
Aplicación móvil de identificación de objetos para reciclaje



TRABAJO DE FIN DE GRADO

Celia Castaños Bornaechea

Grado en Desarrollo de Videojuegos

Facultad de Informática

Universidad Complutense de Madrid

Junio 2021

Documento maquetado con TEXIS v.1.0.

Este documento está preparado para ser imprimido a doble cara.

Aplicación móvil de identificación de objetos para reciclaje

Trabajo de Fin de Grado

Dirigido por: Pedro Pablo Gómez Martín

Grado en Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

Junio 2021

Copyright © Celia Castaños Bornaechea

Resumen

Con el incremento de los envases de un solo uso y el aumento de los tipos de materiales, se ha observado un crecimiento masivo en los residuos, y estos, al no ser gestionados adecuadamente, han provocado un gran impacto medioambiental. Para hacerle frente a este problema una acción sencilla pero efectiva es separar los residuos en el origen. No obstante, con la diversidad de materiales existentes actualmente puede resultar complicado saber cómo desecharlos todos correctamente. Este Trabajo de Fin de Grado busca ofrecer, mediante una aplicación móvil, una pequeña ayuda para solventar estas dudas a personas que lo puedan necesitar.

Para ello, se ha querido aprovechar el gran avance que se ha vivido en visión computacional durante los últimos años y desarrollar una aplicación de identificación de objetos. Para esto se ha utilizado TensorFlow Lite, cuyas librerías facilitan tanto el entrenamiento de la red neuronal necesaria y la generación del modelo, como su importación en aplicaciones móviles. Asimismo, para facilitar la tarea de obtención de datos para el entrenamiento, se ha desarrollado una aplicación de generación de imágenes sintéticas, las cuales se generan a partir de modelos tridimensionales y son utilizadas posteriormente para el entrenamiento de la red.

Todo esto en conjunto permite obtener como resultado una aplicación móvil de identificación de materiales que ha sido entrenada utilizando imágenes generadas sintéticamente.

Palabras clave: reciclaje, identificación de objetos, red neuronal, TensorFlow Lite, generación de imágenes, visión computacional, *dataset*, aplicación móvil.

Todos los materiales pueden encontrarse en el repositorio de GitHub
<https://github.com/celica02/RecyclingFinalDegreeProject>

Abstract

With the increase of single-use packaging and the different types of materials, there has been a massive growth in waste. Inappropriate management of these wastes have caused a huge environmental impact. To deal with this problem, one simple but effective action is to separate the waste at the source point. However, with the diversity of materials available today it can be difficult to know how to correctly dispose all of them. This final degree project seeks to offer help to solve the doubts through a mobile application.

To do so, we wanted to take advantage of the great progress that has been made in computer vision in recent years and develop an application that identifies objects. To achieve this, TensorFlow Lite has been used, whose libraries provides an easy way for training the needed neural network and generate its model, as well as its import into mobile applications. Likewise, to ease the task of obtaining the necessary datasets for training, an application that generates synthetic images from three-dimensional models has been developed. These images are subsequently used for the neuronal network training.

As a result we obtain a mobile application which identifies different materials trained using synthetic images.

Key words: recycling, object identification, neuronal network, TensorFlow Lite, image generation, computer vision, dataset, mobile application.

All materials can be found in the GitHub repository <https://github.com/celica02/RecyclingFinalDegreeProject>

Índice

| | |
|---|------------|
| Resumen | v |
| Abstract | vii |
| 1. Introducción | 1 |
| 1.1. Motivación | 1 |
| 1.2. Objetivos | 2 |
| 1.3. Herramientas utilizadas | 3 |
| 1.4. Plan de trabajo | 3 |
| 2. Estado del arte | 9 |
| 2.1. Aplicaciones de reciclaje | 9 |
| 2.2. Identificación de objetos e imágenes | 12 |
| 2.3. Redes neuronales | 14 |
| 2.4. TensorFlow y TensorFlow Lite | 17 |
| 2.5. Generación sintética de imágenes | 18 |
| 2.6. Unity | 20 |
| 3. Aplicación de generación de imágenes | 23 |
| 3.1. Introducción | 23 |
| 3.2. Aplicación | 25 |
| 3.2.1. Modelos 3D | 27 |
| 3.2.2. <i>Dataset</i> | 29 |
| 4. Entrenamiento y selección de <i>dataset</i> | 31 |
| 4.1. Experimento | 31 |
| 4.2. Entrenamiento | 33 |
| 4.3. Resultados | 34 |
| 5. Aplicación de identificación de objetos | 39 |
| 5.1. Introducción | 39 |
| 5.2. Identificación de objetos | 40 |

| | |
|---|-----------|
| X | ÍNDICE |
| 5.3. Pruebas | 41 |
| 6. Conclusiones y trabajo futuro | 47 |
| 6.1. Recapitulación | 47 |
| 6.2. Conclusiones | 48 |
| 6.3. Trabajo futuro | 49 |
| A. Materiales | 57 |
| A.1. Materiales adjuntos | 57 |
| Bibliografía | 59 |
| Índice alfabético | 65 |

Índice de figuras

| | |
|--|----|
| 2.1. Aplicaciones de reciclaje | 10 |
| 2.2. Estructura de una red neuronal artificial. | 15 |
| 3.1. Capturas de ejemplo de los fondos | 26 |
| 3.2. Imágenes de ejemplo de un modelo con transparencia online y en el editor | 28 |
| 4.1. Organización de las carpetas con las imágenes separadas. | 34 |
| 4.2. Variación de la exactitud de la red neuronal a lo largo del entrenamiento | 35 |
| 4.3. Variación de la exactitud de la red neuronal durante el <i>test</i> | 36 |
| 4.4. Variación de la exactitud final de la red neuronal según la proporción de imágenes reales y sintéticas | 37 |
| 5.1. Interfaz de la aplicación de identificación | 41 |
| 5.2. Variación de la precisión de la red neuronal con los modelos seleccionados destacados | 42 |
| 5.3. Comparación con distintos objetos metálicos | 43 |
| 5.4. Comparación con distintos objetos de vidrio | 44 |
| 5.5. Comparación con distintos objetos de plástico | 44 |
| 5.6. Comparación de la misma botella de plástico con segundos de diferencia | 45 |

Índice de Tablas

| | |
|---|----|
| 3.1. Imágenes disponibles por material | 29 |
| 4.1. Imágenes disponibles por material separadas en entrenamiento y <i>test</i> | 33 |

Capítulo 1

Introducción

1.1. Motivación

La generación de residuos está ligada al modelo actual de desarrollo de la sociedad y constituye uno de los principales problemas ambientales a los que se enfrenta el mundo [47]. Los residuos se pueden clasificar en dos tipos: los producidos por la actividad industrial, llamados residuos industriales, y los generados por la propia actividad humana, denominados residuos urbanos.

Bien es cierto que gran parte de la contaminación y emisiones de CO₂ provienen de grandes empresas, por ejemplo en 2018 el 25 % de las emisiones en España fueron generadas por solamente diez compañías [2]; pero este TFG se centrará en los residuos urbanos, que son en aquellos sobre los que la población puede concienciarse y tomar medidas directas al respecto.

El problema de los residuos sólidos urbanos viene del incremento en los últimos años de utilización de envases sin retorno. Estos embalajes pueden ser de diferentes materiales como celulosa, vidrio, plástico o mixtos (papel plastificado, telas plastificadas, etc.) lo que complica su tratamiento, puesto que se debe llevar a cabo una selección y separación previa [12].

Una mala gestión de los residuos puede provocar impactos medioambientales irreversibles. A día de hoy ya se pueden observar muchos de estos efectos que parecía que vendrían en el futuro [33]. Podemos destacar entre ellos el incremento de las temperaturas en todo el globo, la desaparición de los glaciares, tanto los de las montañas como los de los casquetes polares [7], o la disminución en un 68 % de la población de vertebrados [3].

En España se observa no sólo el aumento de las temperaturas en todo el territorio, sino también la del Mediterráneo, además del incremento del nivel de este mismo. También la dilatación del verano unos 9 días por década, que da lugar a que actualmente contemos con 5 semanas más que a comienzos de los años ochenta. Otros efectos han sido la desaparición de más de la mitad de los glaciares españoles y los cambios en la distribución, comportamientos y alimentación de la biodiversidad, entre otros factores [2].

Pero la gestión adecuada de residuos es algo al alcance de la mano de cualquier ciudadano o ciudadana de a pie.

Mezclar materiales no sólo es contaminante porque dificulta la recuperación y reciclaje de estos, sino que además el proceso de separación de residuos también es costoso y contaminante, y no siempre tiene resultados satisfactorios. Esto es debido a que muchas veces los materiales recuperados, al haberse juntado con otros, son de baja calidad y con un alto nivel de partículas no reciclables. Por eso, para facilitar este proceso encontramos contenedores especiales para los distintos residuos.

Separar los desechos de manera adecuada es una gran aportación al cuidado del medioambiente, pero como se ha comentado anteriormente, hay una gran cantidad de materiales y residuos diferentes y en ocasiones puede resultar difícil y confuso cómo deben separarse. Debido a esta dificultad surge la motivación de realizar este Trabajo de Fin de Grado. Para esto se ha querido plantear y desarrollar una aplicación de identificación de objetos que utilizando la cámara de un teléfono móvil indique la manera adecuada de desechar el residuo identificado. De esta forma, sería sencillo solventar de manera cómoda para el ciudadano las dudas que puedan surgir, fomentando así el reciclaje.

1.2. Objetivos

El objetivo principal del proyecto es el desarrollo de una aplicación para dispositivos móviles de identificación de objetos enfocado al reciclaje. A través de ella los usuarios podrán identificar diferentes residuos con el fin de solventar de manera fácil y rápida las dudas sobre cómo desecharlos correctamente, ya que la aplicación dará información sobre el material y cuál es la manera adecuada de reciclarlos.

Para conseguir su correcto funcionamiento es necesario utilizar técnicas de visión artificial, que actualmente suponen el uso de redes neuronales entrenadas. Para ello, es necesario disponer de un amplio número de imágenes de residuos etiquetadas correctamente (conocido como *dataset*) que se utilice para “enseñar” (entrenar) a la red neuronal. Conseguir dicho *dataset* se convierte en uno de los principales subobjetivos del proyecto. Con el fin de facilitar el proceso de obtención de las imágenes, surge una segunda aplicación a desarrollar. El objetivo de esta es tener una alternativa que evite el proceso tedioso y lento que puede resultar la obtención de las imágenes. Se trata de una aplicación para ordenador de generación de imágenes sintéticas a partir de modelos tridimensionales. Esta aplicación genera a partir de los modelos disponibles una serie de imágenes de cada uno hasta obtener la cantidad deseada para el entrenamiento de la red neuronal.

1.3. Herramientas utilizadas

Para comenzar, se ha utilizado Android Studio para la creación de la aplicación de identificación de objetos. Para entrenar y obtener el modelo necesario para su correcto funcionamiento se ha creado un *script* en Python utilizando las librerías de Numpy y Tensorflow; como editor se ha utilizado PyScripter.

Para desarrollar la aplicación de generación de imágenes se ha utilizado el motor de videojuegos Unity. Los modelos que se utilizan en esta aplicación han sido conseguidos desde varios orígenes tales como CGTrader¹, Free3D², TurboSquid³, 3DModelHeaven⁴ y Unity Asset Store⁵.

Por último, se ha usado GitHub⁶ como plataforma para el control de versiones y TexMaker para el desarrollo de la memoria a partir de la plantilla TEXIS.

Todo el proceso se ha realizado desde un dispositivo Windows y para las pruebas en teléfono móvil se ha usado uno con sistema operativo Android.

1.4. Plan de trabajo

El trabajo se divide en tres partes: la generación de imágenes, el entrenamiento de la red neuronal y el desarrollo de la aplicación de identificación de objetos.

La primera parte consiste en el desarrollo de una aplicación que cargue modelos 3D, separados por material, y realice numerosas imágenes a cada uno cambiándoles la posición, la rotación y el fondo para obtener diversidad en las imágenes. Estas imágenes deberán ser guardadas separadas por el material al que corresponden, igual que lo estaban los modelos al cargarlos. El desarrollo de la aplicación se divide en dos iteraciones, una primera en la que se desarrollan todas las funcionalidades principales y simultáneamente se va comprobando que todo funciona correctamente. En cambio, en la segunda iteración tiene lugar la generación del *dataset*.

Con las imágenes generadas del paso anterior tiene lugar el entrenamiento de la red neuronal, la segunda parte del proyecto. Para llevar a cabo esto se deberá investigar sobre los distintos tipos de redes neuronales y elegir la opción más adecuada para que el resultado, el modelo entrenado, sea utilizado desde una aplicación móvil. Además, se requerirá obtener algunos *datasets* de imágenes reales de los materiales para poder realizar pruebas y

¹<https://www.cgtrader.com/>

²<https://free3d.com/es/>

³<https://www.turbosquid.com/>

⁴<https://3dmodelhaven.com/>

⁵<https://assetstore.unity.com/>

⁶<https://github.com/celica02/RecyclingFinalDegreeProject>

comparaciones sobre la tasa de aciertos de la red en distintos casos.

Por último, como se ha mencionado antes, queda el desarrollo de la aplicación para dispositivos móviles que, haciendo uso de la cámara y el modelo entrenado, identifique el material del objeto al que se está enfocando y después indique al usuario cómo se debe reciclar dicho material. En este punto del proceso también deben llevarse a cabo diversas pruebas para comprobar el funcionamiento de la aplicación. Para ello se comparará la actividad de la aplicación respecto a diferentes objetos cotidianos con el objetivo de estudiar el rendimiento de la aplicación en cada caso. Esto, además, se realizará con varios de los modelos entrenados previamente para observar la diferencia en el comportamiento de cada caso.

Introduction

Motivation

Waste generation is linked to the current development model of society and constitutes one of the main environmental problems the world is facing [47]. Waste can be classified into two categories: those produced by industrial activity, called industrial waste, and those generated by human activity itself, called urban waste.

A large part of the pollution and CO₂ emissions come from large companies. For example, in 2018 25 % of emissions in Spain were generated by only ten companies [2]. However, this coursework will focus on urban waste, which is where population can mainly be aware of and take measures in this regard.

Urban solid waste problem comes from recent years increment use of non-return packaging. These packages can be made of different materials such as cellulose, glass, plastic or mixed (laminated paper, laminated fabrics, etc.) which complicates their treatment, since a previous selection and separation must be done [12].

Poor waste management can cause irreversible environmental impacts. Nowadays, it can already be seen many of the effects that seemed to come in the future [33]. Among them, it can be highlighted the increasing temperatures around the globe, the disappearance of glaciers (both in mountains and in the polar ice caps [7]) or the decrease in 68 % of vertebrates'population [3].

In Spain, temperatures have increased throughout both national territory and the Mediterranean, among its sea level increment. Also, dilation of summer around 9 days per decade, which means that it is currently 5 weeks longer than at the beginning of the eighties. Some other effects are the disappearance of more than half of the Spanish glaciers and changes in the distribution, behaviour and nutrition of biodiversity, among other factors [2].

But proper waste management is something within the reach of any citizen.

Mixing materials is not only considered polluting because it makes their recovery and recycling difficult. Waste separation is a costly and polluting

process which not always have satisfactory results. This is due to the fact that, sometimes, the recovered materials are low-quality made and come with tons of non-recyclable particles because they have been combined with other material kinds. Therefore, to facilitate this process we find special containers for each type of waste.

Separating waste properly is a great contribution to take care of the environment. As discussed above, there are a lot of different materials and types of waste, which sometimes can be difficult and confusing knowing how they should be separated correctly. Thus, due to this problem, the motivation for this project comes up. The main objective is to develop an object identification application that, using the camera of a mobile phone, indicates the appropriate way to dispose an identified waste. With this application, it would be easy for everyone to solve in a comfortable way any doubts about recycling that may appear, promoting this way proper eco-friendly attitudes.

Objetives

The main objective of the project is the development of a mobile application focused on recycling. This app identifies objects and gives information about the material and the appropriate way to recycle them. Through it, users will be able to identify different wastes to solve quickly and easily any doubts about how dispose them correctly.

To achieve its correct behaviour artificial vision techniques are needed, which currently involve the use of trained neural networks. To do so, it is necessary to have a large number of correctly labeled waste images (known as dataset) that are used to “teach” (train) the neural network. Obtaining said dataset becomes one of the main sub-objectives of the project. In order to facilitate the process of obtaining the images, it is decided to develop a secondary application. The objective of this one is to avoid the tedious and slow process that obtaining images can be. It works as a computer application that generates synthetic images from three-dimensional models. The application generates as many synthetic images, from the available models, as desired to train the neural network.

Tools

Android Studio has been used to develop the identification mobile application. To train and obtain the model necessary for the application performance, a script was coded in Python using Numpy and Tensorflow Lite libraries and PyScripter as the editor.

The image generator application was developed using the video game engine Unity. Every three-dimensional model used in this application have

been sourced from CGTrader⁷, Free3D⁸, TurboSquid⁹, 3DModelHeaven¹⁰ and Unity Asset Store¹¹.

Lastly, for the project's version control GitHub¹² has been used and for the project report development, the TeXISTemplate in TexMaker editor was chosen.

The entire process has been developed on a Windows device and the application tests were made on an Android mobile device.

Work plan

The coursework is divided into three different parts: the image generation, training the neural network and develop the mobile object identification application.

The first one is focused on the develop of an application that loads 3D models and makes numerous images for each one, changing their position, rotation and background to obtain diversity in the images. The models are organised by their material so the images generated must be arranged the same way. The application development is divided into two iterations: the first one is where all the main functionalities are developed and simultaneously being verified. Meanwhile, in the second iteration takes place the dataset generation.

Once the images are generated, the neural network training takes place, which is the second part of the project. To develop this, it is necessary to investigate the different types of neural networks and choose the most appropriate option to be used in a mobile application. In addition, it will be necessary to obtain some datasets made of real images of the materials selected. Then, make tests and comparisons to check the accuracy in different cases.

Finally, the development of the mobile devices application continues, using the trained model to identify the material of the object that is being focused on with the camera. Besides the material, it is also indicated how it has to be recycled. At this point in the coursework various tests must be done to verify the application performance. To do so, the different trained modelsaccuracy will be compared using diverse everyday objects.

⁷<https://www.cgtrader.com/>

⁸<https://free3d.com/es/>

⁹<https://www.turbosquid.com/>

¹⁰<https://3dmodelhaven.com/>

¹¹<https://assetstore.unity.com/>

¹²<https://github.com/celica02/RecyclingFinalDegreeProject>

Capítulo 2

Estado del arte

2.1. Aplicaciones de reciclaje

Con el modelo actual de sociedad, donde se ha incrementado la utilización de envases de un único uso, ha habido un aumento desmesurado de los residuos urbanos [47], cuya recuperación y reutilización es importante para disminuir el impacto medioambiental que producen. Para llevarlo a cabo, es importante la separación de los residuos desde el origen, ya que permite que sean clasificados y almacenados de tal manera que se puedan reciclar correctamente [23]. Para ello, se pide ayuda a la ciudadanía con el objetivo de fomentar la separación de residuos en el hogar.

Se tienen así diferentes tipos de residuos, que se depositan en contenedores distintos, identificados por colores. Para que tenga éxito, es necesario realizar la separación en el origen correctamente, ya que de no ser así los esfuerzos serían en vano.

Se hacen campañas muy diversas para concienciar a la población de la importancia de este proceso, además de enseñar a llevarlo a cabo. Pero no siempre es fácil saber en qué contenedor depositar ciertos residuos. Para ayudar a la población, estas campañas informativas reparten folletos o imanes para la nevera y, en última instancia, se puede recurrir a los puntos limpios de cada localidad. No obstante, con la presencia universal de teléfonos móviles en los hogares, las aplicaciones para ellos se han convertido en una vía informativa útil que está disponible en el momento en el que se necesita.

Cada país usa una gestión de recursos distinta, y por tanto las guías para los ciudadanos deben adaptarse a las idiosincrasias de cada sistema; debido a eso, las aplicaciones de ayuda están adaptadas a cada país. Puesto que abarcar todas las opciones es complejo, este trabajo se centra en el sistema utilizado en el territorio nacional.

En España, en aplicaciones de reciclaje que identifiquen objetos hay va-



Figura 2.1: Aplicaciones de reciclaje

rias. Una de ellas es “AIRE: Asistente Inteligente de Reciclaje”¹ (figura 2.1a), disponible tanto para iOS, Android y navegador, es una aplicación publicada por Ecoembes, empresa encargada del reciclaje de los residuos de los contenedores amarillo y azul en España. Se trata de un asistente virtual con el que se tiene una conversación de chat, el usuario escribe el material o el objeto que desea reciclar y el *bot* le responde explicando la forma idónea de desecharlo. Además del chat, esta aplicación también cuenta con la posibilidad de enviar imágenes y audios para la identificación del objeto.

Otra es “ReciclaYa”² (figura 2.1b), aplicación desarrollada por Carrefour. En este caso la identificación se lleva a cabo mediante el código de barras del ticket de la compra que se haya hecho. Según qué productos se hayan adquirido, indica cómo reciclar cada uno de sus envases. Esta funcionalidad está sólo disponible para algunas de las marcas principales, ya que son las propias empresas las que brindan esa información. Algunas de estas empresas sobre las que se disponen los datos son Carrefour, Coca Cola, Nestlé, Central Lechera Asturiana, Mahou San Miguel, PepsiCo, El Pozo o Pescanova, entre muchas otras.

Además de aplicaciones de identificación de residuos, también se encuentran numerosas aplicaciones enfocadas a la ayuda y el aprendizaje sobre reciclaje. Algunos ejemplos son “Residuos” (figura 2.1c) de la Generalitat de Catalunya³ en colaboración con Ecomenbes y Ecovidirio. Esta aplicación cuenta con una base de datos de residuos con su respectivo contenedor adecuado y el proceso que sigue una vez es desecharlo en este. Cuenta con dos maneras de informar sobre cada desecho. La primera es por contenedores, se selecciona el contenedor respecto al que se quiere tener más información,

¹<https://www.ecoembes.com/proyectos-destacados/chatbot-aire/>

²<https://www.reciclaya.app/es/como-funciona>

³<https://play.google.com/store/apps/details?id=cat.gencat.mobi.residus>

sobre este sale un listado de todos los residuos disponibles en la base de datos reciclables en él, y finalmente se selecciona uno de ellos. La segunda manera es introducir en el buscador el residuo. En ambos casos se explica el proceso que sufre este desecho desde el contenedor, su paso por la planta de selección y reciclaje, y en qué se transforma finalmente. Además de informar sobre la forma adecuada de deshacerse de cada residuo, también cuenta con un buscador de puntos limpios a partir de la localización del dispositivo o mediante un buscador.

Otro ejemplo es “Recicla y suma” (figura 2.1d) de Pensumo⁴. Es una herramienta pensada para incentivar el reciclaje animando al ciudadano a que se acostumbre a separar en casa los residuos que genera para luego acudir a los contenedores y reciclarlos. Pensumo es una *start-up* que recompensa económicamente el reciclaje gracias a la colaboración de distintas empresas. El objetivo principal es generar el hábito en el ciudadano, para que, animado por el incentivo económico, acabe interiorizando la necesidad de reciclar y aprenda a depositar correctamente los residuos. Los usuarios sacan fotos en el momento que depositan los residuos en el contenedor correcto y reciben una pequeña cantidad de dinero por la acción.

Alejándose un poco del reciclaje como tal, encontramos “The Planet App”⁵ por Clean Planet Ventures, SL. Es una aplicación de concienciación, aprendizaje y ayuda sobre la huella de carbono que genera cada uno. Permite al usuario calcular su huella de carbono y conocer las emisiones que genera separadas por categorías. La característica principal es la creación de un plan de sostenibilidad personalizado para adquirir hábitos y realizar acciones que disminuyan las emisiones que genera cada uno. Además, se pueden comparar las emisiones con las de distintos grupos poblacionales o las de otros usuarios.

En el lado internacional existen muchas más aplicaciones, como “My Little Plastic Footprint”⁶, de Plastic Soup Foundation. Esta es muy similar a la aplicación anterior, pero en vez de ser respecto a la huella de carbono es sobre el plástico que genera el usuario. En este caso, la aplicación indica el “Índice de Masa Plástica” (PMI por sus siglas en inglés). El concepto hace un símil con el índice de masa corporal, pero refiriéndose al consumo de plástico del usuario. Una vez calculado, comienza la denominada “dieta de plástico”, que consiste en ir adquiriendo como hábito alternativas al plástico que se utiliza normalmente. A medida que se van incorporando los consejos de la aplicación al perfil del usuario, el PMI de este se irá reduciendo.

“Grow Recycling”⁷, de Gro Play Digital, es un juego educativo para niños utilizado en colegios y guarderías en Estados Unidos y Europa para impartir educación sobre sostenibilidad. El juego consiste en alimentar a distintos

⁴<https://reciclaysuma.com/>

⁵<https://theplanetapp.com/>

⁶<https://mylittleplasticfootprint.org/>

⁷<https://www.groplay.com/apps/grow-recycling/>

cubos de basura con los residuos correspondientes para que niños y niñas puedan aprender a cuidar del planeta. Otra aplicación educativa de reciclaje es “Recycle Coach”⁸, de Municipal Media Inc. Con ella se puede aprender sobre el proceso de reciclaje, los diferentes materiales y el proceso de depósito y recogida de residuos en distintas localidades del mundo. Además, cuenta con *tests*, artículos y actividades para aprender sobre el desechado de residuos.

Como se ha descrito en la introducción, el objetivo del TFG es hacer una aplicación de este tipo pero que indique cómo reciclar un objeto a partir de una imagen tomada con el móvil. Para eso es necesario realizar una identificación de objetos, sobre lo que se describen a continuación varias aplicaciones de ejemplo.

2.2. Identificación de objetos e imágenes

La detección e identificación de objetos es un campo de la visión artificial y el procesamiento de imágenes. El objetivo de la visión computacional es desarrollar algoritmos que reciban una imagen de entrada y produzcan una interpretación describiendo los objetos presentes, en qué posición y la relación espacial tridimensional entre ellos. Durante mucho tiempo no hubo unanimidad sobre cuál era el mejor paradigma para la identificación de objetos [4] aunque hoy en día está generalizado el uso de redes neuronales. Su evolución ha permitido que actualmente sea posible detectar y reconocer objetos en conjuntos de miles de imágenes complejas.

La detección, seguimiento y reconocimiento de objetos en imágenes es uno de los problemas claves en visión computacional [10]. Normalmente, en cada imagen sólo aparece una pequeña cantidad de los objetos posibles, pero estos pueden encontrarse en una gran cantidad de posiciones y escalas que se deben tener en cuenta [5].

Esto es algo utilizado en diversos ámbitos que requieren la identificación de objetos, personas o caras. Un ejemplo es VOCUS (Visual Object detection with a CompUtational attention System) [13], un sistema capaz de seleccionar automáticamente secciones de interés en imágenes y detectar objetos específicos. Cuenta con dos modos de trabajo, un primer modo de exploración en el que no se especifica un objetivo y en el que se buscan zonas de interés. Son consideradas zonas de interés aquellas en las que hay grandes contrastes o características únicas que las hacen destacables frente a lo demás (por ejemplo una oveja negra en un rebaño de ovejas blancas). El segundo modo es de búsqueda con un objetivo definido basándose en la información previamente aprendida.

Otro ejemplo es el reconocimiento de matrículas de los coches presentes

⁸<https://recyclecoach.com/>

en una imagen [38]. En este caso se presenta una doble identificación, ya que primero debe localizarse la matrícula en la imagen y posteriormente sobre esta se tiene que llevar a cabo la segmentación e identificación de los caracteres que la componen.

SynthDet [24] es un proyecto cuyo fin es identificar los distintos productos contenidos en carritos de la compra con el objetivo de facturar automáticamente las adquisiciones de los clientes en supermercados. Está basado en Amazon Go [40], supermercado de la empresa Amazon, ubicado en Seattle (Estados Unidos). El modelo de Amazon Go se basa en un sistema de cámaras y sensores distribuidos por el establecimiento que analizan los movimientos de los clientes. Estos, además de obtener la información necesaria para realizar los cobros, también extraen información sobre las pautas de comportamiento de los consumidores [40]. SynthDet propone realizar la misma tarea de identificación y facturación de productos, pero entrenando el modelo a partir de imágenes sintéticas. Esta forma de entrenamiento se trata con mayor profundidad en la sección 2.5.

Otros proyectos relacionados son la detección de drones [43] y la identificación de piezas industriales [8], que cuentan con un trasfondo similar de identificar objetos en entornos reales.

Pero la inclusión y el desarrollo de la detección y la identificación de objetos no se limita al entorno de la investigación y a usos específicos, sino que en la vida cotidiana también se percibe un crecimiento de este ámbito. Actualmente el uso de la detección e identificación de objetos se está expandiendo por más aplicaciones. Una de las más conocidas y extendidas es Google Lens⁹, la cual tiene diversas funcionalidades dentro del mundo de la detección de objetos. Esta aplicación es una de las más completas en este dominio, a partir de una imagen o de la cámara del dispositivo, escanea y traduce textos; identifica objetos y busca en Google otros similares; reconoce lugares y edificios, ofreciendo información sobre ellos; detecta animales y plantas; y registra operaciones matemáticas sobre las que ofrece explicaciones y resultados de la web.

Otra aplicación de Google con identificación de objetos es Google Fotos¹⁰, una plataforma de intercambio y almacenamiento de fotografías y videos. Puede utilizarse en conjunto con Google Lens para tener todas las funcionalidades mencionadas anteriormente sobre las imágenes almacenadas en la aplicación, pero además Google Fotos cuenta con su propia funcionalidad de identificación de texto, objetos y personas. En la aplicación pueden realizarse búsquedas que devuelven una selección de imágenes y videos en los que aparece la persona, objeto o texto buscado, además de funcionar con posturas y acciones.

Otras aplicaciones similares, aunque algo más rústicas, son las recogidas

⁹<https://lens.google/intl/es-419/>

¹⁰<https://www.google.com/intl/es/photos/about/>

en el paquete de Microsoft Office. En este caso la identificación se presenta como una funcionalidad de accesibilidad para personas con discapacidad visual. Se trata del texto alternativo, disponible para imágenes, formas, vídeos, gráficos y tablas, entre otros. En algunas de sus aplicaciones Microsoft va un paso más allá y genera el texto alternativo de las imágenes automáticamente, identificando personas, objetos y situaciones.

2.3. Redes neuronales

Todos los sistemas mencionados en el apartado anterior (2.2) están implementados haciendo uso de redes neuronales. Las redes neuronales artificiales son una técnica de clasificación del aprendizaje automático, disciplina de la Inteligencia Artificial (IA) cuyo objetivo es la creación de mecanismos capaces de aprender de forma autónoma. Estas surgen inspiradas por la funcionalidad sofisticada del cerebro humano, donde miles de millones de neuronas se encuentran interconectadas y procesan información de manera paralela. Tratan de emular el comportamiento del cerebro humano caracterizado por el aprendizaje a través de la experiencia y la extracción de conocimiento a partir de un conjunto de datos [11]. Además, simulan las características básicas de las redes neuronales biológicas; el procesamiento paralelo, la memoria distribuida y adaptabilidad. Esto permite a las redes artificiales aprender y generalizar a partir de un conjunto de datos de relación matemática desconocida [19]. De esta forma, adquieren, almacenan y utilizan conocimientos basados en la experiencia en lugar de ser programados de manera explícita [59].

Aunque existen muchas definiciones para las redes neuronales, una de las más extendidas establece que una red neuronal artificial se define como un grafo dirigido, donde los nodos representan las neuronas y las aristas la sinapsis. Aquellos nodos que no cuentan con conexiones entrantes son consideradas las neuronas de entrada, y los que no tiene conexiones salientes las de salida. El resto de neuronas se consideran capas ocultas, las cuales cuentan con conexiones de entrada y de salida [11, 15, 19, 37]. En la figura 2.2 se muestra la estructura habitual de una red neuronal artificial con una capa oculta.

Las neuronas están conectadas entre sí mediante líneas. A cada una de estas conexiones se les asocia un valor numérico llamado peso que indica su relevancia dentro de la red. A través de dichas conexiones las neuronas reciben datos de entrada. Estos se multiplican por sus respectivos pesos y se lleva a cabo un sumatorio, el valor obtenido se aplica en la función de activación cuyo resultado es transmitido a otra neurona [55, 46, 6]. Durante el aprendizaje de la red se modifica el valor de los pesos asociados a las conexiones entre neuronas con el fin de generar una salida a partir de unos datos de entrada. Se considera que el aprendizaje ha terminado cuando los

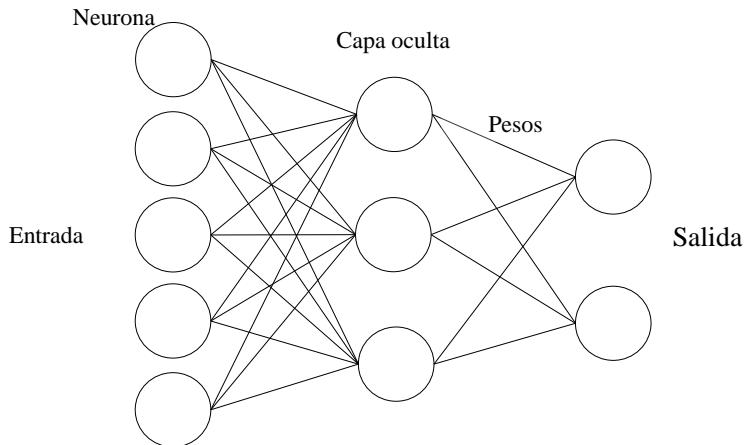


Figura 2.2: Estructura de una red neuronal artificial.

valores de los pesos permanecen estables.

Algunos autores consideran que las redes neuronales artificiales se inspiraron en el trabajo de Ramón y Cajal quien, en 1888, demostró que el sistema nervioso está compuesto por una red de células individuales conectadas entre sí, las neuronas [11]. Otros nombran a Alan Turing como pionero al estudiar el cerebro como una forma de ver el mundo de la computación. Independientemente, en todos los casos se toma a Warren McCulloch (neurobiólogo) y Walter Pitts (estadístico) como los primeros teóricos que concibieron fundamentos de la computación neuronal en el artículo "*A logical calculus of Ideas Imminent in Nervous Activity*" en 1943 [32]. Posteriormente, en 1958, Frank Rosenblatt comenzó el desarrollo de lo que terminaría conociéndose como perceptrón [42].

El perceptrón es un sistema clasificador de patrones capaz de reconocer patrones similares a los del entrenamiento pero que no había visto antes. Sin embargo, contaba con ciertas limitaciones, la más importante fue que no era capaz de resolver clasificar clases no separables linealmente, como por ejemplo la función lógica OR exclusiva¹¹. Esto fue demostrado matemáticamente en los años 60 por Minsky y Papert [34], lo que desencadenó un fuerte desinterés en la computación neuronal.

No fue hasta principios de los años 80 cuando se consiguió devolver el interés en este campo con el desarrollo de una red recurrente por parte de John Hopfield [20] y el redescubrimiento de David E. Rumelhart, Geof-

¹¹XOR u OR exclusiva es una puerta lógica cuya salida es verdadera si una, y sólo una, de las entradas es verdadera. Si ambas son falsas o ambas son verdaderas la salida es falsa.

frey E. Hinton y Ronald J. Willieams [44] del algoritmo de aprendizaje de propagación hacia atrás (*backpropagation*) planteado por Paul Werbos en 1974 [6, 27, 31, 14].

Existen dos tipos de redes neuronales principales, las redes convolucionales y las recurrentes. Las primeras se desarrollaron para el reconocimiento de imágenes y son útiles para distinguir elementos esenciales en una entrada de la red compleja [39]. Una particularidad de estas redes es la operación de convolución¹² que se realiza en las primeras capas utilizándose como filtro, esto permite que tenga muchas aplicaciones en el tratamiento de imágenes [30]. Por otro lado, las redes neuronales recurrentes son aquellas en las que la salida de una neurona vuelve en forma de entrada, por lo que puede haber ciclos en las conexiones entre las neuronas de cada capa. Cuentan con la peculiaridad de que no sólo aprenden de los datos sino también de secuencias de estos. Todo esto las convierte en buenas candidatas para trabajar con datos con dependencias estructurales en una dimensión, como el texto, o el audio [39, 17].

También se lleva a cabo una distinción en las redes neuronales según el paradigma de aprendizaje que siguen. En función de él, existen tres tipos principales, las redes neuronales supervisadas, las no supervisadas y las de aprendizaje por refuerzo. Las redes neuronales supervisadas trabajan con datos etiquetados y se caracterizan por la presencia de un agente externo que controla el entrenamiento denominado supervisor. Este determina la respuesta que debería generar la red a partir de una entrada determinada y controla la salida de la red. En caso de que el resultado no coincida con el deseado se modifican los pesos de las conexiones con el fin de conseguir una salida aproximada a la esperada. En cambio, las redes con aprendizaje no supervisado no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red recibe datos sin etiquetar, es decir, no existen datos de salida correspondientes a una determinada entrada. En este modelo se agrupan los datos según sus propiedades y patrones, su objetivo es encontrar una estructura o configuración en los datos. Por último, el aprendizaje por refuerzo se basa en mejorar los resultados de los modelos mediante retroalimentación que recibe del entorno según la salida seleccionada. El *feedback* que recibe indica con qué grado de satisfacción cumple la salida, conocida como acción, con el objetivo [31, 46, 49].

El concepto de red neuronal deja abierta la decisión de su estructura: cuántas capas ocultas poner, de qué tamaño poner cada una, etcétera. Con el tiempo se han ido descubriendo estructuras de redes que funcionan bien para dominios concretos, por lo que se han utilizado una y otra vez y las distintas librerías para el uso de redes neuronales las incorporan entre sus opciones

¹²Una convolución es un operador matemático que transforma dos funciones f y g en una tercera que representa la magnitud en la que se superponen f y una versión trasladada e invertida de g .

predefinidas. Algunas de las redes neuronales más expandidas actualmente son AlexNet, VGGNet, MobileNet e GoogLeNet.

AlexNet [26] es una red ampliamente utilizada que se dio a conocer en la competición de ImageNet [45], donde consiguió el primer puesto con una tasa de error del 15.3 % (2012). Fue la primera red neuronal en utilizar ReLu¹³ como función de activación en vez de la función sigmoide¹⁴. Este cambio mejoró notablemente la velocidad de entrenamiento de las redes de aprendizaje profundo [9, 22, 57].

La segunda red destacada es VGGNet. Esta también saltó a la fama gracias a ImageNet en 2014. La diferencia más destacable con AlexNet es la gran cantidad de capas convolucionales que tiene. Además de una arquitectura más profunda destacó también por su simplicidad [50, 57, 36].

GoogLeNet, también conocida como InceptionNet, fue desarrollada por Google, siendo de las primeras redes que no se basó en añadir más capas convolucionales para mejorar la red, sino que cambió la estructura de estas usando filtros de varios tamaños. El objetivo era tener filtros de diferentes tamaños para un mismo objeto. Esto provocó que la red fuera más “ancha” en vez de más profunda [51].

La última red es MobileNet, esta también fue desarrollada por Google. Surgió con la intención de adaptar GoogLeNet a dispositivos móviles. Su objetivo fue reducir el número de cálculos y parámetros pero manteniendo unos resultados similares [48, 21].

2.4. TensorFlow y TensorFlow Lite

En 2011 el equipo de investigación encargado de la inteligencia artificial de aprendizaje profundo de Google, Google Brain, comenzó a investigar el uso de redes neuronales a gran escala para utilizarlo en los productos de la empresa. Como resultado desarrollaron DistBelief, un sistema escalable de entrenamiento e inferencia utilizado para GoogLeNet [51]. A partir de la experiencia obtenida en el desarrollo de DistBelief, y con un entendimiento más completo de las necesidades y requisitos necesarios para generar un sistema mejor, desarrollaron TensorFlow¹⁵.

Este es un entorno de trabajo de código abierto de aprendizaje automático que utiliza grafos de flujo de datos. En estos grafos los nodos simbolizan las operaciones matemáticas, tales como sumas, multiplicaciones, factorización de matrices, y demás; mientras que los bordes representan los conjuntos de datos multidimensionales (tensores) [25].

TensorFlow puede ejecutarse en una gran variedad de plataformas dife-

¹³ReLu transforma los valores de entrada anulando los negativos y manteniendo los positivos.

¹⁴La función sigmoide transforma los valores de entrada a una escala entre 0 y 1.

¹⁵<https://www.tensorflow.org/>

rentes, desde máquinas sencillas de una o varias GPUs, como los dispositivos móviles, hasta sistemas a gran escala de cientos de máquinas especializadas que cuentan con miles de GPUs. Su API y sus implementaciones de referencia fueron lanzadas en noviembre de 2015 como un paquete de código abierto bajo una licencia de Apache 2.0 [1, 53, 58].

Para utilizar modelos de TensorFlow en dispositivos móviles, microcontroladores, sistemas embebidos o relacionados con IoT (Internet of Things), Google desarrolló la librería de TensorFlow Lite. Esta permite la optimización de los modelos para reducir su latencia y tamaño [35]. Su entrenamiento se lleva a cabo en una computadora, cuyo resultado se traslada posteriormente al dispositivo sin necesidad de utilizar un servidor. Esto se realiza mediante aprendizaje por transferencia utilizando la librería Model Maker. Este tipo de aprendizaje aprovecha el conocimiento aprendido por la red en casos previos, permitiendo reducir la cantidad de datos de entrenamiento necesarios y acortar el tiempo de entrenamiento. Actualmente, esta librería admite varios modelos previamente entrenados como EfficientNet-Lite [52], MobileNet [48] o ResNet-50 [18].

Como resultado del entrenamiento se generan dos archivos en el ordenador. El primero contiene las etiquetas para la clasificación, se trata de un archivo de texto plano con una lista de todos los objetos detectables por la red. El segundo, con extensión *.tflite*, es el modelo entrenado para TensorFlow Lite. Ambos archivos deben importarse en la aplicación móvil para su uso, la cual puede ser para Android, iOS o Raspberry Pi. Estas deben estar desarrolladas en Java; Swift u Objective-C; y Python, respectivamente, ya que son los lenguajes para los que está disponible la API.

2.5. Generación sintética de imágenes

Un problema del entrenamiento de redes neuronales es que se requieren muchos datos, y en caso de que la red esté enfocada a la identificación de objetos estos datos de entrenamiento deben de ser imágenes etiquetadas. Para facilitar la obtención de los *dataset* en este Trabajo de Fin de Grado se plantea la posibilidad de utilizar imágenes sintéticas.

Actualmente hay un contacto continuo con los gráficos generados informáticamente. Desde el desarrollo de videojuegos hasta el diseño de un reactor nuclear se apoyan en la informática gráfica. Esto abarca dos grandes campos de aplicación. Por un lado se aplica en el proceso de reconstruir un modelo 2D o 3D a partir de una imagen, esto es denominado procesamiento de la imagen. El proceso contrario es la síntesis de la imagen, consiste en la construcción sintética de imágenes a partir de modelos de dos o tres dimensiones. Dichos modelos pueden haber sido producidos por métodos artificiales o recogidos del mundo real [16].

Una vez que se dispone del modelo, tiene lugar la generación de la ima-

gen sintética. Históricamente el proceso requería mucho tiempo si se quería calidad fotorrealista, pero actualmente se pueden conseguir buenos resultados en tiempo real. Aun así, existen principalmente dos aproximaciones. La primera es usar herramientas de modelado con las que se crean los objetos en 3D y se ven representados, y la segunda es usar esos modelos para mostrarlos desde diferentes puntos en tiempo real, algo que hoy habitualmente significa usar motores de videojuegos.

SYNTHIA [41] es un *dataset* de imágenes fotorrealistas generadas a partir de frames obtenidos de la renderización de un mundo virtual formado a partir de modelos tridimensionales. Estas imágenes además van acompañadas de anotaciones semánticas a nivel de *pixel* que etiquetan los distintos objetos presentes en la escena, distinguiendo hasta 13 clases distintas. Este *dataset* está orientado al entrenamiento de coches autónomos, sistemas de conducción asistida y proyectos relacionados, como complemento para diferentes *datasets* reales (Camvid, KITTI, U-LabelMe, CBCL).

Otro ejemplo similar, es la generación de imágenes efectivas para el entrenamiento de detección de objetos partiendo de un conjunto pequeño de imágenes reales y el modelo tridimensional del objeto a identificar [43]. Los parámetros pueden reutilizarse para generar un número ilimitado de imágenes de entrenamiento del objeto de interés en poses arbitrarias. Por cada imagen real se computan cinco parámetros de la pose, tres de orientación y dos de traslación que permite colocar el modelo 3D en la posición deseada. Una vez se obtiene la imagen, para hacerla lo más similar a la imagen real, pasa por un postprocesamiento que involucra agregar desenfoque, para simular la lejanía y el movimiento del objeto; ruido, que imita el ruido que añade la cámara al tomar la fotografía; y propiedades del material del objeto. Esto permite que las imágenes generadas sintéticamente sean más similares a las reales, obteniendo un *dataset* sintético muy similar a uno real.

Un proyecto basado en el anterior, es la detección de objetos en contextos industriales [8]. En este proyecto se parte del modelo renderizado de una pieza utilizada en el entorno industrial con el que se genera un *dataset*. Como en el caso anterior, para añadir más diversidad e identificar las piezas en un mayor número de situaciones, las imágenes se generan con diferentes fondos creados a partir de una selección de imágenes de localizaciones reales. Posteriormente, además, se altera la iluminación y se añaden sombras de distintas formas e intensidades creando imágenes desenfocadas o en movimiento.

Perception [54] es un paquete de Unity en desarrollo que proporciona herramientas para generar *datasets* sintéticos a gran escala. Está planteado para tareas de aprendizaje automático, tales como detección de objetos, segmentación semántica y más. Este *dataset* está formado por *frames* capturados por sensores simulados, y que posteriormente se utilizan para el entrenamiento y la validación del modelo. Perception, además de permitir al usuario que produzca sus propios datos para la tarea que requiera, ofrece

una cantidad de etiquetas comunes ya generadas para facilitar la creación de datos sintéticos. Además de la generación de *datasets* sintéticos incluye herramientas para generar *keypoints*, poses y animaciones aleatorias. Este paquete puede utilizarse para distintos de proyectos, desde simulación de coches autónomos o *smart cities*¹⁶, pasando por demostraciones de recoger y colocar objetos mediante un brazo robótico¹⁷, hasta detección de objetos utilizando únicamente imágenes sintéticas para el entrenamiento.

Esto último hace referencia al proyecto SynthDet [24], ya mencionado anteriormente en la sección 2.2. En este caso se utiliza Perception para crear el *dataset* utilizado en el entrenamiento de la red neuronal. Generan imágenes de productos habituales de supermercados con el fin de estudiar la viabilidad de los supermercado sin cajas registradoras. Este objetivo se basa en el supermercado Amazon Go [40], creado por Amazon en 2017.

2.6. Unity

Unity es una herramienta de creación de videojuegos desarrollada por Unity Technologies. Es ampliamente utilizada tanto dentro de este ámbito como fuera de él debido a la cantidad de funcionalidades que ofrece de manera intuitiva y sencilla. Otros campos en los que se utiliza son la realización de contenido cinematográfico, transporte y producción, e incluso en arquitectura, ingeniería y construcción. No obstante, su objetivo principal es el desarrollo de juegos y aplicaciones para un gran abanico de plataformas; desde PC, Mac y Linux, pasando por Android e iOS, hasta consolas de videojuegos como PlayStation 4, Xbox One, Nintendo Switch o videojuegos en la nube como Google Stadia.

La creación de contenido se realiza mediante el editor visual, que forma la interfaz, y la programación se crea vía *scripting*. Uno de los puntos a favor de utilizar motores de videojuegos es que permiten a los usuarios centrarse en el desarrollo de su aplicación sin necesidad de llevar a cabo partes más complejas, como la simulación de físicas o el renderizado, ya que es el motor quien se encarga de ello.

Unity utiliza una arquitectura basada en componentes. El motor proporciona unos propios (*Transform*, *RigidBody*, etc.) y permite que los desarrolladores creen los suyos utilizando C#. El ciclo de vida de los componentes está definido por el motor, de manera que durante la ejecución de la aplicación, mediante la clase base *MonoBehaviour*¹⁸, se va llamando a métodos específicos para notificar el avance del juego y darles la oportunidad de actuar. Se tiene así métodos como *Start()*¹⁹, método que se llama en el frame

¹⁶<https://github.com/Unity-Technologies/Unity-Simulation-Smart-Camera-Outdoor>

¹⁷<https://github.com/Unity-Technologies/Robotics-Object-Pose-Estimation>

¹⁸<https://docs.unity3d.com/ScriptReference/MonoBehaviour.html>

¹⁹<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html>

en el que se inicializa *script*, o `Update()`²⁰, que se llama en todos los frames.

²⁰<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html>

Capítulo 3

Aplicación de generación de imágenes

RESUMEN: Para dotar a la aplicación de las capacidades de reconocimiento de imágenes, es necesario pasar por un proceso previo de entrenamiento con cientos o miles de ellas correctamente etiquetadas. La obtención de un conjunto de entrenamiento con las características requeridas es una tarea complicada. En este capítulo se describe y desarrolla una opción alternativa que permite generar esas imágenes de forma sintética a partir de modelos tridimensionales.

3.1. Introducción

El objetivo del proyecto es reconocer e identificar distintos desechos y su material principal, e indicar cómo deben reciclarse adecuadamente. Para ello, se utilizan técnicas de visión artificial, lo que conlleva el entrenamiento de una red neuronal. Eso significa que se requieren fotos correctamente etiquetadas, indicando lo que contienen, para que el sistema aprenda a reconocerlas. Ese entrenamiento es crítico para que todo funcione, y para ello se necesita un gran número de imágenes.

Conseguir tantas imágenes, distintas y claras, supone un trabajo costoso y lento; sin tener en cuenta el almacenaje de estas. Contamos con varias opciones para conseguir las imágenes. Una de ellas, y quizás la más obvia, es descargarlas de la red. Esta opción dista de ser la ideal debido a lo tedioso que resulta este procedimiento; buscar, seleccionar y descargar una a una miles de imágenes que cuenten con una buena resolución es un proceso muy lento.

Otra forma es realizar las fotografías personalmente o llevar a cabo un vídeo y utilizar los *frames* como imágenes de entrenamiento. El principal

problema de estas opciones es que no sólo se necesitan muchas imágenes, sino también mucha variedad. Para que la red aprenda a reconocer un determinado tipo de objeto, por ejemplo botellas, se necesitan fotografías de muchas botellas distintas, lo que supone un coste de adquisición que puede ser significativo. Una opción sería ir al establecimiento y realizar las fotos o el vídeo allí sin comprarlas. Si se realizan en un establecimiento existe el riesgo de violar la protección de datos de los clientes y los trabajadores de alrededor, además de que no esté permitido realizar fotografías ni grabaciones dentro del comercio. Otro factor menos significativo es que al tratarse de una aplicación de reciclaje sería preferible que se trate de objetos ya usados, como una lata abierta, una botella vacía o un papel arrugado.

Hay una opción adicional para evitar el laborioso proceso de conseguir imágenes etiquetadas: aprovechar trabajos ya realizado. En particular, los avances en aprendizaje automático durante los últimos años han permitido que se convierta en un campo más accesible para interesados con diferentes niveles de conocimiento. Esto ha provocado un gran aumento de los materiales disponibles de manera libre de este campo.

Una posibilidad que se presenta es utilizar *datasets* de libre uso como COCO (Common Objects in Context) [28]. Aunque este resulta no ser tampoco la opción perfecta, es un acercamiento a una posible solución. Estos *datasets* tan amplios cuentan con tal diversidad de objetivos para identificar que llegan a tener decenas de miles de imágenes. Para poder utilizarlo en este proyecto sería necesario examinarlo entero y seleccionar y separar aquellas que se pudieran utilizar. Esto, como en los casos anteriores, es una tarea sumamente lenta sin ninguna garantía de obtener imágenes suficientes.

Igual que existen *datasets* tan amplios, también los hay pequeños y enfocados a la detección de uno o pocos objetos. A pesar de la limitación presente al tener que encontrar *datasets* específicos para cada objeto que se quiera identificar, resulta la opción más accesible entre las mencionadas.

Otro ámbito en el que se ha conseguido una gran mejoría en las últimas décadas es en los gráficos de contenidos digitales audiovisuales, ya sean películas, videojuegos, videoclips, anuncios, etc. Esta mejora llega hasta tal punto que a veces puede resultar difícil distinguir entre realidad y CGI (*Computer-generated imagery*). Teniendo en cuenta esto, las dificultades mencionadas en la obtención de imágenes y las oportunidades que ofrece la automatización, no se ha querido desaprovechar la posibilidad de explorar sus posibilidades.

En la sección 2.5 se describieron algunos trabajos existentes en los que se han usado imágenes sintéticas para entrenar redes neuronales. Para este Trabajo de Fin de Grado se propuso comprobar y comparar su eficacia entrenando la red neuronal con distintos porcentajes de imágenes reales y sintéticas. En un extremo, un dataset para entrenamiento puede estar compuesto únicamente por fotografías etiquetadas de objetos reales. Como se ha

comentado, conseguirlas puede resultar muy costoso. La ventaja es que el entrenamiento debería conseguir resultados sumamente satisfactorios, asumiendo la obtención de un *dataset* suficientemente bueno.

En el otro extremo está el entrenamiento a partir únicamente de imágenes sintéticas. El coste de adquisición debería ser, en principio, mucho menor. El riesgo que se corre es que el resultado también lo sea si la calidad de las imágenes resulta ser pobre. Para conseguir un entrenamiento adecuado, las imágenes deben tener una calidad aceptable, lo que incrementa la dificultad de encontrar modelos 3D adecuados, o, en su defecto, el coste de generarlos.

La pregunta que se plantea es si existe un punto medio idóneo, en el que se mezclen ambos extremos. En lugar de utilizar un *dataset* grande de fotografías a objetos físicos, o de imágenes sintéticas realistas, se plantea si es viable utilizar un *dataset* reducido de fotos reales enriquecido con imágenes sintéticas de calidad intermedia. Para obtener una respuesta, es necesario realizar en primer lugar una aplicación para generar las imágenes sintéticas, tal y como se describe a continuación. Su uso y el análisis de los resultados se detallará en el capítulo 4.

3.2. Aplicación

La aplicación de generación de imágenes ha sido desarrollada utilizando el motor de videojuegos Unity. Las características principales de la aplicación son la carga de modelos tridimensionales y la toma de capturas. La aplicación cuenta con un Game Manager que se encarga de la gestión del ciclo de vida de la aplicación y las variables principales (rutas de directorios, cantidad de imágenes a generar o su extensión). Toma este nombre ya que es el que suele darse en los juegos al gestor global, y se ha mantenido por inercia de uso de Unity.

Al importar los modelos en el motor, es necesario crear, para cada uno de ellos, un objeto reutilizable que facilite trabajar con él de forma sencilla. En Unity esto se consigue a través de los denominados *prefabs*. En el proyecto, los *prefabs* creados se han organizado en distintas carpetas según el material al que pertenecen (metal, plástico, vidrio) lo que facilitará posteriormente el guardado de las capturas generadas.

Durante el tiempo de vida de la aplicación se accede a cada una de estas carpetas para cargar, uno a uno, los *prefabs* contenidos en ellas. En cada frame se toma una captura de la escena, en la que se ha instanciado el *prefab* con el modelo correspondiente. El modelo es colocado con una posición y rotación aleatorios diferentes en cada captura. De esta forma, a partir de un mismo objeto se obtienen imágenes muy distintas, ya que son capturados desde diversos ángulos y distancias. Para que la posición aleatoria generada no quede fuera del campo de visión de la cámara, se establecen unos valores máximos y mínimos para cada eje de coordenadas. Aún así, para prevenir



(a) Un plano de fondo



(b) Dos planos de fondo

Figura 3.1: Capturas de ejemplo de los fondos

posibles errores, antes de hacer la captura se comprueba que el modelo se encuentra dentro del campo de visión.

Para generar más diversidad en las imágenes, cada una cuenta con un fondo generado de manera aleatoria. Este se compone de dos planos superpuestos, separados levemente para que no se combinen, a los que se les asigna una textura de manera aleatoria. Uno de ellos siempre está visible, mientras que el otro sólo se muestra a veces, configurado con posición y rotación aleatorias (figura 3.1).

Una vez establecido el objeto y el fondo, se toma la captura. Estas, igual que los modelos, se guardan separadas en carpetas según el material. Esto significa que, por ejemplo, las capturas de un objeto de metal se guardan en una carpeta llamada **metal**. Como con los modelos, todas las carpetas de los distintos materiales se almacenan en una carpeta raíz.

Puesto que las imágenes generadas van a ser utilizadas para el entrenamiento de una red neuronal supervisada, todas ellas deben de estar correctamente etiquetadas. Debido a que el objetivo de la aplicación no es identificar todos los objetos presentes en una imagen, sino identificar únicamente el material del objeto principal, no es necesario tener diversos objetos en cada imagen y delimitar cada uno de ellos con su etiqueta correspondiente. En este caso, el etiquetado se hace por jerarquía de carpetas. Esto quiere decir que hay una carpeta raíz que contiene distintas subcarpetas, una para cada tipo de material identifiable por la red; y dentro de cada una de estas car-

petas se almacenan todas las imágenes correspondientes a esa etiqueta. Para el entrenamiento de la red simplemente hay que indicar la ruta de la carpeta raíz y espera que dentro de esta estén las carpetas que serán las etiquetas con las imágenes de entrenamiento en su interior .

Un último punto importante es la unicidad de las capturas. Para ello se guardan usando el nombre del modelo añadiéndole la hora, minutos, segundos y milisegundos del momento en que ha sido tomada. Las capturas pueden exportarse en tanto en formato PNG como JPG, mediante la configuración del Game Manager. PNG es un formato sin pérdida que permite imágenes con fondo transparente. Puesto que esa propiedad no es necesaria en este proyecto, y que los resultados son similares con ambos formatos, es preferible el uso de JPG debido a que los archivos ocupan menos espacio. Esto es importante ya que se necesita un gran número de imágenes para el entrenamiento de la red neuronal y así no se ocupa tanto espacio en la máquina.

3.2.1. Modelos 3D

Como se ha comentado en la sección 2.5, para generar las imágenes sintéticas se parte de modelos tridimensionales. Todos los modelos seleccionados y utilizados en este proyecto son de libre uso, conseguidos desde varias páginas dedicadas a esto (CGTrader¹, Free3D², TurboSquid³, 3DModelHeaven⁴ y Unity Asset Store⁵). El formato con el que se ha trabajado es FBX, el más extendido debido a su interoperabilidad entre aplicaciones de creación de contenido digital. Permite que los recursos tridimensionales tengan la mínima pérdida de datos entre los distintos programas de edición 3D.

Existen varias dificultades principales con la obtención de modelos tridimensionales. En primer lugar está el realismo del modelo. Puesto que las capturas son para el entrenamiento de una red que se va a usar sobre objetos reales, es importante que los modelos utilizados sean lo más fiel posible al objeto real que representan. La creación de modelos tridimensionales de este tipo conlleva un gran esfuerzo y trabajo por parte de un profesional que los cree. Por lo tanto, es comprensible que sea complicado encontrar este tipo de modelos de manera libre y gratuita. No obstante, la hipótesis planteada es que el uso de modelos de calidad intermedia, pero de bajo costo, se compensa en el entrenamiento al incorporar fotografías reales.

El segundo inconveniente surge al realizar la importación de los modelos en Unity, donde salen a relucir fallos de los modelos en cuanto a geometría, texturas y materiales. Esto, en algunos casos, termina provocando la

¹<https://www.cgtrader.com/>

²<https://free3d.com/es/>

³<https://www.turbosquid.com/>

⁴<https://3dmodelhaven.com/>

⁵<https://assetstore.unity.com/>



(a) Imagen de referencia del modelo, obtenido desde CGTrader de aseryith



(b) Modelo al ser cargado.

Figura 3.2: Imágenes de ejemplo de un modelo con transparencia online y en el editor

inutilidad de los modelos, lo cual supone que sea necesario su sustitución.

Los modelos en los que más problemas se encuentran son aquellos que presentan transparencias, como el vidrio o algunos plásticos. Al ser modelos que no requieren texturas, se dificulta su importación en Unity, ya que el material tiene que configurarse desde cero. La figura 3.2 muestra un ejemplo, donde la imagen asociada al modelo en la página de descarga es la de la figura 3.2a⁶. Sin embargo, al importar el modelo en Unity la transparencia se pierde y el objeto aparece opaco (figura 3.2b). Conseguir el mismo tipo de material transparente o translúcido en Unity requiere el ajuste manual mencionado. Esto provoca que la cantidad de modelos útiles encontrados se vea mermada considerablemente. El resultado de estas dificultades es que en este trabajo finalmente todos los modelos sintéticos utilizados fueron de objetos metálicos, algo que tendrá implicaciones en las pruebas descritas en el capítulo 4.

Un último factor, es la extensión de los modelos. Aunque generalmente se encuentran en formato FBX, u OBJ en su defecto, hay muchos modelos que mantienen la extensión del *software* utilizado para crearlos, y por lo tanto no pueden utilizarse fuera de este.

⁶<https://acortar.link/G4Gbz>

| Materiales | Imágenes reales | Imágenes sintéticas |
|------------|-----------------|---------------------|
| Plástico | 2072 | No |
| Vidrio | 917 | No |
| Metal | 1203 | Sí |

Tabla 3.1: Imágenes disponibles por material

3.2.2. *Dataset*

Al final del proceso explicado durante el presente capítulo se recopilaron varios conjuntos de imágenes. De imágenes reales se obtuvieron varios *datasets* desde la página Kaggle⁷. En particular, se consiguieron *datasets* de tres materiales diferentes (plástico, vidrio y metal) cada uno con aproximadamente 1000 imágenes. El número exacto de imágenes puede consultarse en la tabla 3.1.

Como se ha comentado previamente, para generar las imágenes sintéticas sólo se cuenta con suficientes modelos de objetos de metal, teniendo disponibles en total 40 modelos diferentes. Así que las imágenes generadas a partir de ellos se mezclan con las reales de este material en el entrenamiento de la red. Puesto que se desconoce con qué proporciones se obtiene el mejor resultado se llevarán a cabo pruebas de rendimiento. Tanto estas, como el entrenamiento de la red, se explican en el capítulo 4.

⁷<https://www.kaggle.com/>

Capítulo 4

Entrenamiento y selección de *dataset*

RESUMEN: Utilizando los conjuntos de imágenes reales y sintéticas obtenidos, se procede al entrenamiento necesario para dotar a la aplicación de la capacidad de reconocimiento de imágenes. Para elegir la mejor relación proporcional entre reales y sintéticas se realizan una serie de ensayos, cuyos resultados se comparan y estudian para seleccionar la mejor relación entre facilidad de obtención del *dataset* y rendimiento de la red.

4.1. Experimento

Como ya se ha comentado previamente, para el entrenamiento de las redes neuronales usadas en visión artificial es necesario el uso de *datasets* con gran cantidad de imágenes etiquetadas, donde se indique el objeto que representa cada una. Esos *datasets* se utilizan para el entrenamiento de la red neuronal siguiendo el procedimiento habitual del aprendizaje máquina supervisado.

Conseguir un *dataset* de imágenes reales suficientemente rico es muy costoso. Una aproximación distinta es hacer uso de imágenes sintéticas creadas a partir de modelos tridimensionales. En el capítulo 3 se describió una aplicación desarrollada en Unity para la generación de dichas imágenes a partir de modelos de objetos cotidianos colocados en posiciones y fondos aleatorios. Con ella, se generaron 1000 imágenes de objetos metálicos a partir de modelos de calidad intermedia.

La cuestión que quedó abierta es si el uso de imágenes sintéticas empeora, y en qué grado, los resultados de una red neuronal entrenada con ellas. El objetivo de este capítulo es analizar esos resultados con *datasets* que mezclen

imágenes reales y sintéticas en diferentes proporciones. Se pretende así encontrar el porcentaje que maximice la cantidad de imágenes sintéticas, que en principio deberían ser más fáciles de conseguir, sin reducir significativamente el rendimiento del resultado.

Esta investigación se ha llevado a cabo con el *dataset* obtenido para el prototipado (explicado en la sección 3.2.2) de la aplicación. Esto significa que sólo uno de los tres materiales (metal) cuenta con imágenes sintéticas y reales mezcladas. En cambio, los dos materiales restantes (vidrio y plástico) están formados por completo por imágenes reales.

Para la realización del experimento se va a entrenar la red un total de once veces, en las que se va a ir aumentando paulatinamente la cantidad de imágenes sintéticas en el metal. El primer entrenamiento se realiza con un *dataset* compuesto íntegramente por imágenes reales. En el siguiente se empiezan a añadir imágenes sintéticas, sustituyendo a las reales, cuyo porcentaje se incrementará en un 10 % por cada entrenamiento. Es decir, el *dataset* contará con 0 % de imágenes sintéticas en el primero entrenamiento, en el segundo tendrá un 10 %, un 20 % en el tercero y así sucesivamente hasta que esté compuesto al 100 % por imágenes sintéticas. El total de las imágenes siempre será el mismo (1000), pero se irán sustituyendo imágenes reales por sintéticas. Para cada uno de los casos se verá su rendimiento y se comparará con el de los demás para seleccionar el más adecuado para su uso.

En aprendizaje máquina supervisado, los *datasets* se dividen en dos partes, una para entrenamiento y otra para *test*, que será lo que indique la exactitud resultante del entrenamiento. Esta separación se puede hacer cargando todo el *dataset* y después dividirlo mediante código, o bien las imágenes se pueden separar en carpetas y cargar cada una como datos de entrenamiento o *test* respectivamente.

En este proyecto es de suma importancia que todas las imágenes de *test* sean reales, ya que el uso del modelo va a ser sobre objetos reales. Por lo tanto, la distinción entre datos de entrenamiento y *test* se hace previamente separando los archivos. Para ello, de las imágenes reales de cada material se seleccionan, de manera aleatoria, aproximadamente un 10 % que serán utilizadas para *test*. Esto deja el reparto de imágenes como se muestra en la tabla 4.1. Eso significa que los porcentajes indicados previamente para la parte de entrenamiento no son para el *dataset* completo, dado que el *test* siempre se llevará a cabo con imágenes reales.

Al tratarse de una red con aprendizaje supervisado los datos deben ir correctamente etiquetados, en este caso esto se realiza mediante la estructura de carpetas (mostrada en la figura 4.1). Se cuenta con dos carpetas principales, una para entrenamiento y otra para *test*. Ambas carpetas contienen tres subcarpetas en su interior, una para cada material identificable (plástico, vidrio y metal), en las que se guardan las imágenes correspondientes. Esta organización de carpetas es de suma importancia ya que las etiquetas para

| Materiales | Imágenes reales entrenamiento | Imágenes reales test |
|------------|-------------------------------|----------------------|
| Plástico | 1958 | 214 |
| Vidrio | 811 | 106 |
| Metal | 1103 | 100 |

Tabla 4.1: Imágenes disponibles por material separadas en entrenamiento y *test*

el modelo se adquieren de estos directorios.

4.2. Entrenamiento

La fase de entrenamiento del proyecto está basada en las recomendaciones para generar modelos de TensorFlow Lite, ya que el modelo que se va a utilizar para la identificación es el de este formato. Para llevar a cabo el entrenamiento de la red neuronal se ha realizado un *script* que se encarga de la lógica de este. Dicho *script* está basado en uno de los ejemplos de Tensorflow Lite y puede usarse desde Colaboratory¹, plataforma de Google que permite programar y ejecutar en Python desde el navegador. El ejemplo elegido como referencia es un programa de identificación de flores². Se ha seleccionado este debido a su similitud con el proyecto respecto a la identificación del objeto principal en una imagen.

Para generar el modelo usando las imágenes obtenidas se utiliza la biblioteca Model Maker de TensorFlow Lite, que permite realizar el entrenamiento utilizando aprendizaje por transferencia. Para ello se ha utilizado el modelo de EfficientNet-Lite, que aún no siendo el más exacto es el más recomendable para dispositivos móviles por su tamaño y latencia [29] permitiendo ser utilizado por un mayor número de dispositivos independientemente de si son de gama baja o alta. El entrenamiento para este proyecto se lleva a cabo desde la computadora, evitando tener que cargar todo el conjunto de imágenes en Colaboratory. Para ello se tiene un *script* que realiza la carga de las imágenes, el entrenamiento, la creación del modelo y las pruebas de rendimiento.

El entrenamiento se divide en varios episodios. Estos corresponden al número de veces que el algoritmo recorre todos los datos. En cada episodio los datos se dividen en lotes, en los que se recoge una pequeña parte del *dataset*. Cada lote se recorre en una iteración, por lo que cada episodio tiene las iteraciones necesarias para recorrer todos los lotes y, por lo tanto, el

¹<https://colab.research.google.com/notebooks/intro.ipynb>

²<https://colab.research.google.com/drive/1sqBewUnvdAT00-yblj55EBFb2sM24XHR?hl=es-419>

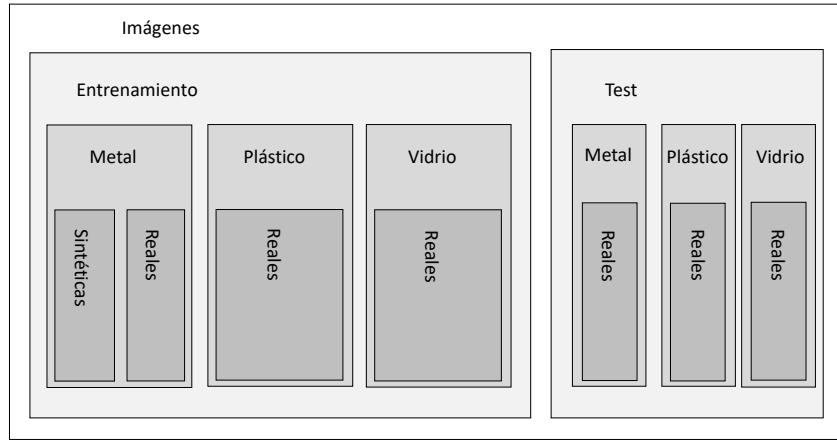


Figura 4.1: Organización de las carpetas con las imágenes separadas.

conjunto de datos completo. Esto significa que el número de iteraciones de cada episodio corresponde al resultado de dividir la cantidad de datos entre el tamaño de lote [56]. Por ejemplo, en los entrenamientos se tiene un total de 3769 imágenes (1000 para metal, 811 para vidrio y 1958 para plástico) y un tamaño de lote de 32, para procesar todos los lotes, en cada episodio se necesitan 117 iteraciones.

Una vez que finaliza el entrenamiento y el *test* de la red, se genera un archivo con extensión *.tflite*, además de un documento de texto plano con las etiquetas de los distintos materiales. Como se ha comentado, este proceso se repite once veces, con *datasets* en los que se combinan distintos porcentajes de imágenes reales y sintéticas. En la sección siguiente se analiza la información recopilada para seleccionar el mejor de los modelos, que será en última instancia utilizado en la aplicación móvil de ayuda al reciclaje descrita en el capítulo 5.

4.3. Resultados

Como último punto, queda determinar con qué porcentaje de imágenes sintéticas se obtiene el mejor resultado.

La figura 4.2 muestra el crecimiento de la exactitud (en inglés *accuracy*)

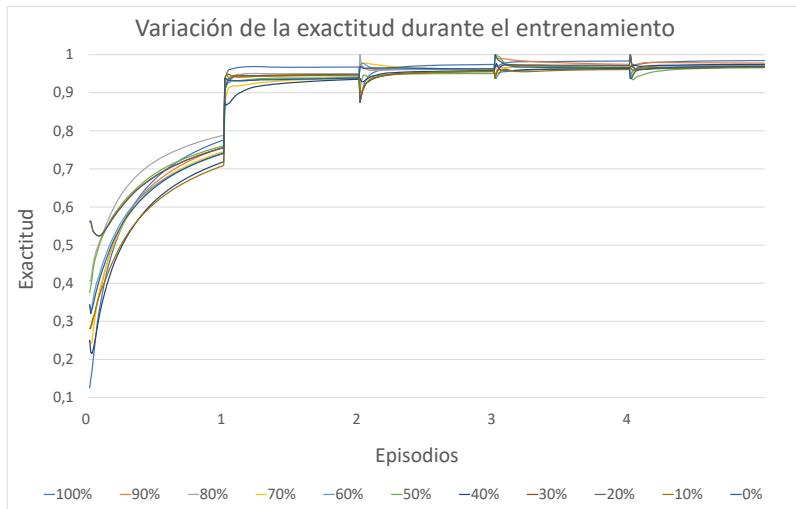


Figura 4.2: Variación de la exactitud de la red neuronal a lo largo del entrenamiento

de la red durante el entrenamiento. La exactitud (o tasa de éxito) es el porcentaje de aciertos de la red respecto a los resultados reales. Se encuentra representado para los distintas combinaciones de imágenes generadas y reales. Al ser la exactitud durante el entrenamiento, esta se calcula usando directamente las mismas imágenes del *dataset* de entrenamiento, no las de *test*. En este proceso se le está enseñando a la red a categorizar las imágenes, así que el valor de la exactitud se obtiene a partir de si clasifica correctamente las mismas imágenes sobre las que se le está entrenando. Que la tasa de éxito crezca de manera similar para todos los *datasets* significa que están aprendiendo bien para las imágenes que se les están proporcionando. Es decir, en el caso extremo donde todas las imágenes de los objetos metálicos son sintéticas, la red las está categorizando correctamente como metal.

Para todos los *datasets*, a partir del segundo episodio, se obtiene una tasa de éxito superior al 90 %. Puesto que dicho valor no llega a alcanzar el 100 %, se considera que no hay sobreajuste. Si fuera así, significaría que los modelos han sido sobreentrenados y por lo tanto conocen el resultado deseado.

En la figura 4.3, por el contrario, se observa cómo varía la exactitud al probarse el modelo con los datos de *test*, aquí se sacan las primeras conclusiones sobre la red. Similar a como se realizó el entrenamiento, para el *test* los datos también se dividen en lotes. En cada iteración la red se prueba

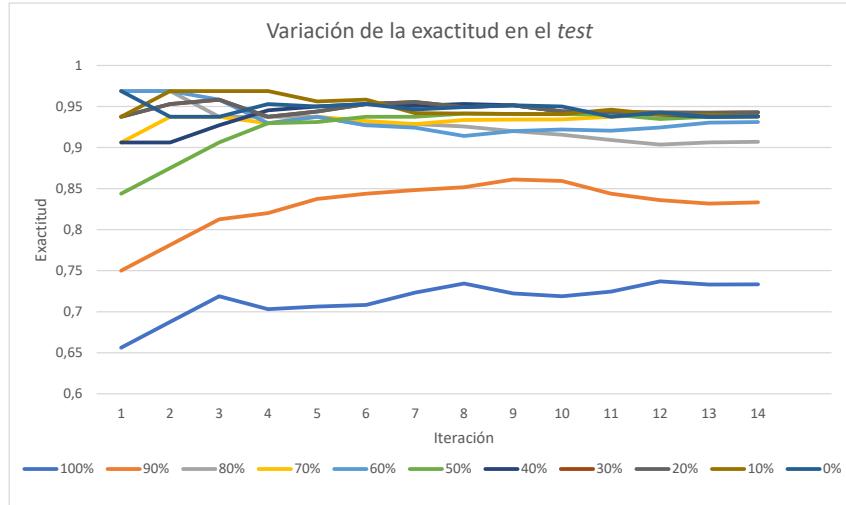


Figura 4.3: Variación de la exactitud de la red neuronal durante el *test*

sobre cada lote de imágenes y la tasa de acierto se va actualizando según los resultados obtenidos. Se observa que la mayoría de las opciones se mantienen casi todo el proceso en una precisión mayor del 90 %. Los casos en los que se puede contemplar una caída de la exactitud, son aquellos que cuentan con mayor porcentaje de imágenes sintéticas para el entrenamiento. Esto se debe a que al haber visto muy pocas imágenes reales de este material durante el entrenamiento ahora en el *test* no las identifica como metal, provocando que la tasa de aciertos decrezca notablemente.

La figura 4.4 muestra el valor final de la exactitud en los distintos casos. A partir de esta última figura pueden sacarse las conclusiones de con qué porcentaje se obtendrían los mejores resultados.

Los dos *datasets* que cuentan con la mayor cantidad de imágenes sintéticas (90 % y 100 %) serían los menos recomendables para utilizar debido a su baja exactitud. En cambio, entre el 0 % y el 70 % de imágenes generadas, se observa que la exactitud siempre está por encima del 90 %, aunque nunca llega a superar el 95 %.

De cara a elegir cuál de los modelos entrenados utilizar en la aplicación para dispositivos móviles, se tiene en cuenta el rendimiento de la red y la facilidad de obtención del *dataset*. Esto se traduce en que se busca un modelo con un alto nivel de exactitud pero que también cuente con un gran número

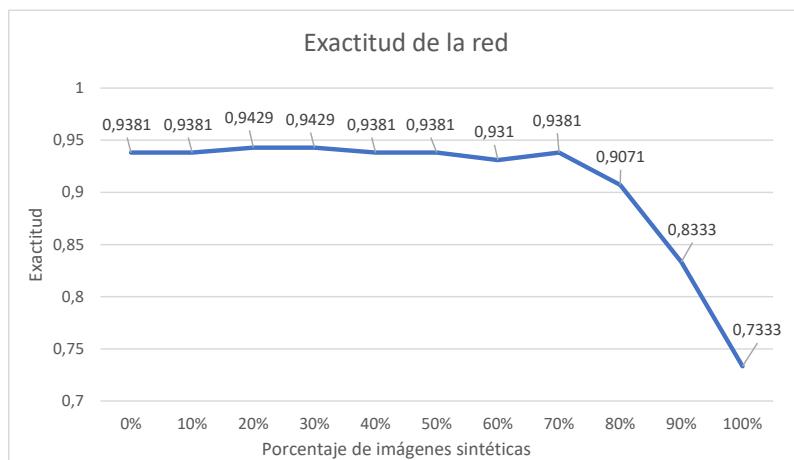


Figura 4.4: Variación de la exactitud final de la red neuronal según la proporción de imágenes reales y sintéticas

de imágenes generadas, ya que estas, con lo desarrollado en el proyecto, se pueden conseguir de manera más fácil que las reales.

Finalmente, se considera como mejor opción el modelo entrenado con un 70 % de imágenes generadas en el material de metal. Esta opción es de las que mayor precisión tienen y a la vez permite tener un *dataset* que funciona adecuadamente formado principalmente por imágenes sintéticas, lo que facilita su obtención.

Capítulo 5

Aplicación de identificación de objetos

RESUMEN: En este capítulo se trata el desarrollo de la aplicación móvil y se estudia su funcionamiento. Esta aplicación tiene como objetivo ayudar a los usuarios a cómo reciclar los objetos a los que se apunte con la cámara. Para eso, tiene que ser capaz de reconocerlos basándose en modelos entrenados previamente. Además, se pone a prueba su funcionamiento y se comparan los resultados con los de distintos modelos. La aplicación es para dispositivos con sistema operativo Android y está llevada a cabo en Android Studio con Java y la librería de TensorFlow Lite para introducir la identificación.

5.1. Introducción

Como se ha comentado, la aplicación de identificación de objetos para reciclaje es el objetivo principal de este Trabajo de Fin de Grado. Se trata de una aplicación para Android que, utilizando la cámara del dispositivo en el que está instalada, identifica el material del objeto al que está enfocando el usuario. En la interfaz aparecen las tres opciones con más probabilidad, acompañadas del porcentaje de confianza respecto a esa opción. Independientemente de cuántas etiquetas de materiales distintas haya, siempre se muestran las tres con mayor confianza. La aplicación está constantemente interpretando lo que recibe desde la cámara, así que las etiquetas y sus respectivos porcentajes se actualizan ininterrumpidamente sin necesidad de estar tomando fotografías para cada elemento.

5.2. Identificación de objetos

Para realizar la identificación se ha incorporado TensorFlow Lite y sus librerías al proyecto en Android Studio. Esto permite importar un intérprete, el cual carga el modelo seleccionado (el compuesto por un 70 % de imágenes sintéticas para el metal) y permite ejecutarlo ofreciéndole una serie de datos de entrada, finalmente, tras ejecutarlo, se muestran por pantalla los resultados obtenidos. El trabajo se ha basado en las guías y recomendaciones que ofrece TensorFlow Lite en sus ejemplos como ayuda para incorporar y utilizar el modelo de la red neuronal entrenado previamente (capítulo 4).

Durante el flujo de la aplicación se hace uso de la cámara del dispositivo. Los *frames* desde esta son los datos que se utilizan como entrada para la identificación. Dichos datos se reciben en formato *bitmap*, matrices donde cada casilla tiene asignado un color y que en conjunto forman una imagen. Ese *bitmap* es convertido a *byte buffer*, denominado *IMG data*, haciéndolo legible para la identificación. Además, utilizar *byte buffers* como entrada permite que la API de Java sea más rápida¹. Estos datos se cargan en el intérprete de TensorFlow Lite y finalmente se obtienen los valores de resultado. Estos valores obtenidos son índices respecto a las etiquetas acompañados de la probabilidad de que esa imagen corresponda a dicha etiqueta. Este resultado se devuelve en un array, correspondiendo cada posición a cada una de las etiquetas disponibles. Finalmente, se seleccionan las tres etiquetas con mayor probabilidad y son mostradas por la interfaz.

La interfaz de la aplicación se divide en dos zonas. En la figura 5.1 se muestra la aplicación en funcionamiento, donde pueden apreciarse las distintas características de esta. En la parte central se refleja todo aquello que se recibe a través la cámara del dispositivo y en la parte inferior se sitúa un panel en el que aparecen los resultados de la identificación. Estos se muestran en orden según de qué material se considera que se trata acompañado del contenedor en el que se debe desechar y el porcentaje de confianza de cada elemento.

Como se comentó en el capítulo 3, al hablar de la obtención del *dataset* utilizado para el entrenamiento, para el prototipado se ha trabajado con tres materiales (metal, vidrio y plástico). En la aplicación se muestran las tres posibilidades de identificación sobre las que se tiene más confianza, lo que significa que en este caso se presentan todas las opciones disponibles. Esto es independiente de la cantidad de materiales disponibles identificables. Es decir, si como trabajo futuro la cantidad de materiales se ampliara, y no se alterara esta funcionalidad, se continuarían mostrando las tres opciones con mayor probabilidad.

El panel de los resultados puede desplegarse hacia arriba, de esta forma se visualizan varias características como la resolución con la que se recogen

¹https://www.tensorflow.org/lite/performance/best_practices

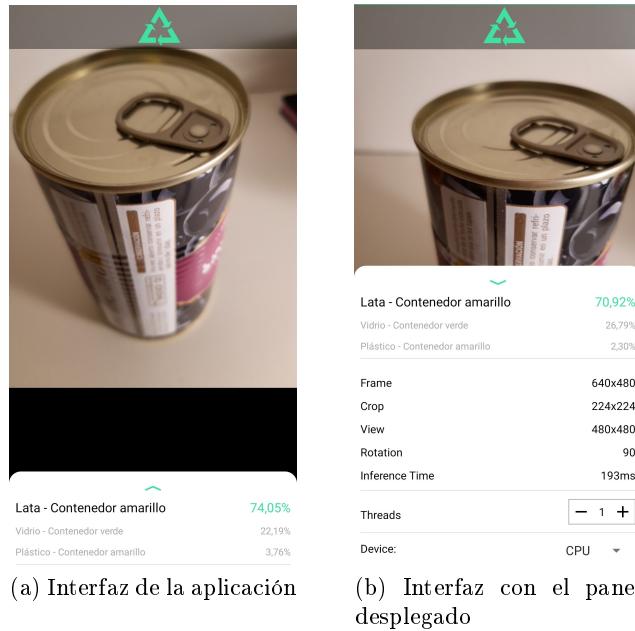


Figura 5.1: Interfaz de la aplicación de identificación

las imágenes, la rotación del dispositivo y el tiempo de inferencia, que es el tiempo que tarda en detectar de qué objeto se trata. Además de estas características, cuenta con dos opciones configurables por el usuario que permiten la mejora de la aplicación. La primera es la elección de cuántos hilos utiliza la aplicación, lo que permite acelerar la ejecución de los operadores. Esta aceleración tiene un coste; el aumento de los hilos utilizados hará que se necesiten más recursos y batería. La otra opción es elegir entre utilizar la CPU o la GPU. La GPU es uno de los aceleradores que TensorFlow Lite puede aprovechar. En dispositivos de gama media o alta la GPU es más rápida que la CPU, lo que reduce notablemente el tiempo de inferencia.

5.3. Pruebas

Como ya se ha comentado, a la vista de los resultados descritos en el apartado 4.3, se decidió que para la aplicación móvil se usaría el modelo para cuyo entrenamiento se habían utilizado un 70 % de imágenes generadas para la categoría “metal”, el cual con un 93 % de exactitud. No obstante, se ha querido corroborar su correcto funcionamiento una vez introducido el modelo en la aplicación y utilizándolo sobre objetos nuevos del mundo real. Para ello se ha comparado el rendimiento de este modelo con el de otros tres de los entrenados. Se han seleccionado, como pueden verse resaltados en la figura 5.2, los dos modelos extremos, correspondiendo a los compuestos por

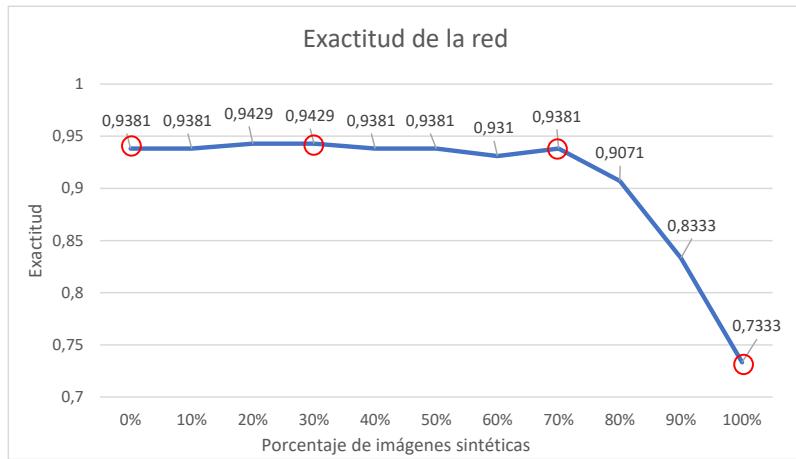


Figura 5.2: Variación de la precisión de la red neuronal con los modelos seleccionados destacados

completo por imágenes sintéticas y por imágenes reales, respectivamente; el modelo con el mejor resultado respecto a precisión y facilidad de obtención del *dataset*, con el 70 % de imágenes generadas; y, por último, el simétrico del anterior, con el 30 % de imágenes sintéticas, el cual resultó ser uno de los dos modelos con mayor exactitud en el *test*.

Para las comparaciones se han cogido diversos objetos y se ha estudiado qué etiqueta cuenta con el mayor valor de confianza para cada uno. También se tiene en cuenta aquellos casos en que las dos primeras etiquetas tienen porcentajes muy similares y por lo tanto van cambiando entre primera y segunda posición. Puesto que la aplicación para el usuario final (sección 5.2) tan sólo utiliza uno de los modelos, se ha desarrollado una versión alterada para la realización de las pruebas. En esta segunda versión, denominada aplicación de *test*, se incorporan los cuatro modelos simultáneamente y para cada objeto identificable se muestran los resultados de todos ellos. Al contrario que con la aplicación original, en esta no se va cambiando la posición del material con mayor confianza, sino que el orden se mantiene estable, generando una tabla en la que puede apreciarse la oscilación de los porcentajes con cada objeto detectado. En dicha tabla las filas indican los tres tipos de materiales. En cambio, cada columna se corresponde con cada uno de los modelos. Estos se identifican indicando los porcentajes de imágenes reales y

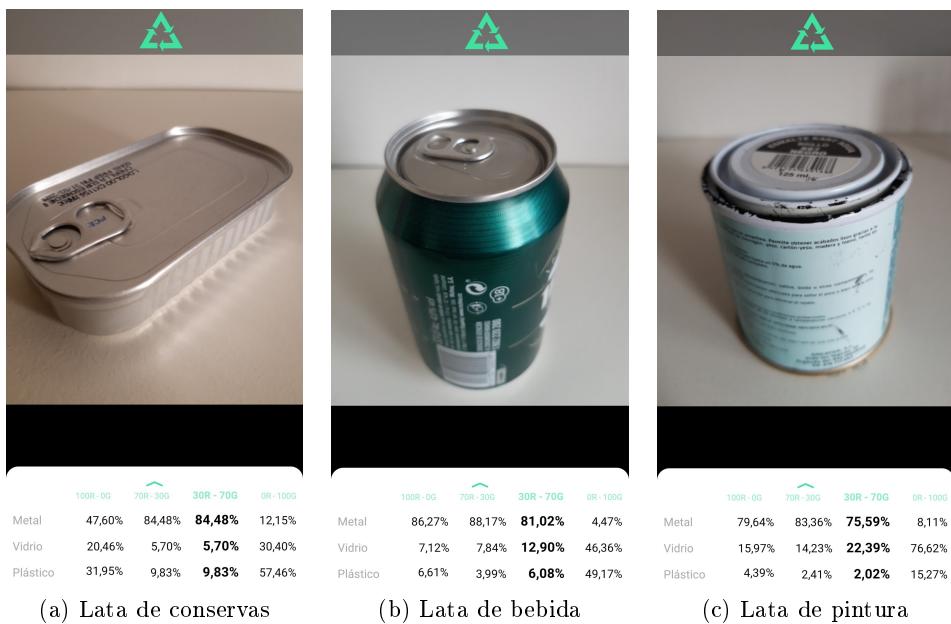


Figura 5.3: Comparación con distintos objetos metálicos

generadas usadas en su entrenamiento, como puede verse en la figura 5.3.

Como se trata de casos de prueba, se prescinde de la información sobre en qué contenedor debe reciclarse cada objeto, dejando así más espacio para los datos de los modelos. Además, para mayor claridad durante el uso de la aplicación de *test*, la columna correspondiente al modelo seleccionado para la aplicación final se le da cierto énfasis para distinguirla del resto.

El objeto de estudio principal han sido los objetos metálicos, ya que son aquellos que han sido entrenados con imágenes generadas. Sin embargo, también se ha probado sobre objetos de los otros materiales para comprobar el rendimiento respecto a estos.

En la figura 5.3 se pueden observar los resultados respecto a distintos objetos metálicos. Casi todos los modelos detectan con un nivel alto de confianza que en los tres casos se trata de objetos metálicos. La única excepción es el del modelo entrenado por completo con imágenes sintéticas, el cual no da una respuesta determinante entre vidrio y plástico.

Por otro lado, los ejemplos de vidrio resultan tener resultados sumamente positivos, los cuales se muestran en la figura 5.4. Destaca frente al resto de objetos la identificación de una botella de vino, donde todos los modelos, sin excepción, la identifican correctamente con más de un 95 % de seguridad. En cambio con la taza de cristal los porcentajes disminuyen considerablemente, este resultado es esperado debido a lo complicado que puede resultar en ocasiones distinguir objetos de vidrio y plástico si son translúcidos.

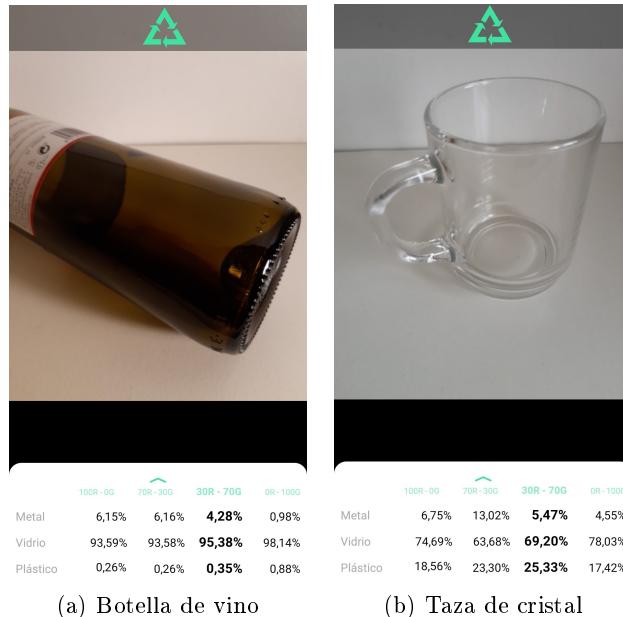


Figura 5.4: Comparación con distintos objetos de vidrio

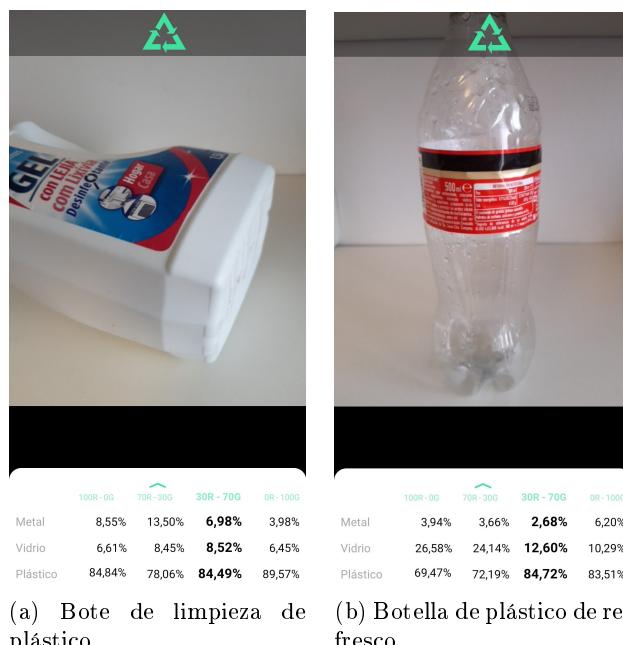


Figura 5.5: Comparación con distintos objetos de plástico

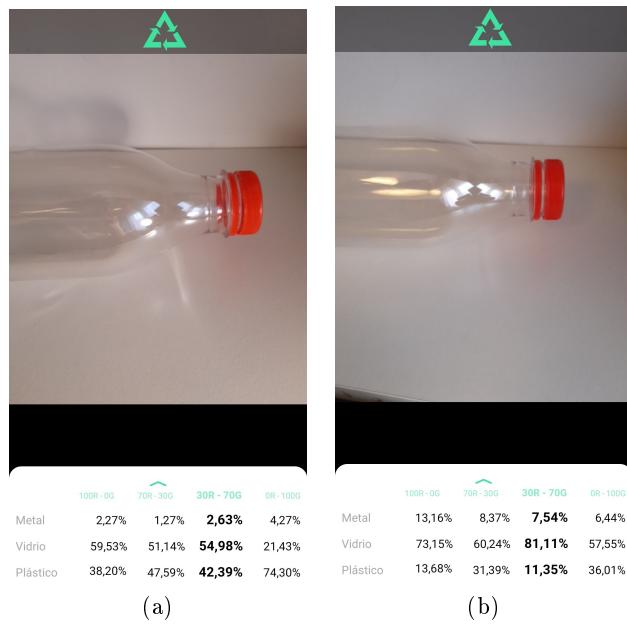


Figura 5.6: Comparación de la misma botella de plástico con segundos de diferencia

Por último como pruebas de materiales de plástico se han tomado tres objetos. Dos de ellos (figura 5.5) se aprecia el correcto funcionamiento detectando adecuadamente el material. En cambio, en el tercero (figura 5.6), los porcentajes cambian drásticamente constantemente alternando entre vidrio y plástico, como puede verse en la figura. Esto se achaca a lo mencionado anteriormente de la dificultad de diferenciar el material en objetos translúcidos.

Con todos los resultados obtenidos de las pruebas de la red neuronal (capítulo 4) y los realizados con objetos reales, se puede afirmar que la aplicación funciona adecuadamente. No obstante, aunque no se obtienen unos porcentajes de rendimiento como los de la sección 4.3, se considera que los resultados han sido positivos.

Por último, el tiempo de inferencia de la aplicación es de unos 200ms, aproximadamente 700ms en la de *test* (al tener que ejecutar cuatro modelos), utilizando la CPU del dispositivo. En cambio, si se realiza con la GPU estos valores se reducen a alrededor de la mitad.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Recapitulación

El resultado del proyecto es un prototipo de una aplicación de ayuda al reciclaje para dispositivos con sistema operativo Android. Muestra aquello a lo que se apunta con la cámara del dispositivo e identifica y ofrece información sobre el material que compone el objeto además de indicar la manera adecuada de desecharlo. Como se ha comentado a lo largo del documento, para llevarla a cabo es necesario utilizar técnicas de visión artificial, las que actualmente conllevan la generación de un modelo entrenado por una red neuronal. Esto supone la necesidad de establecer varios subobjetivos.

El primer subobjetivo que se planteó es la obtención de un *dataset* amplio, claro y variado con imágenes de todos los materiales que se vayan a tener disponibles. Buscando facilitar la obtención de *datasets*, se ha desarrollado una aplicación de generación de imágenes sintéticas a partir de modelos tridimensionales. Esta está generada en la plataforma Unity, motor de videojuegos multiplataforma creado por Unity Technologies. El funcionamiento de esta aplicación consiste en ir cargando los distintos modelos 3D disponibles en los recursos, separados por material, y realizar numerosas capturas a cada uno hasta que se recorren todos. En cada captura tomada tanto la posición y rotación del objeto, como el fondo de la imagen, se establecen de manera aleatoria, proceso necesario para contar con una alta diversidad en las imágenes. Debido a los inconvenientes surgidos durante la obtención de los modelos, finalmente se obtuvo un *dataset* compuesto por tres materiales diferentes (metal, vidrio y plástico), donde para metal se mezclan imágenes reales con sintéticas.

Una vez generado el *dataset* se dio paso al segundo propósito para realizar la aplicación. Este se divide en dos pasos diferentes. El primero, y primordial, es el entrenamiento de la red neuronal y la generación del modelo entrenado utilizando las imágenes generadas previamente. El resultado del entrenamiento es incorporado posteriormente en Android Studio para su uso en la

aplicación final. Para este apartado se ha utilizado una librería de TensorFlow Lite que permite el entrenamiento por transferencia utilizando la red EfficientNet y proporciona herramientas para llevar a cabo el entrenamiento de manera fácil e intuitiva.

Como segundo paso, una vez finalizado el entrenamiento de la red neuronal, se realizan diversas pruebas y comparaciones con la finalidad de encontrar la proporción entre imágenes sintéticas y reales que ofrecen el mejor equilibrio entre facilidad de obtención del *dataset* y exactitud del modelo entrenado. La conclusión de las pruebas fue que a partir del 30 % de imágenes reales la precisión sufre muy pocos cambios y se mantiene siempre por encima del 90 % de confianza. Es decir, la precisión del modelo cuando todas las imágenes son reales y cuando se utiliza en uno de los materiales sólo un 30 % de estas, es muy similar. De esta forma, se tomó la decisión de utilizar el modelo con el 70 % de imágenes generadas y el 30 % restante reales para el metal, que cuenta con un 93 % de confianza.

Finalmente, tiene lugar el desarrollo de la aplicación para dispositivos Android mediante la herramienta Andorid Studio y utilizando TensorFlow Lite. La aplicación, a través de la cámara, recibe imágenes de los *frames* capturados y consultando el modelo importado trata de identificar el objeto que aparece en la imagen. Devuelve como resultado las etiquetas disponibles ordenadas según el porcentaje de seguridad con el que considera que se trata de ese objeto, informando al usuario del material, el contenedor donde desecharlo y la confianza con la que considera que se trata de dicho material.

Para probar su funcionamiento se hicieron ensayos sobre objetos del mundo real. Para ello, se realizaron cambios en la aplicación dando lugar a otra de *test*. A través de ella se compara el funcionamiento de varios modelos simultáneamente. Con los datos obtenidos, se corroboró el correcto funcionamiento del modelo entrenado con el *dataset* seleccionado.

6.2. Conclusiones

En el proceso se han ido observando diversas dificultades y conclusiones importantes que han afectado al desarrollo del trabajo y que se deben tener en cuenta en posibles ampliaciones futuras o el desarrollo de otros proyectos similares o relacionados.

Se ha observado la dificultad real de obtener *datasets* amplios sobre objetos concretos si no se cuenta con muchos recursos, como se ha experimentado en el desarrollo del proyecto, y lo que ha provocado limitarse a contar con solamente tres materiales. Una solución a este problema puede ser el uso de imágenes sintéticas para completarlos. Este trabajo ha demostrado que, al menos en el contexto planteado, esta opción tiene resultados altamente positivos, ya que poblando una parte significativa del *dataset* con imágenes sintéticas, la precisión apenas se ve afectada. Tras la realización de pruebas

en el entorno real, se corrobora el correcto funcionamiento de la aplicación obteniendo resultados acertados en la identificación y con confianza considerablemente positiva. Por este motivo, se considera que la aplicación de generación de imágenes es un aporte útil para muchos proyectos que se salen del estándar en el ámbito del reconocimiento de objetos e imágenes.

No obstante, para que esto se cumpla, conseguir imágenes sintéticas tiene que resultar más sencillo que conseguir fotografías, ya que sigue existiendo cierta dificultad para generar el *dataset*. Esto es provocado por la obtención de modelos tridimensionales, los cuales es importante que cuenten con un alto nivel de realismo en las texturas y materiales. Esto se debe a que la manera en que la luz incide y se refleja sobre ellos puede resultar poco realista y generar problemas en el entrenamiento de la red neuronal, haciendo que el problema de la generación de *datasets* no quede solventado en su totalidad.

La iluminación es otro punto decisivo, a pesar de las facilidades que ofrece Unity para crear distintos tipos de iluminación, si no se tiene suficientes conocimientos y experiencia, el resultado puede terminar siendo pobre y, de nuevo, poco realista. Esto es un problema ya que la aplicación va a utilizarse sobre objetos reales y si no hay un equilibrio entre lo que se utiliza para el entrenamiento y el uso final, la precisión de la aplicación queda notablemente disminuida. Esto pudo observarse en los entrenamientos en los que el *dataset* contaba con más de un 80 % de imágenes sintéticas, al ocupar estas la mayor parte del entrenamiento y diferir cómo se ve en la realidad, los resultados fueron peores en comparación con el resto de proporciones.

En vista de los resultados obtenidos durante el trabajo, y con el apoyo de otros proyectos donde se ha trabajado con *datasets* sintéticos, se puede afirmar que si se va a utilizar un *dataset* compuesto por imágenes sintéticas es necesario que estas se encuentren entremezcladas con reales para un correcto funcionamiento. Esto se debe a que las cámaras virtuales y las reales son sensores diferentes.

Por último, una última observación a tener en cuenta, es que al llevar a cabo el entrenamiento respecto a tres materiales solamente, y donde las imágenes de sólo uno de ellos se mezclan con sintéticas, se observa que esto afecta no solamente al material que tiene las imágenes generadas, sino que también influye a los demás, generando errores en la identificación de objetos compuestos de materiales sobre los que se ha entrenado por completo con imágenes reales.

6.3. Trabajo futuro

El proyecto cuenta con diversos puntos ampliables. El primero es la obtención de un *dataset* más amplio, añadiendo más variedad de materiales y objetos identificables. Esto puede hacerse a partir de imágenes reales únicamente, o bien mezclándolas con imágenes sintéticas. Para esta segunda

opción, es necesario conseguir o generar modelos tridimensionales de todos los materiales y objetos que se quieran incorporar, los cuales deben tener una calidad bastante elevada. Otro factor ampliable, para permitir mayor diversidad en las imágenes generadas, es la extensión del número de imágenes disponibles para el fondo.

Con el objetivo de generar imágenes más realistas, sería necesario revisar la iluminación presente en la escena en la que tiene lugar las capturas, acercándolo a cómo se ven los objetos posteriormente en un entorno real. Asimismo, es necesario adquirir un *dataset* de imágenes reales de los materiales que se quieran agregar. La cantidad de imágenes de este estaría regido por si se va combinarse con imágenes sintéticas o no.

Una vez obtenido el *dataset*, simplemente es necesario entrenar y generar el modelo para, finalmente, importarlo en la aplicación mediante Android Studio. Todos los materiales añadidos además deben relacionarse con el contenedor o lugar de desecho apropiado, que es la información que busca el usuario.

Para mejorar la red y ampliar el *dataset*, podría desarrollarse la opción de que los usuarios puedan enviar sus propias fotografías etiquetadas con el material del que se trate, dando lugar a un *dataset* más completo que se encuentre en continuo crecimiento ofreciendo así un mejor servicio a los usuarios.

Otra mejora adicional interesante, es convertir la generación de imágenes en una aplicación ejecutable independiente de Unity, en la que los usuarios puedan generar *datasets* para sus proyectos a partir de modelos propios tuyos que importen en esta, o aprovechar algunos de los que se ofrezcan por defecto, por ejemplo los utilizados en este trabajo. Aunque la importación de modelos en tiempo de ejecución es algo poco extendido en el mundo de los videojuegos, debido a la cantidad y diversidad de usuarios con los que cuenta Unity, esta funcionalidad ha sido ampliamente discutida y explorada. Pueden incluso encontrarse paquetes para llevar esto a cabo, por ejemplo “TriLib 2 - Model Loading Package” de Ricardo Reis¹, “Runtime OBJ Importer” de Dummiesman² u “OBJReader” de Starscene Software³, así como numerosas discusiones en foros sobre el tema.

Respecto a la aplicación de Android también existen varias posibilidades de trabajo futuro. Una de ellas es llevar a cabo una interfaz más personalizada que pueda ofrecer más información sobre los distintos cubos disponibles o el proceso de reciclaje. Otra, es desarrollar la misma aplicación para dispositivos iOS, permitiendo así su acceso a un mayor número de usuarios. Además, una ampliación necesaria de la aplicación, es la incorporación de

¹<https://assetstore.unity.com/packages/tools/modeling/trilib-2-model-loading-package-157548>

²<https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547>

³<https://starscenesoftware.com/objreader.html#ObjReader>

accesibilidad con el objetivo de permitir su uso a todo tipo de público. Dichas mejoras corresponden, entre otras, a la introducción de una opción de voz, personalización del tamaño de la fuente o de los colores y contrastes.

Conclusions and future work

Summary

The result of the project is the prototype of a recycling-aimed application for Android devices. It shows the object that is pointed at with the device's camera, identifying and offering information about its material, as well as indicating the proper way to dispose it. As it has been commented throughout the document, it is necessary to use artificial vision techniques, which currently involve the generation of a model trained by a neural network. This implies the need to establish several subgoals.

The first sub-objective proposed was to obtain a wide, clear and varied dataset made of images of all the materials that will be available. In order to facilitate the obtaining of these datasets, an application to generate synthetic images from three-dimensional models has been developed. To do so, Unity, has been used. The application flow consists of loading the diverse 3D models available, sorted by material in the resources, and then taking numerous captures of each one. In each capture, both the position and rotation of the object, as well as the background, are established randomly. This is required as a way to compile high diversity in the images. Due to the inconveniences that arose during the obtaining of the models, the final dataset was composed of three different materials (metal, glass and plastic), in which real images are mixed with synthetic images for metal material.

Once the dataset was generated, it was proceed to the second subgoal. This is divided into two different steps. The first one, is the neural network training and the trained model generation using the previously obtained dataset. The result of the training will be later incorporated into Android Studio to be used in the final mobile application. For this section, a TensorFlow Lite library has been used, which allows transfer training using the EfficientNet network and provides tools to perform the model training easily and intuitively.

The second step, consisted of carrying out several tests and comparisons in order to find the ratio between synthetic and real images which offered the best balance between the ease of obtaining the dataset and the accuracy of the trained model.

The conclusion of the tests said that from 30 % onward of real images, the precision barely changes staying above 90 % confidence. Which means that the accuracy of the model when all the images are real and when only 30 % of these are used in one of the materials is quite similar. With these results it was decided to use the model in which metal material data was composed of 70 % generated images and 30 % real images with a 93 % of accuracy.

Lastly, the development of the Android application took place, using the Andorid Studio tool and TensorFlow Lite's libraries. The application receives images of the frames captured by the camera and, consulting the imported model, tries to identify the object that appears in the image. As a result, it sorts the available labels depending on the confidence for each material. This way it is informing the user about the material of the specified object, the container where to dispose it into and the confidence of the identified materials.

In order to test its performance, several tests were carried out on real world objects. To do so, some changes were made to the application leading the creation of a second one, considered as the test application. Using this application, the performance of four models is compared simultaneously. With the data obtained from this tests, the correct performance of the trained model seleccted was corroborated.

Conclusions

During the development, various difficulties and conclusions that affected it were observed, which should be taken into account in possible future extensions or in the development of similar and related projects.

It was a real problem to find large datasets based on some of these specific objects, as not many resources of this kind are available. This has caused limitations, as just having three materials. A solution to this problem can be the use of synthetic images to complete the dataset. The work has shown that, at least in the proposed context, this option has highly positive results, because even though an important part of the dataset was formed by synthetic images, the accuracy was barely affected. After conducting tests in a real environment, the correct performance of the application has been corroborated as a result of the accurate identification with positive confidence obtained. For this reason, the image generator application is considered a useful support to many out-of-the-box projects in the field of image and object recognition.

However, in order for this application to be used, it is necessary that getting synthetic images is easier than getting real photographs, since there is still some difficulty generating a proper dataset. This is mainly caused by the trouble obtaining three-dimensional models with high level of realism in textures and materials. This is important because, otherwise, the way the

light falls and reflects on them can be unrealistic and cause problems in the training of the neural network, which makes the problem of generating datasets not entirely solved.

Lighting is another important element. Despite the facilities that Unity offers to create different types of lighting, without enough knowledge and experience the results can end up being poor and unrealistic. This is a problem since the application is going to be used on real objects, and if there is no balance between that and what is used for training, the precision of the application would noticeably decrease. This could be observed when the dataset had more than 80 % of synthetic images, where the training images differed from the real object. The results were worse in comparison with the rest of the percentages.

With the results obtained during the coursework and the outcome of other synthetic datasets-based projects, it can be stated that the correct performance of a synthetic images dataset depends whether it has been mixed with real images. This happens because virtual and real cameras are different sensors.

Lastly, it must be taken in consideration that the performance mixing real and synthetic images on just one material also influence the outcome of the others which only contained real images in the dataset.

Future work

The project has several scalable features. The first one is obtaining a larger dataset, which would add more variety of materials and objects to identify. This can be done from real images by themselves, or mixing them with synthetic images. In the case of the second option it is necessary to obtain or generate high quality three-dimensional models of all the materials and objects required. Another expandable characteristic to allow greater diversity of the generated images is to extent the number of images available for the background.

So as alike to how the objects are later seen in a real environment. Also, it would be necessary to acquire a real image dataset of the materials added. The amount of images this would require depends on whether it is needed to be combined with synthetic images or not.

Once the dataset has been obtained, just training and generating the model is left to finally import it into the application using Android Studio. All added materials must also be related to its appropriate container or disposal site, which is the information the user will looking for.

To improve the network and expand the dataset, it could be developed a feature where users can send their own tagged photographs, allowing the continuous growth of the application and a better service for the users.

Another additional and interesting improvement would be to convert

the images generator into an executable application independent of Unity, in which users can generate datasets for their projects using their own models imported into it, or use some offered by default, like, for instance, those used in this coursework. Although the runtime import of models is not very widespread in video games, due to the number and diversity of users that Unity has, this functionality has been widely discussed and explored. Packages of this kind can be found to do so, for example “ TriLib 2 - Model Loading Package ” by Ricardo Reis footnote url <https://assetstore.unity.com/packages/tools/modeling/trilib-2-model-loading-package-157548>, “ Runtime OBJ Importer ” from Dummiesman footnote url <https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547> u “ OBJReader ” from Starscene Software footnote url <https://starscenesoftware.com/objreader.html#ObjReader>, as well as numerous forum discussions on the subject.

The Android application itself has also several properties expandable in future work. One of them is to personalized more the user interface so it can offer information about the different containers available or the recycling process. Another option is to develop the application for iOS devices, allowing it to be used by more users. In addition, a necessary extension of the application is the incorporation of accessibility in order to allow its use to all types of users. This improvements correspond, among others, to the introduction of a voice option, customization of the font size, colors and contrasts.

Apéndice A

Materiales

A.1. Materiales adjuntos

Con la memoria se han entregado de manera adjunta en la carpeta de Google Drive dos archivos con extensión *.apk* que corresponden a las dos aplicaciones móviles desarrolladas en este Trabajo de Fin de Grado. Puede realizarse la instalación de ambas para probar su funcionamiento sobre objetos reales.

El resto de materiales desarrollados pueden encontrarse en el repositorio de GitHub <https://github.com/celica02/RecyclingFinalDegreeProject>. Tanto el código de las aplicaciones como los modelos ya entrenados y las imágenes utilizadas para ello se encuentran así almacenadas en sus correspondientes carpetas. Toda la información del repositorio puede encontrarse en el archivo **Readme** de este mismo.

Bibliografía

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: large-scale machine learning on heterogeneous systems, 2015. *Software* disponible en <https://www.tensorflow.org/>.
- [2] C. Alfonso, R. Estévez Estévez, J. M. Lobo, B. Lozano Diéguez, F. Prieto, J. Santamarta, and A. Gaerter. Emergencia climática en España. Diciembre 2016. Disponible en: <https://www.observatoriosostenibilidad.com/2019/11/29/emergencia-climatica-en-espana/>.
- [3] R. Almond, G. M., and T. Petersen, editors. *WWF (2020) Living planet report 2020 - Bending the curve of biodiversity loss.* WWF, Gland, Switzerland, 2020. ISBN 9782940529995.
- [4] Y. Amit. *2D Object Detection and Recognition: Models, Algorithms, and Networks.* Mit Press. MIT Press, 2002. ISBN 9780262011945.
- [5] Y. Amit, P. Felzenszwalb, and R. Girshick. *Object Detection.* Springer International Publishing, Cham, 2020. ISBN 9783030032432.
- [6] X. Basogain Olabe. Redes Neuronales Artificiales y sus Aplicaciones. 2008. Dpto. Ingeniería de Sistemas y Automática, Escuela Superior de Ingeniería Bilbao.
- [7] M. Caballero, S. Lozano, and B. Ortega. Efecto invernadero, calentamiento global y cambio climático: una perspectiva desde las ciencias de la tierra. *Revista digital universitaria*, 8, 2007. ISSN 1067-6079.
- [8] J. Cohen, C. F. Crispim-Junior, C. Grange-Faivre, and L. Tougne. CAD-based Learning for Egocentric Object Detection in Industrial Context.

- In *15th International Conference on Computer Vision Theory and Applications*, volume 5, Valletta, Malta, 2020. SCITEPRESS - Science and Technology Publications. doi 10.5220/0008975506440651.
- [9] G. Cortina Fernández. Técnicas Inteligentes para su Integración en un Vehículo Autómata. Universidad Complutense de Madrid, 2020.
 - [10] B. Cyganek. *Object detection and recognition in digital images: theory and practice*. John Wiley and Sons, Incorporated, 2013. ISBN 9781118618363.
 - [11] R. Flórez López, J. M. Fernández, and J. M. Fernández Fernández. *Las redes neuronales artificiales*. Metodología y análisis de datos en ciencias sociales. Netbiblo, 2008. ISBN 9788497452465.
 - [12] R. Fonfría, R. Sans, and J. de Pablo Ribas. *Ingeniería ambiental: contaminación y tratamientos*. Colección productiva. Marcombo, 1989. ISBN 9788426707420.
 - [13] S. Frintrop. *VOCUS: A visual attention system for object detection and goal-directed search*, volume 3899 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2006. ISBN 9783540327592.
 - [14] L. García Rodríguez. *Algunas cuestiones notables sobre el modelo de Hopfield en optimización*. PhD thesis, Madrid, Noviembre 2018. Universidad Complutense de Madrid.
 - [15] G. A. Gómez Rojas, J. C. Henao López, and H. Salazar Isaza. Entrenamiento de una red neuronal artificial usando el algoritmo simulated annealing. *Scientia Et Technica*, 1(4), 2004. ISSN 0122-1701.
 - [16] P. González López and J. García-Consuegra Bleda. *Informática gráfica*, volume 19. Universidad de Castilla La Mancha, 1999. ISBN 9788489958234.
 - [17] G. Guridi Mateos. Modelos de redes neuronales recurrentes en clasificación de patentes. B.S. thesis, 2017. Universidad Autónoma de Madrid.
 - [18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.
 - [19] J. R. Hilera and V. J. Martínez Hernando. *Redes neuronales artificiales: fundamentos, modelos y aplicaciones*. RA-MA S.A. Editorial y Publicaciones, 01 1995. ISBN 9788478971558.
 - [20] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. ISSN 0027-8424.

- [21] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.
- [22] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016.
- [23] A. Igúarán Guerra, S. Gómez Ruiz, et al. Análisis de las necesidades y dificultades en la disposición de residuos sólidos en la fuente doméstica para el desarrollo de un producto. B.S. thesis, Universidad EAFIT, 2010.
- [24] Y. C. Jhang, A. Palmar, B. Li, S. Dhakad, S. K. Vishwakarma, J. Hoggins, A. Crespi, C. Kerr, S. Chockalingam, C. Romero, A. Thaman, and S. Ganguly. Training a performant object detection ML model on synthetic data using Unity Perception tools, 2020.
- [25] R. Karim. *TensorFlow: Powerful Predictive Analytics with TensorFlow*. Packt Publishing, Birmingham, 3 edition, Marzo 2018. ISBN 9781789136913.
- [26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. Disponible en <https://kr.nvidia.com/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf>.
- [27] P. Larrañaga, I. Inza, and A. Moujahid. Tema 8. redes neuronales. *Redes Neuronales, U. del P. Vasco*, 12, 1997.
- [28] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015.
- [29] R. Liu. Higher accuracy on vision models with EfficientNet-Lite. *TensorFlow Blog*. Disponible en: <https://acortar.link/nvDwl> , 2020.
- [30] M. A. López Pacheco. Identificación de sistemas no lineales con redes neuronales convolucionales. 2017.
- [31] D. J. Matich. Redes neuronales: Conceptos básicos y aplicaciones. 41, 2001. Universidad Tecnológica Nacional, México.
- [32] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 1943.

- [33] A. J. McMichael, D. Campbell-Lendrum, S. Kovats, S. Edwards, P. Wilkinson, T. Wilson, R. Nicholls, S. Hales, F. Tanser, D. L. Sueur, M. Schlesinger, and N. Andronova. Global climate change (chapter 20), 2003. Disponible en <https://www.who.int/docs/default-source/climate-change/publication---global-climate-change-comparative-analysis.pdf>.
- [34] M. Minsky and S. A. Papert. *Perceptrons: An introduction to computational geometry*. The M.I.T. press, 1987. ISBN 9780262631112.
- [35] S. Morant Gálvez. Desarrollo de un sistema de bajo coste para el análisis de tráfico mediante el uso de deep learning. Universidad de Valencia, 2021.
- [36] L. Moreno Díaz-Alejo. Análisis comparativo de arquitecturas de redes neuronales para la clasificación de imágenes. Master's thesis, 2020. Universidad Internacional de La Rioja.
- [37] B. Müller, J. Reinhardt, and M. Strickland. *Neural Networks: An Introduction*. Physics of Neural Networks. Springer Berlin Heidelberg, 1995. ISBN 9783540602071.
- [38] C. Parra Ramos and D. Regajo Rodríguez. Reconocimiento automático de matrículas. 2006.
- [39] R. Pavón Benítez. Técnicas de deep learning para el reconocimiento de movimientos corporales. Universidad Complutense de Madrid, 2020.
- [40] A. Polacco and K. Backes. The amazon go concept: Implications, applications, and sustainability. *Journal of Business and Management*, 24(1), 2018.
- [41] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016. IEEE. ISSN 1063-6919 doi 10.1109/CVPR.2016.352.
- [42] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.
- [43] A. Rozantsev, V. Lepetit, and P. Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137, 11 2014. doi 10.1016/j.cviu.2014.12.006.

- [44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature Publishing Group*, 323(6088), 1986.
- [45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- [46] R. Salas. Redes neuronales artificiales. *Universidad de Valparaíso. Departamento de Computación*, 1, 2004.
- [47] M. Sánchez and J. Castro. *Gestión y Minimización de Residuos*. Fundación Confemetal, 2007. ISBN 9788496743342.
- [48] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [49] O. Simeone. A very brief introduction to machine learning with applications to communication systems. *arXiv preprint arXiv:1808.02342*, 4(4), 2018.
- [50] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.
- [52] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.
- [53] A. Terceño Ortega. Análisis de un modelo predictivo basado en google cloud y tensorflow. Universidad Complutense de Madrid, 2017.
- [54] Unity Technologies. Unity Perception package, 2020. Disponible en <https://github.com/Unity-Technologies/com.unity.perception>.
- [55] S. C. Wang. *Artificial Neural Network*, volume 743 of *The Springer International Series in Engineering and Computer Science*. Springer US, Boston, MA, 2003. ISBN 9781461503774.
- [56] P. Warden and D. Situnayake. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2019. ISBN 9781492051992.

- [57] W. Yu and Y. Bai. Visualizing and comparing AlexNet and VGG using deconvolutional layers. 2016. Disponible en <https://icmlviz.github.io/icmlviz2016/assets/papers/4.pdf>.
- [58] J. Zamorano Ruiz et al. Comparación y análisis de métodos de clasificación con las bibliotecas scikit-learn y TensorFlow en Python. 2019. Universidad de Málaga.
- [59] J. Zurada. *Introduction to Artificial Neural Systems*. West, 1992. ISBN 9780314933911.

