de 2023

TÍTULO:	Desarrollo de un Modelo de Inteligencia Artificial para la Detección Automática de Productos en un Frigorífico
AUTOR:	Mario Laustalet Fernández
TUTOR:	Alfonso Bahillo Martínez
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	Ramón J. Durán Barroso
VOCAL:	Juan C. Aguado Manzano
SECRETARIO:	D. Alfonso Bahillo Martínez
SUPLENTE:	Noemí Merayo Álvarez
SUPLENTE:	Rubén M. Lorenzo Toledo
FECHA:	11 de septiembre de 2023
CALIFICACIÓN:	

Resumen

En este Trabajo de Fin de Grado (TFG), se busca desarrollar un modelo de inteligencia artificial que permita realizar una detección de productos en un frigorífico utilizando imágenes enviadas por un sistema IoT cada vez que se abre el frigorífico. Este proyecto se centra en la transformación de un frigorífico convencional en uno inteligente, haciéndolo económicamente asequible. Si bien tiene múltiples aplicaciones, una de sus principales funciones es el seguimiento de los hábitos alimenticios. El proyecto comenzó realizando reconocimiento de imágenes mediante ResNet50, con una amplia base de datos de imágenes de alimentos. Sin embargo, este enfoque se encontró con limitaciones, ya que solo podía reconocer un alimento por imagen. Para superar esta limitación, se realizó detección de objetos mediante YOLO (You Only Look Once). Se creó una nueva base de datos de imágenes de alimentos, que se etiquetaron a través de la herramienta en línea makesense.ai. Este conjunto de datos etiquetados se utilizaron para entrenar el algoritmo de detección de objetos YOLO. Los resultados muestran que el nuevo algoritmo es capaz de detectar múltiples alimentos en una imagen, lo que mejora significativamente su utilidad y aplicabilidad en comparación con el enfoque inicial. Esta investigación muestra el potencial que tiene la inteligencia artificial y el IoT para contribuir a la salud y el bienestar en la vida diaria.

PALABRAS CLAVE: Inteligencia Artificial, Reconocimiento de imágenes, Detección de objetos, Frigorífico, Internet de las cosas, Alimentos, Salud mental, Modelos de aprendizaje profundo

Abstract

In this Bachelor's Final Project, the goal is to develop an an artificial intelligence model that allows for the detection of products inside a refrigerator using images sent by an IoT system every time the refrigerator is opened. This project focuses on transforming a conventional refrigerator into a smart one, making it economically viable. Although it has multiple applications, one of its primary functions is monitoring eating habits. The project started by performing image recognition through ResNet50, with a comprehensive database of food images. However, this approach encountered limitations, as it could only recognize one food per image. To overcome this limitation, object detection was performed using YOLO (You Only Look Once). A new food image database was created, which was labeled through the online tool makesense.ai. This labeled dataset was used to train the YOLO object detection algorithm. The results show that the new algorithm can detect multiple foods in an image, significantly enhancing its utility and applicability compared to the initial approach. This research demonstrates the potential of artificial intelligence and IoT in contributing to health and well-being in everyday life.

KEYWORDS: Artificial Intelligence, Image Recognition, Object Detection, Refrigerator, Internet of Things (IoT), Food products, Mental health, Deep learning models

Agradecimientos

En primer lugar, quisiera expresar mi más sincero agradecimiento a mi tutor Alfonso Bahillo Martínez, por su aportación a lo largo de todo este proyecto.

Quiero agradecer a mi compañero y amigo Ignacio González por realizar juntos este proyecto, apoyándonos mutuamente tanto en el aspecto técnico como motivacional.

También debo agradecer a mis amigos por estar a mi lado durante este viaje académico, brindándome apoyo moral.

No me puedo olvidar de mi familia, los que han estado todos los días de este viaje que comencé en 2018.

Quiero agradecer a todos aquellos que, de una forma u otra, contribuyeron a la realización de este trabajo, desde quienes participaron en pruebas, hasta quienes me dieron un simple consejo o palabra de aliento, sobre todo quiero agradecer a Andrés Martín, Alberto Sánchez y Victoria Pacho en este sentido.

Por úlitmo, quiero dedicar un especial agradecimiento a mí mismo. A pesar de los desafíos y obstáculos que surgieron a lo largo de este viaje académico, demostré determinación y perseverancia. Finalizar esta carrera no solo es el culmen de años de estudio, sino también la prueba de mi compromiso y pasión por el aprendizaje. Estoy orgulloso de lo que he logrado y agradecido por todo lo que he aprendido sobre mí mismo en el proceso.

Gracias a todos por ser parte esencial de este significativo capítulo en mi vida académica.

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

1.1 Contextualización del Proyecto

La Inteligencia Artificial (IA) y el Internet de las Cosas (IoT) están revolucionando muchos aspectos de nuestra vida. Lo novedoso y de lo que todo el mundo habla son los chatbots como ChatGPT o Bard. Sin embargo, para otras aplicaciones como el uso doméstico, la combinación de la IA y el IoT ha dado lugar a casas inteligentes, donde los electrodomésticos y aparatos electrónicos pueden interactuar entre sí, recopilando y analizando datos para mejorar la calidad de vida del usuario.

En este contexto, los frigoríficos inteligentes han surgido como una tecnología innovadora y potencialmente útil. Estos frigoríficos están equipados con cámaras y sensores, y pueden monitorizar su contenido, ayudando a las personas a gestionar su nevera de una manera más eficiente.

Ciertas investigaciones recientes han sugerido que el estado de salud mental de una persona puede reflejarse en el contenido de su frigorífico. Por lo tanto, existe una oportunidad para desarrollar una tecnología asequible económicamente que permita el seguimiento del contenido del frigorífico, no solo para ayudar en la gestión de alimentos, sino también para mostrar los hábitos alimenticios del usuario.

1.2 Frigoríficos Inteligentes en el Mercado Actual

Si bien se puede conocer muchas de las funciones inteligentes integradas en los frigoríficos actuales y el precio de estos, es complicado conocer los detalles técnicos que usan las marcas. Esto se debe a la confidencialidad con la que se rigen en estas empresas, ya que estas soluciones son una parte fundamental de la propuesta de valor de sus productos. Aún así se analizará las funciones y el precio de los frigoríficos inteligentes de las primeras marcas del mercado.

Samsung dispone de su frigorífico family hub. Este frigorífico cuenta con una pantalla táctil de 21,5 pulgadas desde la que se pueden realizar bastantes funciones. Algunas de ellas son visualizar el interior del frigorífico sin abrirlo, escuchar música, ver recetas y conectarse a otros dispositivos como el teléfono móvil. Sin embargo, el family hub no ha implementado detección automática de alimentos. Para conseguir que, por ejemplo, el frigorífico indique cuando se va a caducar un producto, debe indicar el usuario la fecha de caducidad cuando introduzca el producto en su interior.

Esto hace que al final sea un frigorífico con cámaras y un sistema operativo llamado Tizen basado en GNU/Linux. El precio de este frigorífico a día de hoy es de 1852€.

LG también tiene su frigorífico inteligente. Este utiliza ThinQ, una plataforma para tener el hogar conectado. Mediante ThinQ se reciben avisos tales como si el frigorífico está abierto. También se puede controlar la temperatura desde el teléfono móvil. El precio de este frigorífico es de 2404€.

Una vez vistos los frigoríficos de última generación de las principales marcas se puede llegar a la conclusión de que ninguno de ellos plantea la solución de detectar los alimentos de su interior. Por ello, este proyecto puede servir como primer paso para comenzar a dar este gran paso en el mundo del hogar inteligente. Además, la solución que se propone en este proyecto necesita de una arquitectura que tiene un

coste económico relativamente bajo debido a que se necesita una raspberry pi, una cámara y un sensor de puerta.

1.3 Justificación del Proyecto

El uso de IA e IoT a menudo se ve obstaculizada por su elevado coste. En el caso de los frigoríficos inteligentes, si bien ofrecen una serie de funciones útiles, como el seguimiento del contenido y la notificación de alimentos que van a caducar, estas ventajas a menudo se limitan a los frigoríficos de alta gama, que no son accesibles para todos. Además, hasta ahora no se ha desarrollado una tecnología que permita usar esta información para ayudar a gestionar ciertos problemas de salud mental. El desarrollo de esta tecnología no solo tiene el potencial de aportar a nuestro estado de bienestar más tecnología para nuestra vida cotidiana, sino que también abre la puerta a expertos para entender y monitorizar el hábito alimenticio. Esto es especialmente relevante en el contexto actual, en el que los hábitos alimenticios se han convertido en un tema de gran importancia en el mundo en el que vivimos.

1.4 Objetivos del TFG

Desarrollar un modelo de Inteligencia Artificial que permita el seguimiento del contenido de un frigorífico, proporcionando una herramienta que pueda ser utilizada para gestionar eficientemente los alimentos y como un indicador potencial del estado de salud mental de una persona. Los objetivos específicos incluyen:

- Investigar y seleccionar el algoritmo de inteligencia artificial más adecuado para detectar alimentos a partir de imágenes.
- Crear una base de datos de imágenes de alimentos que se utilizará para entrenar y validar el modelo de IA.
- Etiquetar cada imagen en la base de datos, seguido de un proceso de entrenamiento, testeo y validación del modelo de IA.
- Evaluar el rendimiento del modelo de inteligencia artificial en base a ciertas métricas, como la precisión y el recall.

CAPÍTULO 2

MARCO TEÓRICO

2.1 La inteligencia artificial y sus aplicaciones prácticas

La inteligencia artificial ha revolucionado nuestra vida cotidiana y cómo interactuamos con la tecnología. En términos generales, la inteligencia artificial se refiere a la habilidad de una máquina de presentar las mismas capacidades que los seres humanos, como el razonamiento, el aprendizaje, la creatividad y la capacidad de planear [8].

Existen muchas aplicaciones prácticas de la IA y seguirán aumentando a medida que la tecnología avanza. Aquí se detallan algunas de las principales aplicaciones que son relevantes para el proyecto:

• Reconocimiento de Imágenes: es la rama de la IA que se ocupa de enseñar a los ordenadores a reconocer y entender las imágenes de manera similar a la forma en que los humanos lo hacemos [15]. Un paso más allá del reconocimiento de imágenes es la detección de objetos. No solo identifica varios objetos en una imagen, también indica dónde están ubicados. Esto se utiliza en muchas aplicaciones, como en vehículos

autónomos y el diagnóstico de enfermedades. En este caso, se utiliza la IA para identificar diferentes alimentos en las imágenes del frigorífico.

• Aprendizaje Automático: es un subconjunto de inteligencia artificial que permite que un sistema aprenda y mejore de forma autónoma mediante redes neuronales y aprendizaje profundo, sin tener que ser programado explícitamente, a través de la ingesta de grandes cantidades de datos [1]. En el proyecto, se entrena el algoritmo de IA con una gran cantidad de imágenes de productos que hay en un frigorífico, permitiéndole aprender a identificar diferentes tipos de alimentos de forma autónoma.

• IoT (Internet de las cosas): La IA es una pieza fundamental en el desarrollo de IoT, ya que permite a los dispositivos interconectados operar de manera más inteligente y autónoma. En el proyecto, se integra el algoritmo de IA en un sistema IoT que recopila y analiza imágenes del interior de un frigorífico.

• Salud Mental: La IA también tiene aplicaciones en el campo de la salud mental. A través del análisis de patrones y datos, puede proporcionar información valiosa sobre el estado de salud mental de una persona. En este caso, se utilizan las imágenes del contenido del frigorífico como indicadores potenciales del estado mental del usuario.

2.2 Internet de las cosas (IoT) y su uso en electrodomésticos

El IoT es el proceso que permite conectar los elementos físicos cotidianos al Internet. Pueden ser prendas, accesorios, dispositivos médicos, objetos domésticos, etc [9].

Dentro de estos últimos, en la industria de los electrodomésticos, esta interconexión ha dado lugar a lo que se conoce como electrodomésticos inteligentes. Estos aparatos, gracias a las características de IoT, ofrecen funciones mejoradas, eficiencia energética, mayor comodidad y, en muchos casos, la posibilidad de integrarse con otros sistemas y dispositivos del hogar.

Por ejemplo, en la cocina, los refrigeradores inteligentes pueden controlar su propio contenido, sugerir recetas basadas en los ingredientes disponibles, alertar a los usuarios sobre alimentos a punto de caducar e incluso realizar pedidos en supermercados. No obstante, estas funcionalidades avanzadas suelen estar presentes en modelos de alta gama, no siendo accesibles para todos los usuarios.

Mediante el uso de dispositivos de bajo coste en electrodomésticos convencionales, es posible adquirir funciones similares a las de los electrodomésticos inteligentes sin el elevado coste que implica reemplazar los electrodomésticos existentes. Precisamente esto es el foco del proyecto, conseguir convertir un frigorífico convencional en un frigorífico lo más "inteligente" posible.

La solución parte de un proyecto paralelo desarrollado por un compañero, Ignacio González Domínguez. Este proyecto utiliza una Raspberry Pi, un sensor y una cámara para capturar imágenes del interior de un refrigerador convencional. Cada vez que se abre el frigorífico, el sistema toma una foto y la guarda en la Raspberry.

Este proyecto se encarga de tomar estas imágenes y analizarlas mediante un algoritmo de IA para proporcionar información sobre el contenido del frigorífico. De esta manera, se puede proporcionar las funcionalidades de un frigorífico inteligente sin la necesidad de adquirir un nuevo electrodoméstico.

2.3 La relación entre el estado de salud mental y el contenido del frigorífico

Varios estudios han mostrado una relación entre la calidad general de la dieta y el riesgo de depresión. Por ejemplo, una revisión de 21 estudios de 10 países encontró que un patrón dietético saludable, caracterizado por un alto consumo de frutas, verduras, granos enteros, aceite de oliva, pescado, productos lácteos bajos en grasa y antioxidantes, y un bajo consumo de alimentos de origen animal, se asoció con un menor riesgo de depresión. Por el contrario, una dieta de estilo occidental, que implica un alto consumo de carnes rojas y procesadas, granos refinados, dulces, productos lácteos altos en grasa, mantequilla y patatas, así como un bajo consumo de frutas y verduras, se asoció con un riesgo significativamente mayor de depresión [16].

Además, la investigación en la que se examina si las intervenciones dietéticas pueden ayudar a tratar los problemas de salud mental es relativamente nueva y aún bastante limitada. Sin embargo, algunos estudios han demostrado mejoras prometedoras en los síntomas de la depresión a través de intervenciones dietéticas, aunque este campo de estudio necesita más investigación para formar conclusiones sólidas [16].

Por lo tanto, parece que lo que se encuentra en el frigorífico (es decir, la dieta de una persona) puede tener un impacto en tu salud mental. Sin embargo, es importante tener en cuenta que la investigación en esta área todavía está en desarrollo, y muchos estudios pueden mostrar asociaciones, pero no pueden probarlo al completo.

En este proyecto, se busca utilizar la inteligencia artificial para proporcionar en detalle lo que las personas tienen en sus frigoríficos. El objetivo es desarrollar una herramienta que pueda ayudar en el seguimiento de los hábitos alimenticios a través de la monitorización del contenido del frigorífico lo cual puede ayudar a mejorar la calidad de vida de personas con problemas de salud mental de una forma accesible.

Es importante mencionar que, aunque el seguimiento del contenido del frigorífico puede proporcionar indicadores sobre la salud mental, también puede servir para una variedad de otras funciones. Estas funciones podrían incluir el control de peso o el manejo de enfermedades crónicas, la prevención de desperdicios de alimentos mediante la identificación de productos caducados, e incluso la creación automática de la lista de la compra o sugerencias de recetas basadas en los alimentos disponibles en el frigorífico.

CAPÍTULO 3

METODOLOGÍA INICIAL Y PROBLEMAS ENCONTRADOS

3.1 Introducción y comparativa de algoritmos de IA

Antes de adentrarnos en la creación de un modelo para reconocimiento de imágenes, es fundamental comprender las diferentes clases de algoritmos de IA disponibles y cómo se comparan entre sí:

- Aprendizaje no supervisado: Utiliza algoritmos de machine learning para analizar y agrupar en conjuntos de datos sin etiquetar. Estos algoritmos descubren agrupaciones de datos o patrones ocultos sin necesidad de ninguna intervención humana [2].

- Aprendizaje supervisado: Se define por su uso de conjuntos de datos etiquetados para entrenar algoritmos que clasifican datos o prevén resultados con precisión. A medida que se introducen datos en el modelo, ajusta sus ponderaciones hasta que el modelo se adapte correctamente. Se utilizan varios algoritmos y técnicas de cálculo. Entre ellas:
  - Máquinas de vectores de soporte (SVM): Este es un algoritmo de clasificación que busca el hiperplano que mejor divide un conjunto de datos en clases.
  - Árboles de decisión (DT) y Bosques aleatorios (RF): Estos son algoritmos que dividen el espacio de características en regiones. Los RF combinan múltiples árboles para mejorar la precisión.
  - Redes Neuronales (CNN): Procesan los datos de entrenamiento utilizando la interconectividad de las capas de nodos del cerebro humano. Cada nodo consta de entradas, ponderaciones, un sesgo (o umbral) y una salida. Si el valor de salida supera un límite, se activa el nodo, lo que transmite los datos a la siguiente capa de la red. Las CNN son especialmente poderosas para tareas relacionadas con la visión por computadora, como clasificación de imágenes, detección de objetos y segmentación de imágenes [3].

3.2 Justificación del primer modelo elegido (ResNet50)

Como este proyecto se basa en el reconocimiento y clasifiación de imágenes, las CNN son la mejor elección dentro de los algoritmos de aprendizaje supervisados. Inicialmente, se buscó qué arquitecturas servían para haCer un reconocimiento de imágenes. De entre las posibilidades se eligió ResNet50 debido a las siguientes razones:

- Buen rendimiento en clasificación de imágenes: ResNet50 es conocido por ser realmente bueno identificando y clasificando imágenes, aunque posteriormente se vió que no se debía clasificar imágenes en categorías, sino detectar y clasificar los objetos dentro de la imagen.
- Estructura de red residual: Esto facilita el entrenamiento de redes profundas al abordar el problema del desvanecimiento del gradiente [1]. Permite que la red

[1]Desvanecimiento de gradiente: es un problema en el que el gradiente se va desvaneciendo a valores muy pequeños, impidiendo eficazmente el peso de cambiar su valor, llegando, incluso a que la red neuronal continúe su entrenamiento

aprenda de manera efectiva y alcance una precisión más alta en tareas complejas [7].

- Equilibrio entre rendimiento y velocidad: En este proyecto en el que la velocidad de inferencia no es algo crítico, ResNet50 es una buena opción.

Hay que recordar que esta es la primera arquitectura elegida y como se explicará en futuras secciones, no fue la correcta.

3.3 Preparación y procesamiento de la base de datos

Para realizar este proyecto, era esencial contar con un conjunto de datos lo suficientemente diverso para el reconocimiento de imágenes de alimentos. Se eligió utilizar la base de datos Food-101, disponible en la plataforma Kaggle [6], que es conocida y utilizada comúnmente para tareas de reconocimiento de imágenes. Food-101 es una base de datos que consta de 101000 imágenes con 101 categorías de alimentos. Cada categoría tiene 1000 imágenes, lo que ofrece un equilibrio entre las diferentes clases de alimentos.

También fue utilizada otra base de datos llamada fruits-360, disponible también en Kaggel [5], a diferencia de la anterior, esta base de datos se centra en clases de frutas.

La unión de ambas es una base de datos de alimentos con una amplia cantidad de imágenes. Sin embargo, las imágenes de esta base de datos vienen en varios tamaños. Por lo tanto, fue necesario realizar un paso de preprocesamiento para asegurar que todas las imágenes tuvieran las mismas dimensiones, un requisito clave para entrenar modelos. Para conseguirlo, se utilizó un script en Python llamado *resize.py*, incluido en el repositorio en GitHub[11], que redimensiona todas las imágenes a un tamaño fijo de 384x216 píxeles. El script recorre cada imagen en el directorio que se introduzca, carga la imagen, la redimensiona y finalmente la guarda en otro directorio que se introduzca, manteniendo la estructura de subdirectorios original. Este procesamiento no sólo asegura que el tamaño de las imágenes sea el mismo, sino que también reduce el coste computacional al reducir el tamaño de imagen, algo necesario debido a que el dispositivo en el que se iba a realizar el entrenamiento dispone de recursos algo limitados. En concreto, está limitado en la GPU. Se cuenta con una gráfica integrada AMD Radeon. Al no ser una gráfica dedicada, no se dispone del mismo poder de procesamiento. Como la base de datos es bastante grande había que reducir costes

computacionales en el peso de cada imagen. La base de datos se dividió en imágenes destinadas al entrenamiento (train 70 %), imágenes destinadas a la validación del modelo (val 15 %) e imágenes destinadas a ser probadas (test 15 %). Este conjunto de datos fue una buena elección en lo que se refiere al volumen y estructura de datos. Sin embargo, hubo ciertos problemas con la base de datos que comentaremos en una sección futura orientada a problemas.

3.4 Creación y entrenamiento del modelo de clasificación de imágenes

Para la tarea de reconocimiento de alimentos, era crucial seleccionar una arquitectura de modelo que pudiera manejar la diversidad y complejidad de la base de datos. Como el reconocimiento de los alimentos no tenía que ser en tiempo real, lo mejor era partir de un modelo que, aunque el tiempo para reconocer un alimento fuese alto, ofreciese buen rendimiento. Por lo tanto, se optó por utilizar el modelo CNN ResNet50. Además, esta arquitectura también es conocida por ser fácil de entender para principiantes.

En el caso de este proyecto, se utilizó una versión de ResNet50 que ya había sido entrenada con ImageNet, una base de datos de millones de imágenes etiquetadas. Sin embargo, ImageNet no está diseñado específicamente para reconocer alimentos en el interior de un frigorífico. Por lo tanto, se tuvo que adaptar ResNet50 para poder reconocer alimentos.

Para adaptar ResNet50, se añadieron algunas capas adicionales al modelo para que pudiera identificar las características específicas de cada tipo de alimento en la base de datos. Se utilizó una técnica conocida como Transfer Learning. Esta técnica aprovecha el conocimiento que el modelo ResNet50 ha adquirido de ImageNet y lo modifica con una base de datos específica (de alimentos, en este caso). Este proceso permite lograr un mejor rendimiento y precisión en el reconocimiento de alimentos.

Para la implementación del modelo ResNet50 se empleó el script *modeloPrueba-Real2.py*, que se encuentra en el repositorio de GitHub del proyecto[11]. El primer paso es cargar las bibliotecas necesarias para poder trabajar con el modelo. Una vez cargadas las bibliotecas, se importa el modelo ResNet50 y se adapta el modelo al reconocimiento de alimentos.

Posteriormente, se configura el modelo para el entrenamiento y se cargan los datos de entrenamiento y de validación, utilizando una herramienta llamada ImageDataGenerator. Esta herramienta es útil para manejar grandes cantidades de imágenes.

El modelo se entrena ajustando el número de épocas de entrenamiento, que es un parámetro que determina cuántas veces se examina el conjunto de datos de entrenamiento.

3.5 Implementación del modelo para la clasificación de imágenes

Después de entrenar el modelo, se creó el script *diccionario.py* con el fin de realizar un diccionario de clases, recorriendo todo el directorio de la base de datos, mapeando cada índice de la predicción a su etiqueta de clase. Al ser una gran cantidad de clases (118), la creación de un script que guardase automáticamente las etiquetas de clase en un diccionario era la mejor opción.

Finalmente, se elaboró el script *fasetest1.py* para realizar predicciones de una imagen. Este script carga el modelo entrenado y el diccionario de clases, mencionado anteriormente. Luego se carga la imagen objetivo, es decir, la imagen de la que se quiere predecir la clase de alimento. Luego se define una función para predecir la clase de un alimento en la imagen específica, obteniéndose la etiqueta de clase correspondiente al índice del diccionario. Por último, se muestra por pantalla dicha etiqueta de clase.

3.6 Problemas encontrados durante el proceso

Se encontraron dos problemas fundamentales según se desarrollaba este proyecto, que limitaron el funcionamiento del sistema.

El primer problema estaba relacionado con la base de datos de alimentos utilizada. El conjunto de datos era amplio y diverso, pero no representaba bien los tipos de productos que normalmente se encuentran en un frigorífico. La base de datos contenía imágenes de alimentos preparados como guacamole, huevos rancheros, platos de pasta, entre otros. Sin embargo, en un frigorífico, es más común encontrar productos

envasados como botellas, tarros, bricks, etc. La falta de estos productos en la base de datos dificultó el entrenamiento del modelo para reconocer productos en un frigorífico.

Ante la dificultad de encontrar una base de datos de uso libre que se ajustara a mis necesidades, decidí crear mi propia base de datos. Esta decisión implicó algunos sacrificios, entre ellos la reducción de la variedad de clases de alimentos y el número de imágenes disponibles para cada clase. A pesar de estas limitaciones, consideré que este enfoque era el más adecuado para centrarme en el reconocimiento de los productos más básicos y comunes en un frigorífico. En la sección 6 detallo las características específicas de esta nueva base de datos.

El segundo problema estaba relacionado con la capacidad del modelo para identificar varios alimentos en una sola imagen. Inicialmente, el modelo estaba configurado para identificar solo un alimento en la imagen, es decir, lo que es conocido como una clasificación de imágenes. Sin embargo, en realidad, un frigorífico suele contener varios alimentos diferentes. Por tanto, era necesario que el modelo pudiera identificar y localizar múltiples alimentos en una sola imagen. Este problema mostró que la metodología utilizada no era la más adecuada para realizar esta tarea y que había que cambiar dicha metodología. Así se pasó a la detección y clasificación de objetos en imágenes en lugar de la clasificación de imágenes en categorías.

Estos dos problemas mostraron que se necesitaba hacer algunos cambios importantes en el proyecto. Primero, como se dijo previamente, se debía que ajustar la base de datos que estaba utilizando para que representara de manera más precisa los tipos de alimentos que cualquier persona pudiese encontrar en un frigorífico. Y segundo, se tenía que cambiar la forma en que el modelo de inteligencia artificial estaba trabajando. En lugar de enseñar a identificar un solo alimento en una foto, se tenía que enseñar a reconocer y ubicar varios alimentos en la misma foto.

3.7 Guía resumen del desarrollo de la metodología inicial

- Preparación de la base de datos:
  - Se descargan las bases de datos de kaggel.
  - Luego, se fusionan ambas bases de datos.

  - Se divide la base de datos en 3 subcarpetas (70 % entrenamiento, 15 % validación y 15 % prueba).
  - Utilizar el script de python con nombre *resize.py* para que todas las imágenes de la base de datos tengan el mismo tamaño de 384x216 píxeles.
  - En dicho script se debe introducir el directorio raíz de la base de datos y el directorio de salida de las imágenes redimensionadas. La base de datos mantendrá la misma estructura de subdirectorios.

- Entrenamiento del modelo:
  - Ahora se utiliza el script en python con nombre *modeloPruebaReal2.py*. A este script hay que pasarle el directorio en el que se encuentran las imágenes de entrenamiento y las imágenes de validación.
  - También se deben indicar el número de épocas y el batch size. En caso de que el número de clases sea distinto, se deberá modificar el script indicando el número total de clases de la base de datos. En este caso es de 118 clases.
  - Si se trabaja con una base de datos con otro tamaño que no sea 384x216 píxeles, también se deberá indicar.
  - Una vez ejecutado por completo el script, se habrá generado el modelo que tendrá como nombre *my_model.h5*.

- Creación del diccionario de clases:
  - Como al realizarse una predicción de una clase se realiza mediante índices y no mediante etiquetas. Debe haber un diccionario que traduzca esos índices en etiquetas. Para ellos se utiliza el script en pyhton *diccionario.py*, al que hay que pasarle como entrada el directorio de entrenamiento para que pueda mapear los índices a etiquetas.
  - Si se trabaja con una base de datos con otro tamaño que no sea 384x216 píxeles, también se deberá indicar.
  - Una vez ejecutado el script, se habrá generado el diccionario de clases que tendrá como nombre *class_labels.json*.

- Realización de predicciones:

- Con el script en python llamado *fase_test1.py* se puede predecir la clase a la que pertenece una imagen.
- Para ello se pasan como entradas el diccionario de clases y el modelo generados anteriormente. Además se pasará como entrada la ruta de la imagen de la que se quiere hacer la predicción.
- La imagen deberá tener un tamaño de 382x216 píxeles.

CAPÍTULO 4

REVISIÓN DE LA METODOLOGÍA Y ENFOQUE FINAL

4.1 Elección de YOLO como la nueva metodología

Tras los problemas iniciales encontrados con la clasificación de imágenes en categorías, se decidió cambiar a la detección y clasifiación de objetos en imágenes. Entre las opciones disponibles, se optó por YOLO (You Only Look Once), una metodología de detección de objetos en tiempo real popularmente conocida por su rendimiento eficaz y preciso.

YOLO, para llevar a cabo la detección, primero divide la imagen en una cuadrícula de SxS (figura 4.1 (a)). En cada una de las celdas predice N posibles "bounding boxes" y calcula el nivel de probabilidad de cada una de ellas (figura 4.1 (b)), es decir, se calculan SxSxN diferentes cajas, la gran mayoría de ellas con un nivel de certidumbre

muy bajo. Después de obtener estas predicciones se procede a eliminar las cajas que estén por debajo de un límite. A las cajas restantes se les aplica un paso de "non-max suppression" que sirve para eliminar posibles objetos que fueron detectados por duplicado y así dejar únicamente el más exacto de ellos (figura 4.1 (c)) [10].

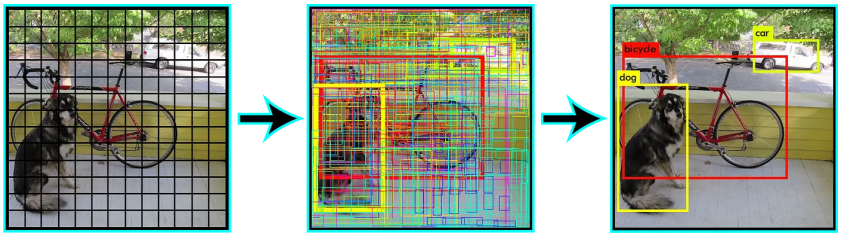

Figura 4.1: Detección YOLO

Ya que YOLO es bastante popular, se buscó en Internet cómo utilizaba la comunidad esta herramienta de una manera eficiente. Esto me proporcionó un rápido conocimiento sobre YOLO. Además, los desarrolladores de YOLO tienen un GitHub en el que dan recomendaciones e indicaciones para el entrenamiento [13].

Por último, pero no menos importante, la compatibilidad de YOLO con Google Colab también influyó en su elección. Google Colab es una plataforma de investigación en la nube que ofrece un entorno de ejecución con GPU, lo que es especialmente ventajoso para entrenar modelos de aprendizaje profundo como YOLO. Google Colab facilita la implementación y la experimentación al permitir que el código Python se ejecute en tiempo real en el navegador. Además, existe una plantilla para YOLO en Google Colab, que evita configuraciones extensas y permite un comienzo rápido. Dicha plantilla está subida en el GitHub de los desarrolladores de YOLO, también se ha subido dicha plantilla junto con las modificaciones para el entrenamiento al GitHub del proyecto[11].

En cuanto a la elección de la versión de YOLO, se optó por YOLOv5. Aunque han salido versiones posteriores hasta la YOLOv8, se seleccionó YOLOv5 por su equilibrio entre rendimiento y eficiencia. Las versiones más recientes, aunque pueden ofrecer mejoras en rendimiento, también era más complejas y demandantes en términos de recursos computacionales. Además, la principal diferencia no era tanto la precisión de la detección, sino que en la rapidez de la detección. Esto no era algo que fuese prioritario, ya que la detección no iba a ser en tiempo real.

Cada una de las versiones de YOLO dispone de ciertas subversiones. Por ejemplo, para YOLOv5 se dispone de las subversiones YOLOv5n (nano), YOLOv5s (small),

YOLOv5m (medium), YOLOv5l (large), YOLOv5x (extra large). Las diferencias son que según aumenta el "tamaño", la detección es más precisa pero más lenta. Como se ha comentado en varias ocasiones, la rapidez no es algo que importe demasiado debido a que la detección no debe ser en tiempo real, se escogió YOLOv5x.

4.2 Creación de la nueva base de datos y proceso de etiquetado en makesense.ai

Dada la dificultad de encontrar una base de datos ya preparada que se ajustara a las necesidades de este proyecto, se decidió crear una base de datos personalizada. El enfoque fue centrarse en una cantidad limitada de clases, las cuales representaran los productos más comunes que podemos encontrar en un frigorífico. como botellas, bricks, tarros, huevos, etc.

Antes de comenzar a hacer fotos a productos y al frigorífico, se organizó la manera de trabajar. Se planificó de una manera rápida los tipos de fotos que se iban a hacer de los distintos productos tal como se pueden observar en las figuras 4.2 y 4.3. Estos productos pueden llegar a ser bastante distintos dentro de una misma clase, por ejemplo, una botella de cava es muy diferente a una botella de agua. En concreto, se realizó la siguiente planificación básica:

Para la construcción de esta base de datos, se crearon unas 650 imágenes repartidas en 14 clases, todas a mano. 90 % de las imágenes se utilizaron para entrenamiento (584) y las 10 % restantes para validación (78). La cantidad de imágenes era algo limitada, aunque fue una tarea difícil realizar igualmente esa cantidad de fotos.

Los desarrolladores de YOLO dan ciertas indicaciones o consejos para un mejor resultado en el entrenamiento. Una de esas indicaciones es la cantidad de imágenes que debe tener la base de datos. Recomiendan una cantidad mayor o igual a 1500 imágenes por clase [14]. Esto era algo inviable, por lo que se sabía que el resultado no iba a ser el mejor. Igualmente, se comentarán los resultados obtenidos en el entrenamiento posteriormente.

Después de crear la base de datos, el siguiente paso fue etiquetar las imágenes. Esta fue la parte más difícil del proyecto, ya que cada imagen tenía que marcarse a mano y

```
BASE DE DATOS:

1.Botellas de agua
        -Botellas de plástico transparente
        -Botellas de plástico de otro color
        -Botella de cristal transparente
        -Botella de cristal de otro color
        -Botellas tumbadas!!!
2.Bricks
        -Bricks estilo caja de zapatos
        -Bricks estilo cuadrado (p.e leche carrefour)
        -Bricks pequeños (p.e los de tomate frito)
        -Bricks que arriba tienen el tapón en medio
        -Bricks tumbados!!!
3.Huevos (cartón)
        -Cartón de muchos huevos
        -Cartón tamaño estándar
        -Cartón más alargado
        -Cartón de pequeño (6 huevos)
4.Latas
        -Latas normales (las míticas)
        -Latas finas (las típicas de red bull)
        -Latas grandes (las típicas de monster)
        -Latas tumbadas!!!???
5.Bandejas de carne y pescado
        -Bandejas de diferentes tamaños y de diferentes productos
6.Salsas
        -Salsas de diferentes tamaños y colores
        -Salsas hacia arriba y hacia abajo
7.Bolsas (queso, verduras en bolsa para pesar, etc)
8.Tuppers
        -Tuppers de diferentes tamaños, colores formas.
        -Tuppers con diferentes comidas dentro
```

Figura 4.2: Planificación de la base de datos parte 1

algunas imágenes tenían entre 15 y 20 objetos que debían identificarse y etiquetarse. Esto se debe a que muchas de las fotos mostraban un frigorífico lleno de productos.

Para etiquetar las imágenes, se utilizó makesense.ai, una herramienta online de código abierto. En makesense.ai, cada imagen se cargó y se trazó un cuadro delimitador alrededor de cada alimento presente. A cada cuadro delimitador se le asignó una etiqueta correspondiente a la clase del alimento.

Este proceso de etiquetado generó, para cada imagen, un fichero de texto .txt correspondiente con la misma estructura de nombre que la imagen original. Este fichero contiene una línea por cada objeto detectado en la imagen, y cada línea sigue el

```
9.Botellines
        -Botellines con el típico color de botellin de cerveza
        -Botellines transparentes
        -Botellines tumbados!!!???
10.Botes de vidrio
        -Mermelada de diferentes colores
        -Tarros de tomate
        -Otros tarros que se me ocurran
11.Mantequilla
        -Intentar pillar diferentes formas y tamaños de mantequilla
        -La forma y tamaño típico
12.Plato (sobras)
        -Fotos sin tapar con un plato hondo
        -Fotos tapando con un plato hondo
13.Naranja
14.Manzana
```

Figura 4.3: Planificación de la base de datos parte 2

formato específico para YOLO: <class><xcenter><ycenter><width><height>, donde <class>es un número entero que representa la clase del objeto, y <xcenter>, <ycenter>, <width>, y <height>son las coordenadas y dimensiones del cuadro delimitador en formato normalizado. Por ejemplo, una línea podría verse así: 13 0.500740 0.311231 0.114213 0.085660.

4.3 Ajuste de parámetros y entrenamiento del modelo

Una vez que la base de datos estaba lista y etiquetada, el siguiente paso fue entrenar el modelo de detección de objetos. Para ello, se recurrió a la plantilla, que he comentado en secciones anteriores, de Google Colab para YOLOv5. Se contó con una GPU NVIDIA Tesla T4 que ofrece Google Colab. Está basada en la arquitectura Turing de NVIDIA. Es especialmente conocida por su eficiencia en tareas de machine learning y deep learning.

Debido al tamaño considerable de la base de datos, se optó por subir la base de datos comprimida a Google Drive y montar dicho Drive en Google Colab. Esto permitió acceder a los datos de forma eficiente, evitando tener que subir repetidamente la base de datos cada vez que se iniciaba un nuevo entrenamiento.

El proceso de entrenamiento se llevó a cabo en varios pasos. En primer lugar, se clonó el repositorio GitHub de YOLOv5 e instaló las dependencias requeridas. Luego, se montó Google Drive y se descomprimió la base de datos.

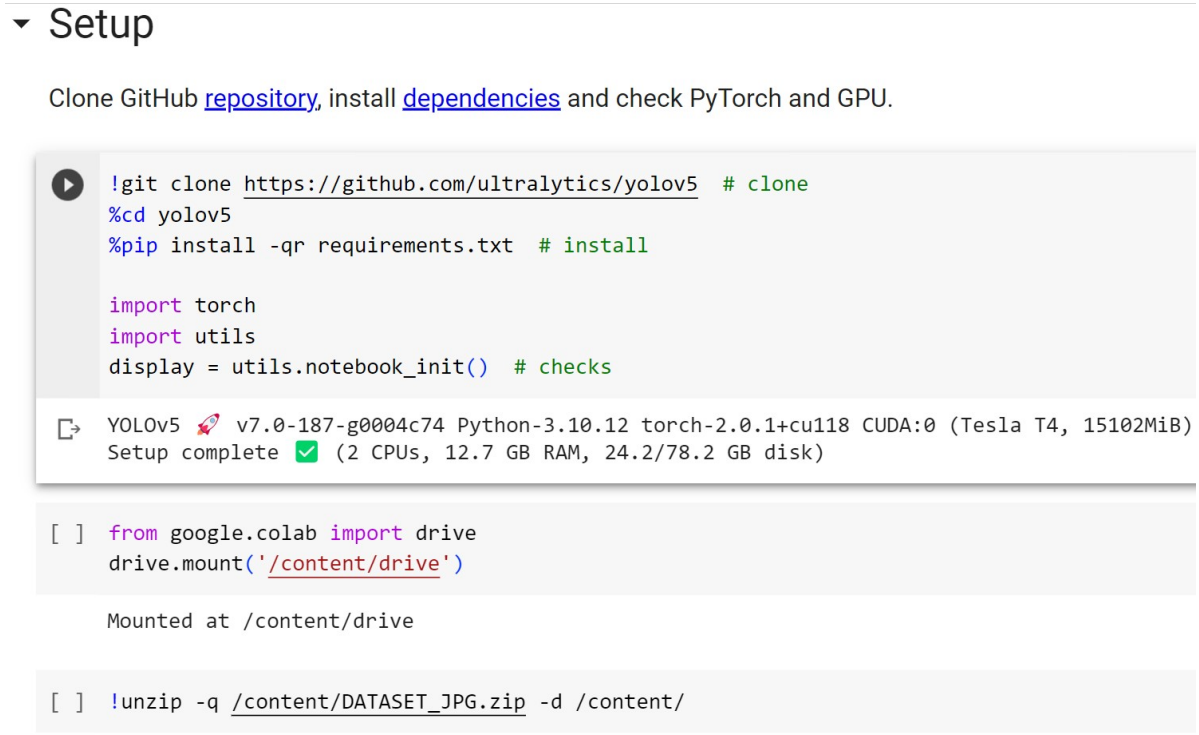

Figura 4.4: Plantilla en Google Colab (Setup)

Después de estos preparativos, se creó y subió a Google Colab un fichero YAML que contenía el diccionario de clases, es decir, una lista de todas las clases de alimentos que el modelo debía reconocer. Es importante mencionar que el orden de los nombres de clases de esta lista del fichero YAML debe ser el mismo orden de los índices de clases del etiquetado. Es decir, en la sección anterior se utilizó el ejemplo de la línea 13 0.500740 0.311231 0.114213 0.085660 en el que el número 13 correspondía a la clase. El número 13 corresponde concretamente a una manzana. Por tanto, en el índice 13 del fichero YAML (figura 4.5) debe estar escrita la palabra Manzana.

```yaml
train: /content/DATASET_JPG/train
val: /content/DATASET_JPG/val

nc: 14
names: ['Botella', 'Brick', 'Huevos', 'Lata', 'Bandeja', 'Salsa', 'Bolsa',
'Tupper', 'Botellin', 'Tarro', 'Mantequilla', 'Plato de sobras', 'Naranja', 'Manzana']
```

Figura 4.5: Fichero .yaml

Finalmente, se inició el entrenamiento del modelo. El comando utilizado para este propósito fue:

!python train.py –img 640 –batch 16 –epochs 100 –data /content/yolov5/data/entreno1.yaml –weights yolov5x.pt –cache.

Este comando indica que el tamaño de las imágenes utilizadas para el entrenamiento es de 640 píxeles, que el tamaño del lote es de 16, que el número de épocas de entrenamiento es de 100, y que se está utilizando una arquitectura YOLOv5x previamente entrenada como punto de partida. También se especifica la ubicación del fichero YAML y se habilita el caché para acelerar el proceso de entrenamiento.

```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 100 --data /content/yolov5/data/entreno1.yaml --weights yolov5x.pt --cache

train: weights=yolov5x.pt, cfg=, data=/content/yolov5/data/entreno1.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=100,
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v7.0-187-g0004c74 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_b
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5x.pt to yolov5x.pt...
100% 166M/166M [00:02<00:00, 61.5MB/s]

Overriding model.yaml nc=80 with nc=14

                 from  n    params  module                                  arguments
  0                -1  1      8800  models.common.Conv                      [3, 80, 6, 2, 2]
  1                -1  1    115520  models.common.Conv                      [80, 160, 3, 2]
  2                -1  4    309120  models.common.C3                        [160, 160, 4]
  3                -1  1    461440  models.common.Conv                      [160, 320, 3, 2]
  4                -1  8   2259200  models.common.C3                        [320, 320, 8]
  5                -1  1   1844480  models.common.Conv                      [320, 640, 3, 2]
  6                -1 12  13125120  models.common.C3                        [640, 640, 12]
  7                -1  1   7375360  models.common.Conv                      [640, 1280, 3, 2]
  8                -1  4  19676160  models.common.C3                        [1280, 1280, 4]
  9                -1  1   4099840  models.common.SPPF                      [1280, 1280, 5]
```

Figura 4.6: Plantilla en Google Colab (Train)

4.4 Implementación del modelo y detección de alimentos

Una vez entrenado el modelo con la base de datos personalizada se generan dos ficheros de pesos. El primero de ellos es best.pt, este fichero es el entrenamiento que ha demostrado mejores resultados. El segundo de ellos es last.pt, este fichero es el último entrenamiento y sirve para seguir entrenando el modelo a partir de ahí. Principalmente, interesa best.pt.

Se descargó el modelo entrenado y los ficheros necesarios para poder realizar la detección de imágenes localmente sin necesidad de realizarlo online con Google Colab.

Se aplica el modelo para la detección de objetos en las imágenes nuevas capturadas por la cámara del frigorífico. Este proceso se realiza a través del script *detect.py* que se encuentra en el repositorio de YOLOv5.

El comando utilizado para ejecutar este script desde la terminal es el siguiente:

python detect.py –weights best.pt –img 640 –conf 0.60 –source prueba.jpg

Donde:

• –weights best.pt indica la ubicación del modelo entrenado que se va a usar.

• –img 640 especifica el tamaño de la imagen que el modelo va a procesar.

• –conf 0.60 representa el umbral de confianza que el modelo debe superar para considerar que un objeto ha sido detectado. (Se puede modificar en función si el modelo está siendo muy restrictivo). Lo ideal sería 0.80 y, a partir de ahí bajarlo en función de las circunstancias. Por ejemplo, el entrenamiento ha sido realizado en un frigorífico en el que la luz es blanca. Para muchos frigoríficos en los que la luz en más cálida con tonos naranjas es recomendable que sea algo menor.

• –source prueba.jpg es la imagen o directorio de imágenes a las que se va a aplicar la detección de objetos.

Como resultado, este comando genera una nueva imagen con cuadros de detección de colores alrededor de los objetos que el modelo ha reconocido en la imagen original. Esta imagen con las detecciones se almacena en la carpeta 'runs/detect' del directorio del repositorio YOLOv5.

CAPÍTULO 5

Implementación en el sistema IoT

Tras tener el modelo basado en YOLOv5x ya entrenado, el siguiente paso es la implementación en el sistema IoT que ha desarrollado Ignacio González Domínguez. Como se dijo en anteriores ocasiones, el proyecto consta de dos partes. La primera de ellas es la de Ignacio González Domínguez y trata de la parte de IoT y la segunda es esta, que trata de la parte de Inteligencia Artificial.

Este sistema IoT se compone de varios componentes conectados: un sensor de apertura de puerta, una cámara y una Raspberry Pi.

Como se puede observar en la figura 5.1, cuando el usuario abre la puerta del frigorífico, el sensor de puerta detecta el cambio de estado y envía una señal a la Raspberry Pi. A su vez, la Raspberry Pi activa la cámara, que captura imágenes del interior del frigorífico. Estas imágenes se procesan y se almacenan en la Raspberry Pi, quedando preparadas para su posterior análisis.

En una etapa inicial del proyecto, se planteó la idea de subir la imagen a la nube directamente para su procesamiento. Sin embargo, para mantener los proyectos más independientes, se decidió cambiar esta idea. Ahora, en lugar de enviar la imagen a la nube, la imagen capturada se almacena localmente en la Raspberry Pi.

Cuando un usuario quiera comprobar el contenido del frigorífico, deberá obtener las imágenes almacenadas en la Raspberry Pi. Después, se puede aplicar el algoritmo de detección de objetos que desarrollado con YOLOv5x a estas imágenes para detectar y categorizar los alimentos.

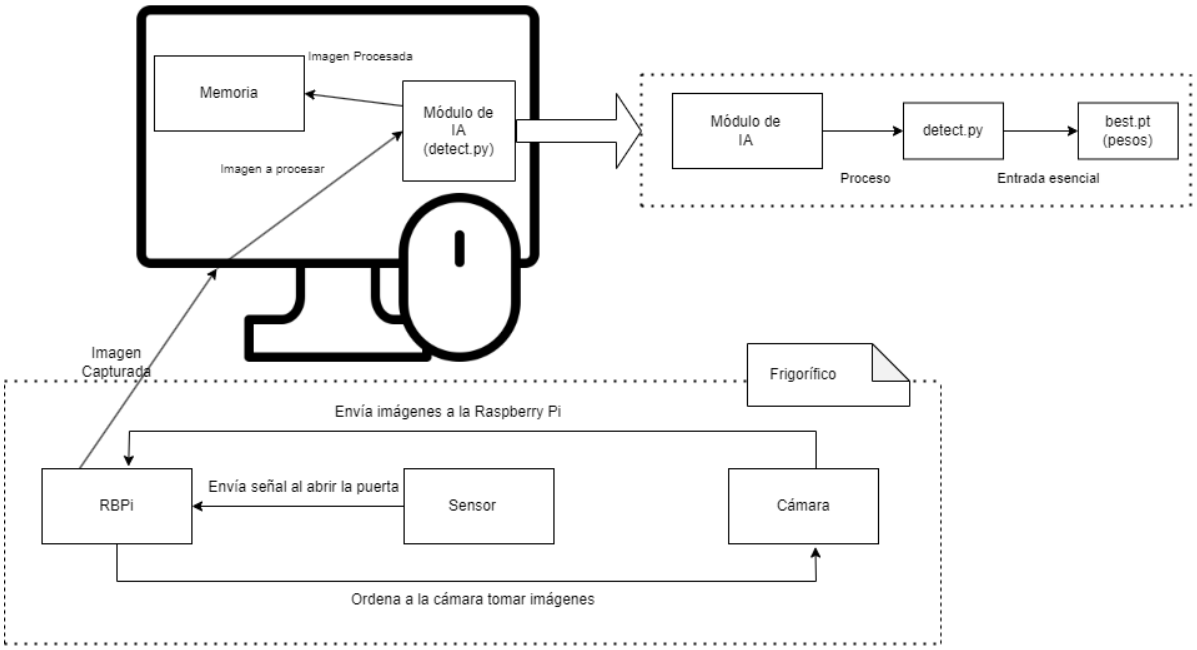

Figura 5.1: Diagrama del sistema completo

La implementación del algoritmo en este sistema IoT aportando una valiosa información sobre los hábitos alimentarios del usuario. Esta información la puede usar un experto en salud mental para poder ayudar a un usuario con ciertos problemas mentales.

CAPÍTULO 6

RESULTADOS Y DISCUSIÓN

6.1 Resultados obtenidos

A la hora de realizar la base de datos, hubo ciertos problemas como conté antes que hicieron que la detección de objetos no sea perfecta.

El primero de ellos fue la cantidad limitada y escasa de productos para poder hacer la base de datos. Esto se debe a que los productos eran propios y es complicado tener una base de datos amplia.

El segundo de estos problemas fue el frigorífico con el que se hicieron las fotos. Se hizo la mayoría de las fotografías en un único frigorífico. Esto hace que, al intentar detectar alimentos en frigoríficos con otras características y condiciones, no se reconozcan igual de bien.

Los resultados que se han obtenido los podemos analizar de dos maneras. La primera de ellas es observando y analizando los gráficos que se proporcionan al

acabar el entrenamiento, es decir, de una manera cuantitativa. Otra manera de analizar resultados es utilizando el modelo en diferentes imágenes de frigoríficos, es decir, de una manera cualitativa.

Por tanto, para la evaluación cualitativa se presentarán ejemplos de detecciones realizadas por el algoritmo en imágenes de frigoríficos reales. En la evaluación cuantitativa, se analizarán las métricas clave obtenidas durante el entrenamiento del modelo de IA.

6.1.1 Resultados cualitativos

En esta sección se muestran algunas imágenes de frigoríficos a las que se ha aplicado el algoritmo. En cada imagen, se mostrarán los cuadros de detección generados por el algoritmo, con etiquetas que indican el tipo de alimento identificado. Estas imágenes ilustrarán de la manera más clara y visual el rendimiento del algoritmo en la identificación de alimentos.

Se presenta una comparación del rendimiento del modelo en dos ambientes diferentes: mi frigorífico personal, donde se obtuvieron la mayoría de las imágenes para el entrenamiento, y en frigoríficos distintos, en el que el modelo no fue previamente entrenado con le objetivo de medir el grado de generalización del modelo desarrollado.

La detección de alimentos en mi frigorífico ha sido efectiva, como se esperaba, dado que el modelo se entrenó principalmente con imágenes tomadas en este ambiente y con la mayoría de los productos que se ven. En las figuras 6.1 y 6.2 se muestran distintas imágenes de mi frigorífico donde el algoritmo ha identificado correctamente los diferentes tipos de alimentos, demostrando su eficacia en el ambiente de entrenamiento.

Al aplicar el modelo en frigorífico y productos distintos, los resultados varían. En algunas ocasiones, el algoritmo ha identificado correctamente varios alimentos, indicando su habilidad para generalizar a partir del aprendizaje obtenido en el ambiente de entrenamiento. Sin embargo, en otros casos, el algoritmo ha confundido algunos alimentos o no los ha detectado. Recordemos que lo ocurrido es normal ya que no se han podido seguir la recomendación de 1500 imágenes que indican los desarrolladores de YOLO. Aquí se muestran ejemplos, para ilustrar los retos a los que se enfrenta el modelo cuando se aplica en ambientes diferentes al de entrenamiento:

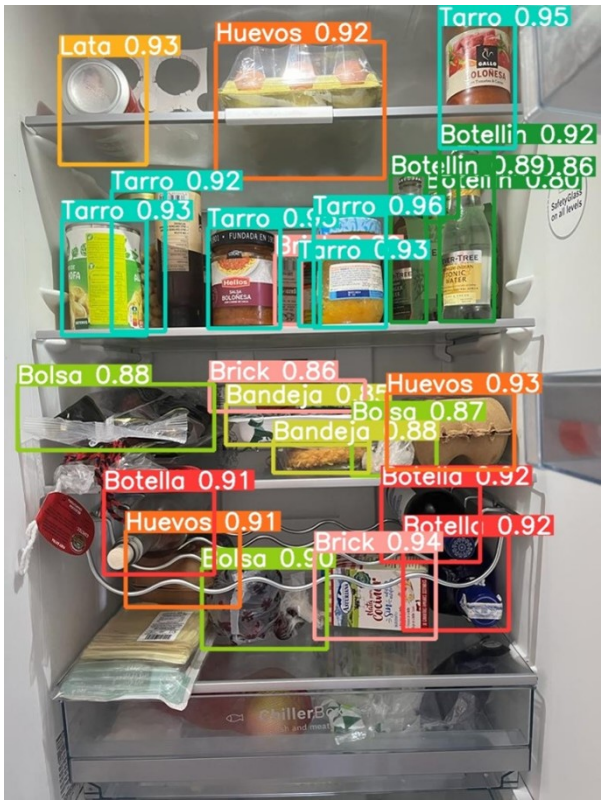

Figura 6.1: Detección en ambiente de entrenamiento

Si nos fijamos en la primera prueba (figura 6.3), podemos ver como el modelo funciona, aunque no totalmente. Si analizamos qué productos debería haber detectado y no lo ha hecho, deberíamos mencionar a las latas de la parte inferior izquierda, la salsa de la tabla medio superior derecha, la bandeja debajo del tupper detectado y el otro tupper que se encuentra al lado de la luz. Si analizamos los productos que se han detectado, pero de una manera errónea podría ser el tupper de arriba que lo detecta como una bandeja y el plato de sobras que detecta como bandeja.

Si nos fijamos en la segunda prueba (figura 6.4), podemos ver como el modelo funciona algo peor. Detecta más productos, pero también se detectan más productos de manera errónea. Hay varios tarros y salsas que detecta como latas, una salsa que detecta como tarro y unas latas que detecta como tarro.

Hay ciertas conclusiones que se han podido apreciar en base a ciertas pruebas realizadas que se comentarán en una sección futura orientada a conclusiones.

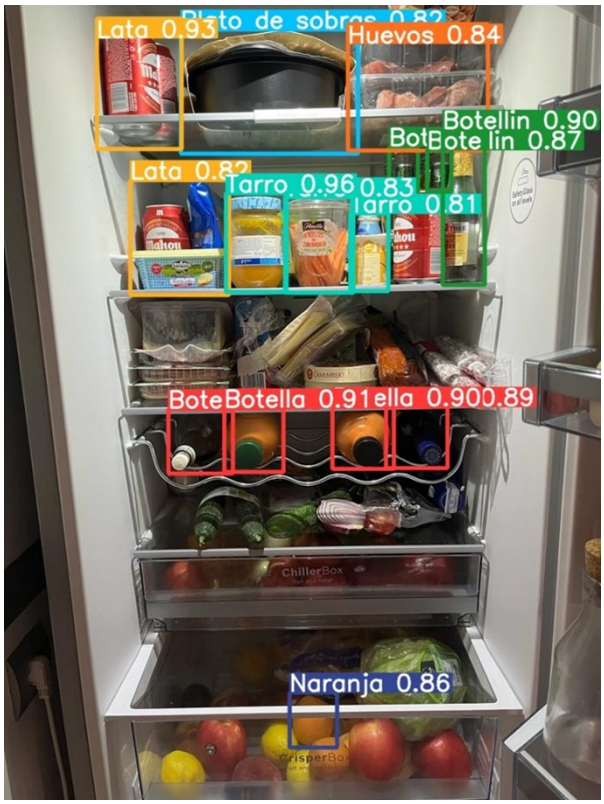

Figura 6.2: Detección en ambiente de entrenamiento 2

6.1.2 Resultados cuantitativos

Para proporcionar una visión cuantitativa del rendimiento del modelo, se generan una serie de gráficas durante el proceso de entrenamiento. Al ser muy limitada y poco variada la base de datos, estos resultados no reflejan con exactitud ciertas métricas. A continuación, se describen y presentan estas gráficas las correspondientes al entrenamiento:

1. Matriz de Confusión

La matriz de confusión es una herramienta muy útil para valorar cómo de bueno es un modelo clasificación basado en aprendizaje automático. En particular, sirve para mostrar de forma explícita cuándo una clase es confundida con otra, lo cual nos, permite trabajar de forma separada con distintos tipos de error [17].

En la figura 6.5 se indica que el entrenamiento ha conseguido que, por ejemplo, de cada 100 cajas de huevos, 99 las detecte correctamente. Otro ejemplo puede ser la bandeja, de cada 100 detecta 69 correctamente y 31 simplemente detecta que es el

Figura 6.3: Detección en un ambiente distinto

fondo. Esto es algo normal y una de las razones por las que ocurre eso es debido a que las bandejas suelen tener plásticos transparentes. Esto hace que, a priori, la detección de bandejas sea más compleja que la detección de huevos.

2. Curva F1

El valor F1 se utiliza para combinar las medidas de precisión y exhaustividad (recall) en un sólo valor. La precisión indica la proporción de verdaderos positivos frente a falsos positivos. El recall indica la proporción de verdaderos positivos frente a falsos negativos.

Figura 6.4: Detección en un ambiente distinto

Analizar el valor F1 es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad (recall) entre varias soluciones [12].

Se calcula como se muestra en la figura 6.7:

El valor F1 es una forma de resumir la efectividad de la clasificación de un modelo en un solo número. Si el valor de F1 es alto (cerca de 1), esto indica que el modelo tiene un buen rendimiento en términos de precisión y recall. Si es bajo (cerca de 0), esto indica que el modelo puede tener problemas para clasificar correctamente los datos.

Por tanto, la curva F1 muestra la evolución de este valor F1 durante el entrenamiento.

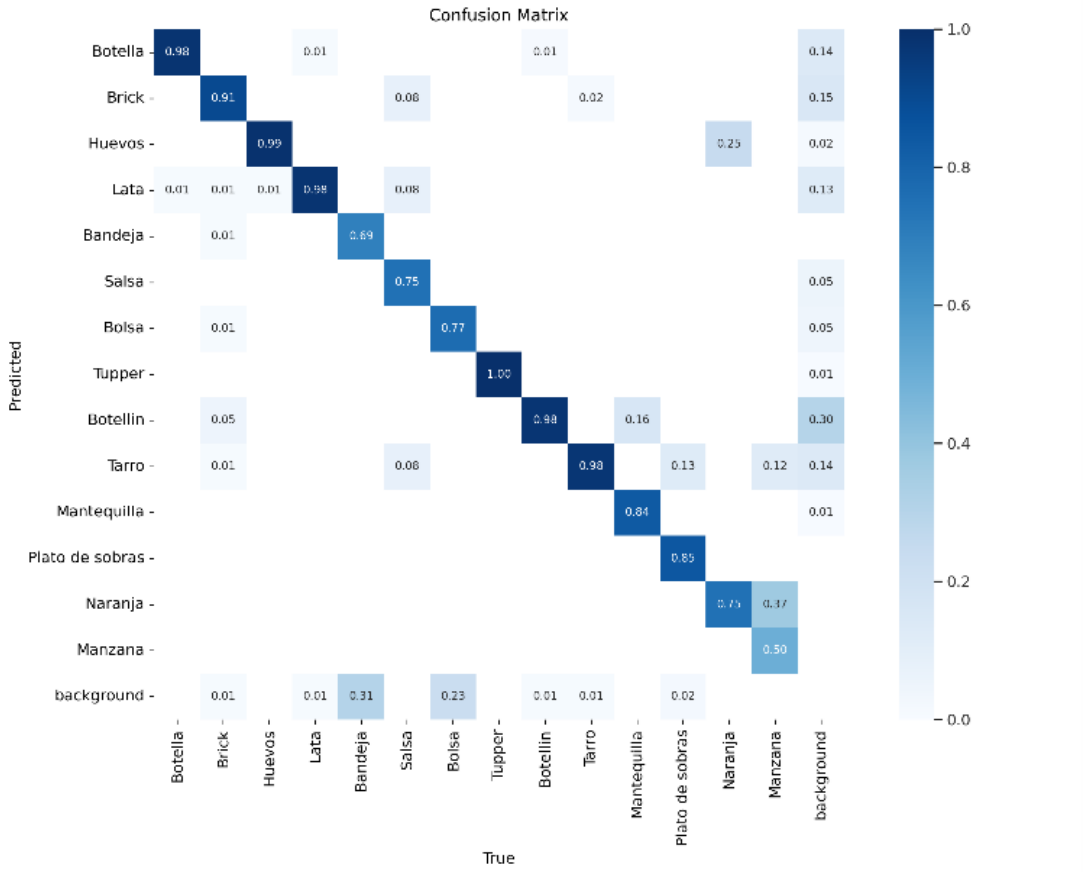

Figura 6.5: Matriz de Confusión

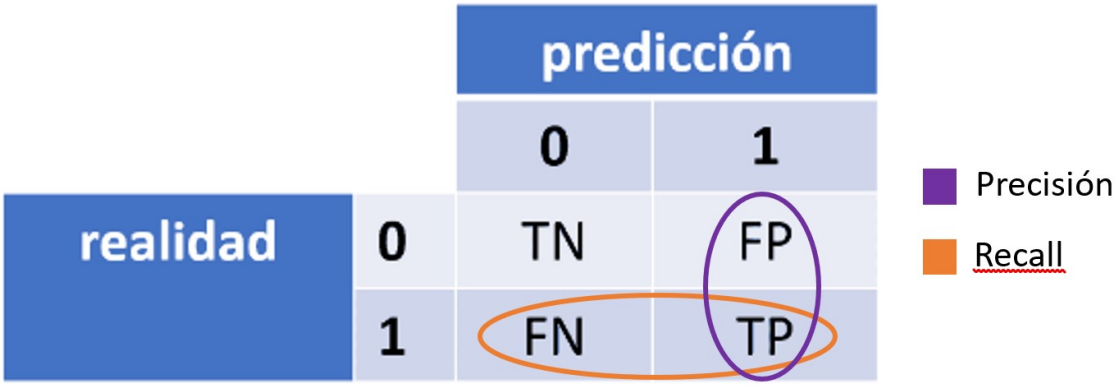

Figura 6.6: Matriz de confusión genérica

El valor "confidence" indica cómo de seguro está el modelo de que la detección es correcta. Un valor bajo está permitiendo al modelo que haga muchas detecciones, incluso las que no está muy seguro de ellas. Sin embargo, un valor muy alto hace que el modelo haga pocas detecciones, solamente de las que está muy seguro de ellas.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figura 6.7: Cálculo del valor F1

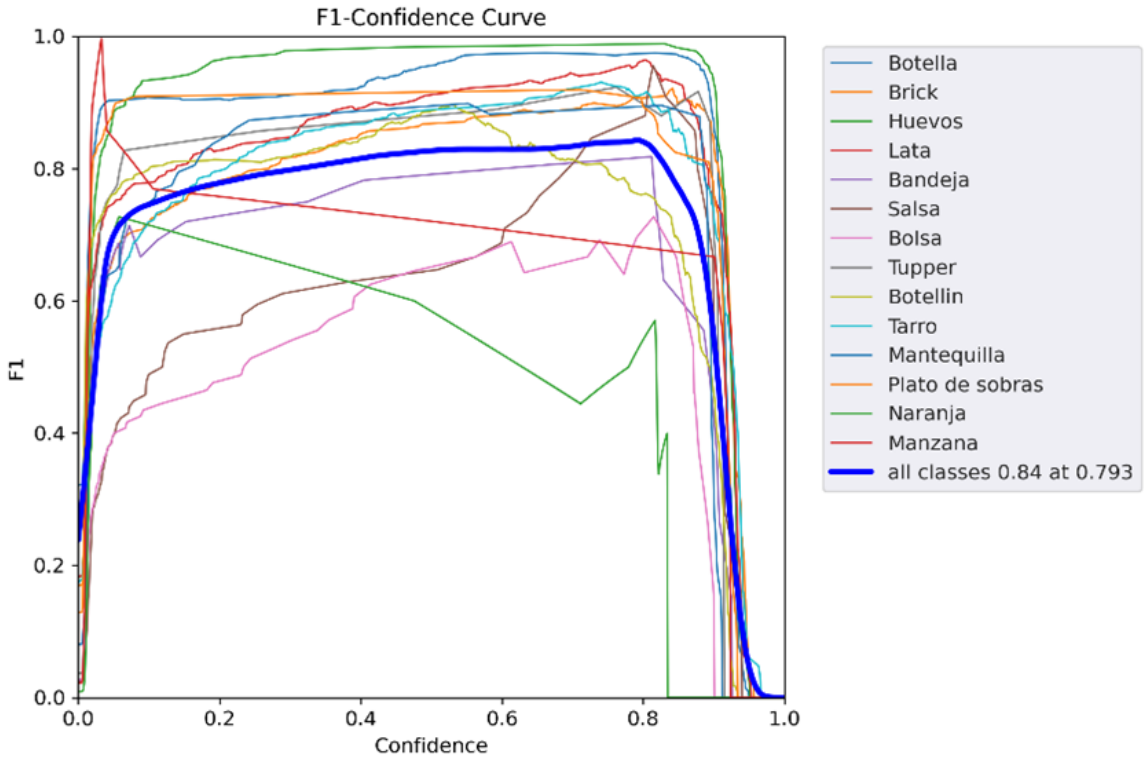

Figura 6.8: Curva F1-confianza

La gráfica la podemos analizar separando en 3 intervalos de confianza:

• Confianza entre 0 y 0.1: En este rango, el valor de F1 aumenta rápidamente hasta 0.7. Esto sugiere que el modelo es capaz de hacer predicciones bastante precisas incluso cuando su nivel de confianza es bajo. Es posible que en este rango de confianza se estén haciendo muchas predicciones correctas de clases minoritarias, las cuales, aunque el modelo no esté muy seguro de ellas, están resultando correctas y aportando al valor F1.

• Confianza entre 0.1 y 0.8: Aquí el valor de F1 aumenta lentamente de 0.7 a 0.8. Esto indica que el modelo se está volviendo más preciso a medida que aumenta la confianza en sus predicciones, aunque no de manera tanto como en el rango anterior.

• Confianza entre 0.8 y 1: En este rango de confianza, el valor de F1 cae drásticamente hasta 0. Esto es bastante interesante, y podría indicar que cuando el modelo está muy seguro de sus predicciones (confianza mayor que 0.8), tiende a equivocarse. Esto se puede deber al overfitting en el entrenamiento, donde el modelo se ha aprendido demasiado bien los datos de entrenamiento y por lo tanto realiza malas predicciones cuando se encuentra con datos diferentes. En otras palabras, el modelo ha memorizado en cierta parte los productos de la base de datos en lugar de aprender a notar características.

3. Curvas P y R

Las curvas P (Precision), R (Recall) muestran el rendimiento del modelo a lo largo del tiempo en términos de precisión, recall. Ya se ha explicado que representan la precisión y el recall a la hora de explicar el valor F1.

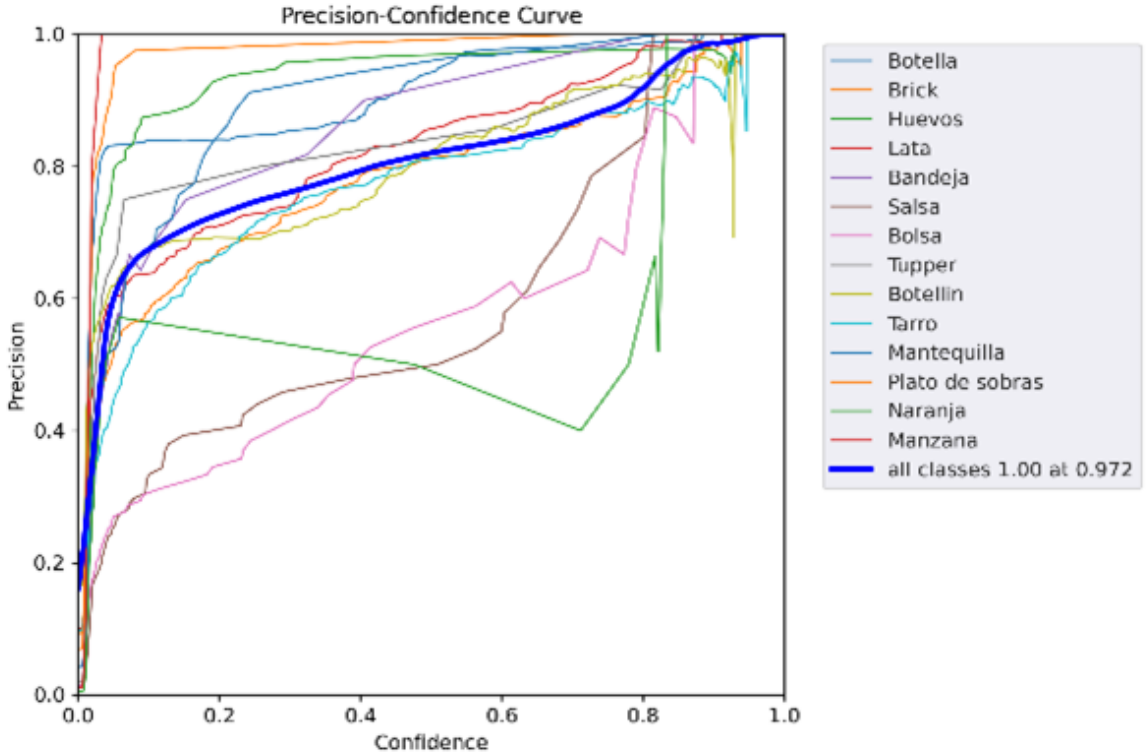

Figura 6.9: Cruva Precisión-confianza

De estas gráficas se pueden sacar las siguientes conclusiones:

• Curva de precisión:

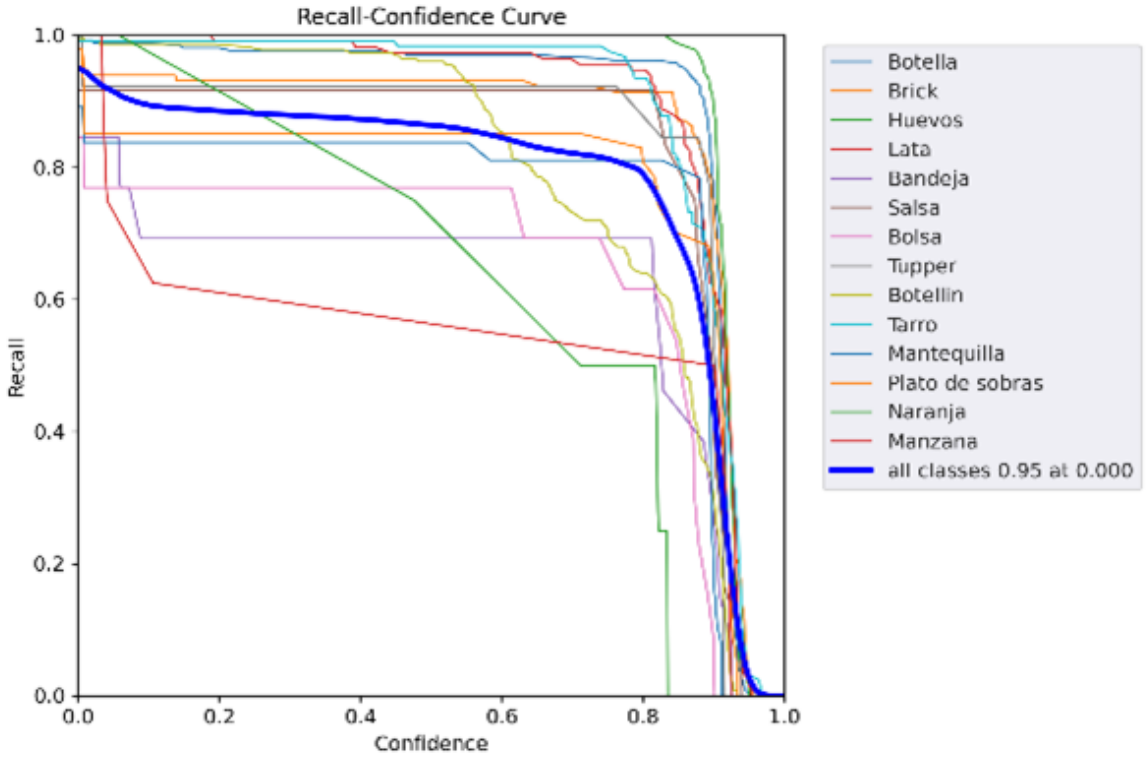

Figura 6.10: Curva Recall-confianza

o Confianza entre 0 y 0.1: En este rango, el valor de precisión aumenta rápidamente hasta alrededor de 0.65. Esto sugiere que incluso cuando el modelo no tiene mucha confianza en sus predicciones, sigue siendo capaz de hacer predicciones bastante precisas. Esto podría ser porque el modelo está haciendo un buen trabajo al identificar ciertas clases, quizás las que están más representadas en los datos.

o Confianza entre 0.1 y 1: Aquí el valor de precisión aumenta lentamente hasta llegar a 1. Este comportamiento sugiere que a medida que el modelo se vuelve más confiado en sus predicciones, su precisión mejora. Llegando a ser perfecta (precisión = 1) cuando su confianza es alta.

Este es un comportamiento típico y deseado en un buen modelo de clasificación: cuando el modelo tiene alta confianza, también debería ser muy preciso. En otras palabras, cuando el modelo dice que está seguro de su predicción, debería tener razón la mayoría de las veces.

- Curva de recall:

o Confianza entre 0 y 0.8: Durante este rango, el recall disminuye gradualmente hasta alrededor de 0.8. Esto significa que a medida que el modelo aumenta su confianza en sus predicciones, es capaz de identificar un porcentaje cada vez menor de las instancias positivas reales. En otras palabras, está dejando de detectar correctamente algunos de los verdaderos positivos.

o Confianza entre 0.8 y 1: Aquí el recall disminuye rápidamente hasta llegar a 0. Esto indica que cuando el modelo tiene mucha confianza en sus predicciones (confianza >0.8), falla en identificar la mayoría, si no todas, las instancias positivas. Es decir, su capacidad para detectar verdaderos positivos se reduce drásticamente.

Esto puede indicar que el modelo es conservador en las predicciones y tiende a predecir la clase negativa cuando no está absolutamente seguro, lo que puede resultar en un alto número de falsos negativos (casos en los que la clase real es positiva pero el modelo la predice como negativa) y, por lo tanto, un bajo recall.

4. Resultados

La gráfica de resultados muestra el progreso del entrenamiento, es decir, cómo mejoró el modelo a lo largo de las épocas de entrenamiento.

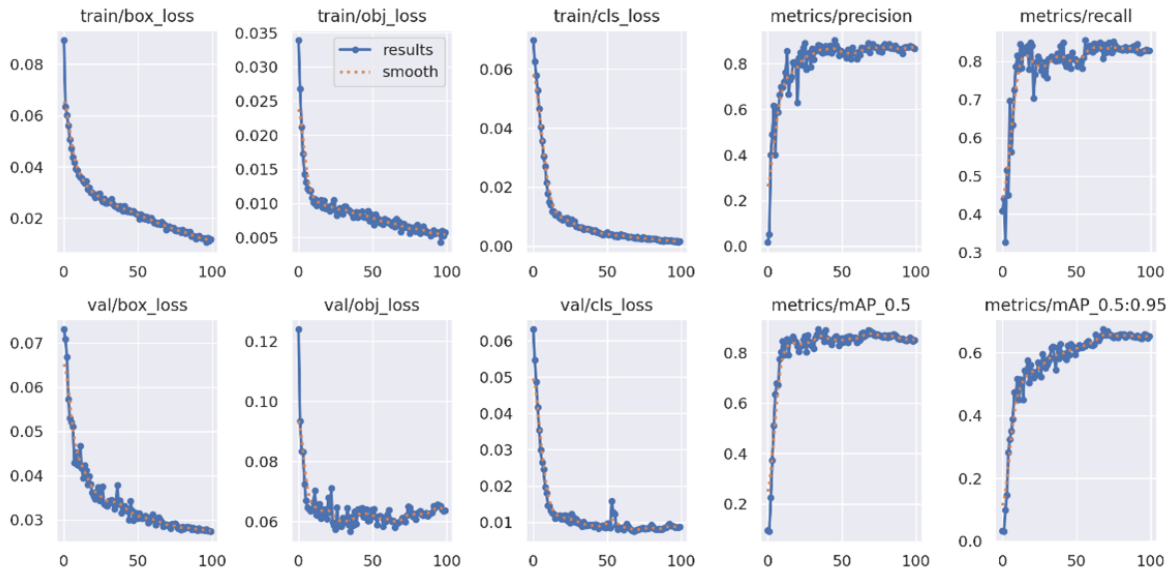

Figura 6.11: Evolución del modelo en el entrenamiento

Como se puede ver en las gráficas, se hicieron 100 épocas de entrenamiento. Se puede observar que, tras ciertas épocas de entrenamiento la precisión y el recall no mejoran significativamente. A menudo, cuando estas métricas dejan de mejorar es

buen momento para dejar de entrenar el modelo. Esto se debe a que ayuda a prevenir el overfitting, además de ahorrar tiempo de entrenamiento y recursos. Esta técnica se le conoce como early stopping. Es importante mencionar que hay que tener cuidado con esta técnica ya que dejar de entrenar demasiado temprano puede producir underfitting [2].

5. Train Batch

El entrenamiento se ha realizado con un tamaño de lote o, en inglés, batch size de 16. Esto implica que las imágenes se agruparán en lotes de 16, así es más eficiente computacionalmente. Un lote contiene 16 imágenes y sus respectivas etiquetas. Al finalizar el entrenamiento, se proporciona como ejemplos ciertos lotes.

6. Validation Batch

Para la validación primero se hace una predicción a partir de las imágenes de validación. También se agrupan por lotes, un ejemplo podría ser la siguiente imagen.

Una vez ya se ha hecho la predicción, se comprueba con las etiquetas y ubicaciones reales de los objetos que se encuentran en cada imagen. Para el ejemplo anterior, se compararía con la siguiente imagen.

6.2 Discusión de los resultados

El modelo YOLOv5x, empleado en este proyecto, fue entrenado durante 100 épocas. Los resultados que hemos obtenido indican que el modelo ha aprendido a ajustarse a los datos de entrenamiento. Sin embargo, se debe tener cuidado, ya que estos resultados pueden indicar de un cierto grado de overfitting o sobreajuste.

Las métricas de validación siguen un patrón similar, con las curvas de pérdida también tendiendo a cero. Esto sugiere que el modelo se comporta adecuadamente con los datos de validación. No obstante, los resultados de las curvas F1-confidence, precision-confidence y recall-confidence indican un comportamiento del modelo a diferentes niveles de confianza que también puede haber sido producido por un cierto overfitting.

[2]Underfitting se refiere a los casos en los que un modelo de datos no es capaz de capturar de forma precisa la relación entre las variables de entrada y salida, de modo que se genera un alto índice de errores en el conjunto de entrenamiento y en los datos no vistos [4]

Figura 6.12: Ejemplo de lote de imágenes de entrenamiento

En la aplicación del modelo a frigoríficos genéricos, se observó que un gran número de alimentos no se detectaron o se detectaron erróneamente, lo que indica una necesidad de mejorar el entrenamiento del modelo.

La visualización de las predicciones de los lotes de validación confirmó que el modelo es capaz de detectar y clasificar correctamente los objetos en las imágenes. Sin embargo, estas predicciones deben compararse con las etiquetas reales para evaluar correctamente el rendimiento del modelo.

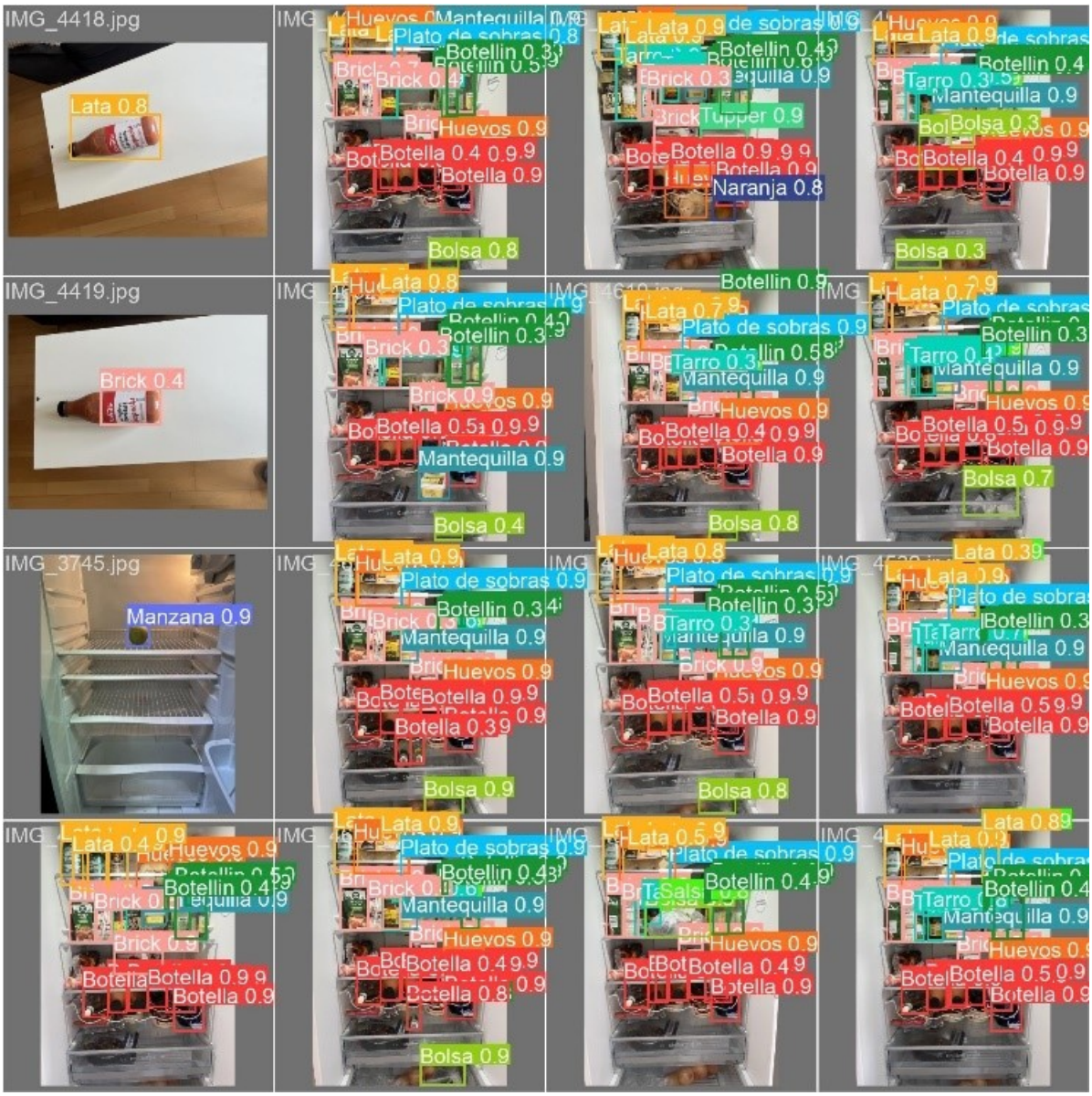

Figura 6.13: Ejemplo de predicción lote de imágenes de validación

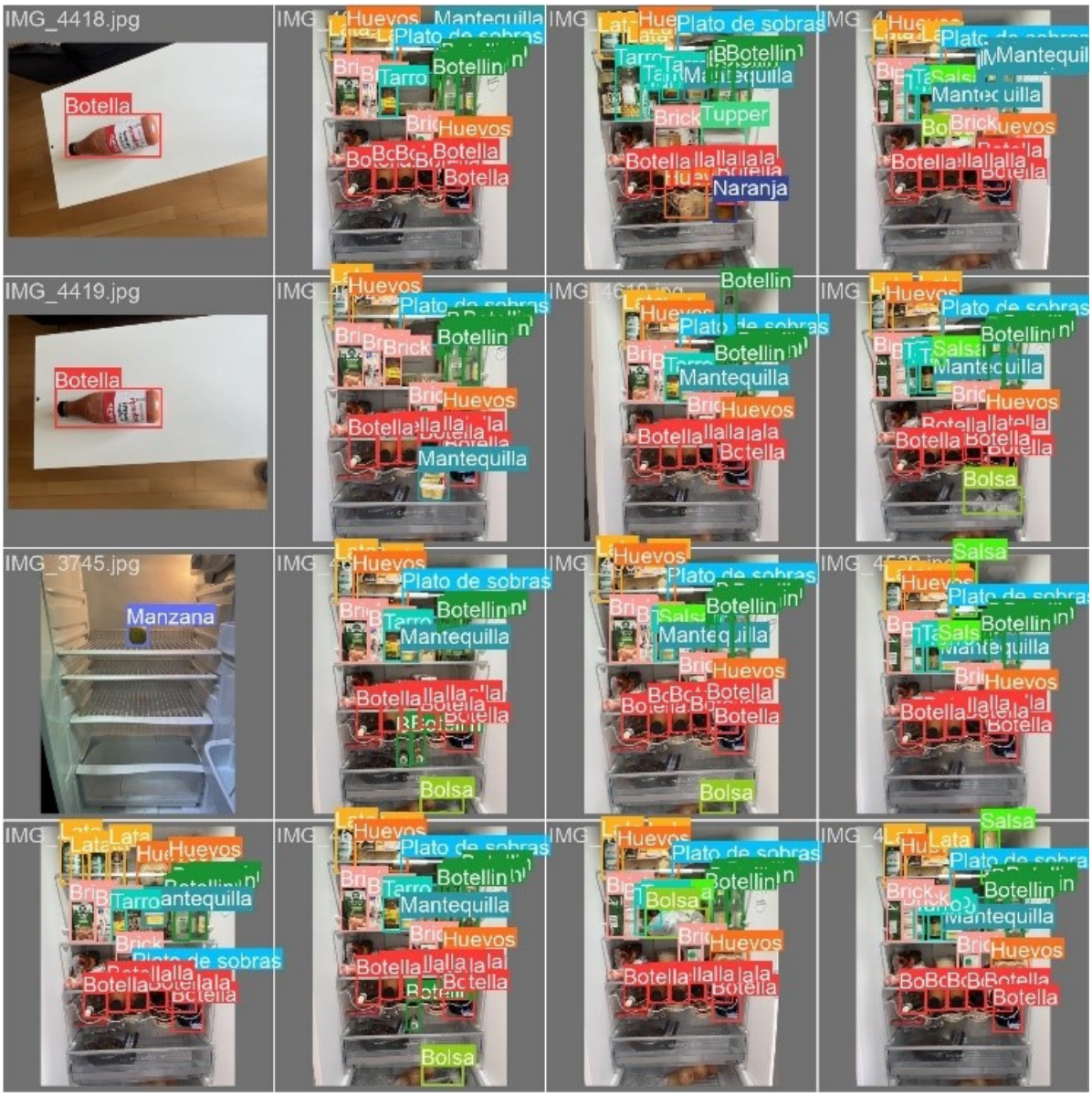

Figura 6.14: Ejemplo de lote de imágenes etiquetadas de validación

CAPÍTULO 7

CONCLUSIONES Y LÍNEAS FUTURAS

7.1 Conclusiones del proyecto

A lo largo de este proyecto se exploraron diversas metodologías. Inicialmente consideramos el uso de ResNet50, pero dio por concluido que YOLOv5 era más adecuado para la tarea específica de detección de alimentos en frigoríficos.

La construcción de una base de datos representativa fue otro desafío. Aunque se disponía de un gran volumen de datos, no reflejaban fielmente un frigorífico real. Esta experiencia recordó la importancia de la relevancia y calidad de los datos, además de su cantidad.

Posteriormente, se realizó un estudio sobre la aplicación de la detección de objetos mediante el uso de la YOLOv5 para el reconocimiento de alimentos en un frigorífico. A lo largo del trabajo, se ha obtenido una comprensión más profunda de cómo los modelos de detección de objetos aprenden y cómo pueden ser optimizados.

Los resultados obtenidos muestran que hay margen para mejorar. El rendimiento del modelo en frigoríficos genéricos ha revelado que una gran cantidad de alimentos

no se detectan o se detectan erróneamente. Esto indica que el modelo puede estar sobreajustándose a los datos de entrenamiento. Esto es bastante común en el aprendizaje automático, especialmente cuando se trata de conjuntos de datos limitados, como en este caso.

Este proyecto tiene el objetivo de utilizar la inteligencia artificial para ayudar a obtener una visión más clara del comportamiento alimentario de personas con problemas de salud mental. A pesar de los límites encontrados, el proyecto sienta las bases para futuros desarrollos que, con los ajustes necesarios, podrían ser utilizados como una buena herramienta en el campo de la salud mental.

Además, este trabajo subraya la importancia de tener un buen conjunto de datos de entrenamiento. Los desarrolladores de YOLO recomiendan 1500 imágenes por clase para un rendimiento óptimo. Nuestro conjunto de datos estaba muy por debajo de ese umbral, lo que probablemente haya contribuido a los problemas encontrados.

El proyecto, sin embargo, ha demostrado que la detección de alimentos en frigoríficos mediante IA es factible. Con mejoras en el conjunto de datos de entrenamiento y ajustes en los parámetros de entrenamiento, es probable que la precisión de la detección mejore. Este trabajo pone de manifiesto la relevancia y el potencial de la inteligencia artificial en nuestra vida cotidiana, incluso en aplicaciones como la asistencia de la salud mental.

7.2 Líneas futuras

El proyecto ofrece una base sólida para la detección de alimentos en frigoríficos mediante el uso de la inteligencia artificial y específicamente de la arquitectura YOLOv5. No obstante, existe un amplio margen para mejoras y expansiones futuras.

Una mejora obvia sería la recopilación de un conjunto de datos más amplio. El limitado número de imágenes usadas para el entrenamiento podría ser la causa de cierto grado de sobreajuste. Se podría ampliar el conjunto de datos, incluyendo imágenes de frigoríficos con diversas condiciones (iluminación, ángulos, etc). Además, para asegurar la diversidad y la robustez del conjunto de datos, sería beneficioso recoger imágenes de frigoríficos de personas con estilos de vida variados, dietas variadas o que compren en supermercados variados.

En relación con el punto anterior, es interesante que se busquen bases de datos con uso libre para proyectos de investigación. Si bien es algo tedioso, es más viable que realizar la captura y etiquetado de más de 20000 imágenes (según recomendación de YOLO).

También sería beneficioso explorar otros modelos en este problema. Aunque YOLOv5 ha demostrado un rendimiento decente, otras arquitecturas de YOLO o incluso otras arquitecturas en general podrían proporcionar resultados superiores.

Dada la posibilidad de que el modelo actual pueda estar sobreajustado a los datos de entrenamiento, se podría investigar como poder minimizar este problema con técnicas como el early stopping.

Sería interesante que el experto en salud mental que vaya a observar la evolución del contenido de su paciente disponga de una interfaz amigable. Se podría desarrollar un sitio web en el que haya una base de datos con el registro del contenido, así podría ser más sencillo para el usuario hacer uso de la herramienta.

A largo plazo, sería útil integrar el sistema de detección de alimentos en frigoríficos con un sistema de recomendación de comidas o una aplicación de seguimiento de la dieta. Esto no solo identificaría los alimentos presentes en el frigorífico, sino que también sugeriría recetas basadas en esos alimentos o proporcionaría información nutricional. Obviamente esto es algo secundario.

ÍNDICE DE FIGURAS

BIBLIOGRAFÍA

[1] *¿qué es el aprendizaje automático?* https://cloud.google.com/learn/what-is-machine-learning?hl=es-419, Accedido el 14 de julio de 2023.

[2] *¿qué es el aprendizaje no supervisado?* https://www.ibm.com/es-es/topics/unsupervised-learning, Accedido el 12 de agosto de 2023.

[3] *¿qué es el aprendizaje supervisado?* https://www.ibm.com/es-es/topics/supervised-learning, Accedido el 12 de agosto de 2023.

[4] *¿qué es el subajuste?* https://www.ibm.com/es-es/topics/underfitting, Accedido el 18 de julio de 2023.

[5] *Fruits 360*, 2017. https://www.kaggle.com/datasets/moltean/fruits, Accedido el 20 de mayo de 2023.

[6] *Food 101*, 2018. https://www.kaggle.com/datasets/dansbecker/food-101, Accedido el 16 de mayo de 2023.

[7] *Redes neuronales residuales – lo que necesitas saber (resnet)*, 2020. https://datascience.eu/es/aprendizaje-automatico/una-vision-general-de-resnet-y-sus-variantes/, Accedido el 12 de agosto de 2023.

[8] *¿qué es la inteligencia artificial y cómo se usa?*, 2021. https://www.europarl.europa.eu/news/es/headlines/society/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa, Accedido el 14 de julio de 2023.

[9] *¿qué es el internet de las cosas (iot)?*, 2023. https://www.redhat.com/es/topics/internet-of-things/what-is-iot, Accedido el 12 de julio de 2023.

[10] Enrique A.: *Detección de objetos con yolo: implementaciones y como usarlas*, 2018. https://medium.com/@enriqueav/detecci%C3%B3n-de-objetos-con-yolo-implementaciones-y-como-usarlas-c73ca2489246, Accedido el 10 de julio de 2023.

[11] Mario Laustalet Fernández: *Tfg*, 15 de agosto de 2023. https://github.com/laustalet/TFG.

[12] Jose Martinez Heras: *Precision, recall, f1, accuracy en clasificación*, 2020. https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/, Accedido el 11 de julio de 2023.

[13] Glenn Jocher: *YOLOv5 by Ultralytics*, Mayo 2020. https://github.com/ultralytics/yolov5.

[14] Glenn Jocher: *Tips for best training results*, 2023. https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/#training-settings, Accedido el 8 de julio de 2023.

[15] Luz Perez: *Descubre la ia que reconoce imágenes*, 2023. https://neuroflash.com/es/blog/descubre-la-ia-que-reconoce-imagenes/#:~:text=El%20reconocimiento%20de%20im%C3%A1genes%20en,que%20los%20humanos%20lo%20hacen, Accedido el 14 de julio de 2023.

[16] Alexandra Sanfins: *Nutrition and mental health: Is there a link?*, 2021. https://www.medicalnewstoday.com/articles/nutrition-and-mental-health-is-there-a-link, Accedido el 25 de mayo de 2023.

[17] Paloma Recuero de los Santos: *Cómo interpretar la matriz de confusión: ejemplo práctico*, 2021. https://telefonicatech.com/blog/como-interpretar-la-matriz-de-confusion-ejemplo-practico#:~:text=La%20matriz%20de%20confusi%C3%B3n%20es,con%20distintos%20tipos%20de%20error, Accedido el 10 de julio de 2023.