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 [46]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 disponer 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 enfoca al sistema utilizado en el territorio nacional.

En España, en aplicaciones de reciclaje que identifiquen objetos contamos

con dos. La primera es "AIRE: Asistente Inteligente de Reciclaje"[1], 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], 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 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 más a la ayuda y el aprendizaje sobre reciclaje. Algunos ejemplos son "Residus" de la Generalitat de Catalunya[3] 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 desechado 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, 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 ambas opciones se explica el proceso que sufre este desecho desde el contenedor, su paso por la plantas 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" de Pensumo[4]. 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

[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`
[4] `https://reciclaysuma.com/`

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 la sostenibilidad. El juego consiste en alimentar a distintos 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 tu zona. Además, cuenta con *tests*, artículos y actividades para aprender apropiadamente 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.

[5] `https://theplanetapp.com/`
[6] `https://mylittleplasticfootprint.org/`
[7] `https://www.groplay.com/apps/grow-recycling/`
[8] `https://recyclecoach.com/`

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. A pesar de que actualmente ya es posible detectar y reconocer objetos en conjuntos de miles de imágenes complejas, todavía no se cuenta con un paradigma aceptado y dominante con el que la mayoría de investigadores trabajen [5].

La detección, seguimiento y reconocimiento de objetos en imágenes es uno de los problemas claves en visión computacional [11]. 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 [6].

Esto es algo utilizado en diversos ámbitos, desde objetos, personas y caras, hasta campos mucho más específicos. Un ejemplo es VOCUS (Visual Object detection with a CompUtational attention System) [14], 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 haya grandes contrastes o características únicas (por ejemplo una oveja negra en un rebaño de ovejas blancas). El segundo modo es de búsqueda con un objetivo definido, en este modo se utiliza información previamente aprendida sobre el objetivo para seleccionar las computaciones más destacadas respecto a ello.

Otro ejemplo es el reconocimiento de matrículas de los coches presentes en una imagen [36]. 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.

En el apartado 2.5 se hablará en más profundidad sobre varios proyectos de generación de *datasets* sintéticos. Todos ellos, además, tienen también como objetivo la identificación de objetos en imágenes.

Destacamos además, algunos proyectos semejantes al que se presenta en este trabajo de fin de grado, ya que cuentan tanto con parte de identificación de objetos, como de generación de *dataset*. En este punto del capítulo se hablará de lo primero, la identificación; en cambio la generación de imágenes se tratará más en profundidad en la sección 2.5.

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 [38], 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 [39]. SynthDet propone realizar la misma tarea de identificación y facturación de productos, pero entrenando el modelo a partir de imágenes sintéticas.

Otros proyectos relacionados son la detección de drones [42] y la identificación de piezas industriales [9], 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 Lens[9], 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 Fotos[10], una plataforma de intercambio y almacenamiento de fotografías y vídeos. 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 vídeos 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 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.

[9] `https://lens.google/intl/es-419/`
[10] `https://www.google.com/intl/es/photos/about/`

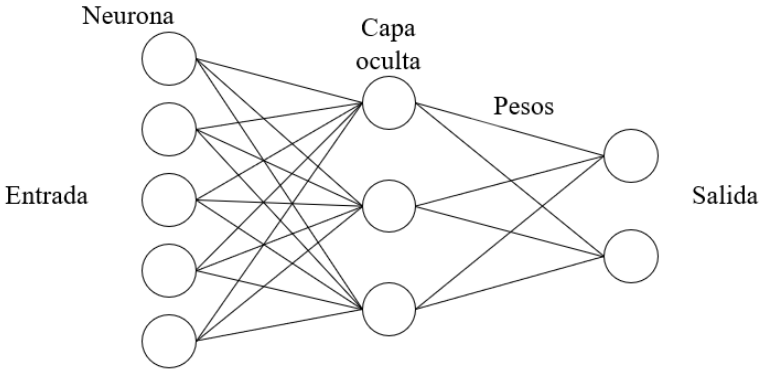

Figura 2.1: Estructura de una red neuronal artificial.

2.3. Redes neuronales

Todos los sistemas mencionados en el apartado anterior (2.2) utilizan redes neuronales por debajo. 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 [12]. 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 [18]. De esta forma, adquieren, almacenan y utilizan conocimientos basados en la experiencia en lugar de ser programados de manera explícita [57].

Una red neuronal artificial consta de una capa de entrada de neuronas, una o varias capas ocultas y una capa final de salida. En la figura 2.1 se muestra la estructura habitual de una red neuronal artificial.

Las neuronas reciben varias entradas y las combinan, le aplican la función de activación y el resultado es transmitido a otra neurona. La función de activación en redes neuronales devuelve una salida que es generada por una o varias entradas. Cada capa de la red cuenta con una función de activación. Las neuronas están conectadas entre sí mediante líneas y a cada una de estas conexiones se asocia con un valor numérico llamado peso [53, 45, 7].

Durante el aprendizaje de la red se modifica el valor de los pesos asociados a las neuronas con el fin de que generar una salida a partir de unos datos de entrada. Se considera que el aprendizaje ha terminado cuando los valores de los pesos permanecen estables.

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 [12, 16, 18, 35].

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 [12]. 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 [31]. Posteriormente, en 1958, Frank Rosenblatt comenzó el desarrollo de lo que terminaría conociéndose como perceptrón [41].

El perceptron 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 exclusiva[11]. Esto fue demostrado matemáticamente en los años 60 por Minsky y Papert [33], 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, Geoffrey E. Hinton y Ronald J. Willieams [43] del algoritmo de aprendizaje de propagación hacia atrás (*backpropagation*) planteado por Paul Werbos en 1974 [7, 27, 30, 15].

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 [37]. Una particularidad de estas redes es la operación de convolución[12] que se realiza en las primeras capas utilizándose como filtro, esto

[11] XOR 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.

[12] Una convolución es un operador matemático que transforma dos funciones f y g en

permite que tenga muchas aplicaciones en el tratamiento de imágenes [29]. 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 [37, 17].

También se lleva a cabo una distinción en las redes neuronales según el paradigma de aprendizaje que siguen, 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 [30, 45, 48].

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 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 [44], donde consiguió el primer puesto con una tasa de error del 15.3 % (2012). Fue la primera red neuronal en utilizar ReLu[13] como función de activación[14] en vez de la función sigmoide[15]. Este

una tercera que representa la magnitud en la que se superponen f y una versión trasladada e invertida de g.

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

[14]

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

cambio mejoró notablemente la velocidad de entrenamiento de las redes de aprendizaje profundo [10, 22, 55].

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 [49, 55, 34].

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 [50].

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 [47, 21].

2.4. TensorFlow y TensorFlow Lite

{**TODO TODO TODO:** Añadir comparaciones con otras opciones: PyTorch(el cual está un poquito en pañales) y Coffe2 (resultados similares pero con menos información disponible)}

El equipo de Google Brain comenzó a investigar en 2011 el uso de redes neuronales a gran escala para utilizarlo en los productos de la empresa. Como resultado desarrollaron DistBelief, sistema escalable de entrenamiento e inferencia, el cual fue utilizado para el entrenamiento de GoogleNet [50].

Basándose en la experiencia con DistBelief y con un entendimiento más completo de las necesidades y requisitos necesarios para generar un sistema mejor, desarrollaron TensorFlow[16]. TensorFlow es un entorno de trabajo de código abierto de aprendizaje automático que utiliza grafos de flujo de datos. Permite su ejecución en una gran variedad de diferentes plataformas, desde máquinas sencillas de una o varias GPUs, como los dispositivos móviles, hasta sistemas a gran escala de cientos de máquinas especializadas con miles de GPUs. La API de TensorFlow y las implementaciones de referencia fueron lanzadas como un paquete de código abierto bajo una licencia de Apache 2.0 en noviembre de 2015 [1, 51, 56].

En los grafos de flujo de datos de TensoFlow 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].

Para modelos en dispositivos integrados y móviles Google desarrolló la

[16] `https://www.tensorflow.org/`

librería de TensorFlow Lite. El entrenamiento de estos modelos se lleva a cabo en una computadora y el resultado se traslada posteriormente al dispositivo, sin necesidad de utilizar un servidor. Principalmente se utiliza la arquitectura MobileNet [47], la cual está diseñada y optimizada para ser ejecutada en móviles, y sirve para la detección y clasificación de objetos, detección de caras y reconocimiento de lugares en imágenes, aunque TensorFlow Lite también acepta el uso de SqueezeNet [22] e InceptionNet [50].

Como resultado del entrenamiento, se generan dos archivos en el ordenador en el que se haya llevado a cabo. El primero contiene las etiquetas disponibles para la clasificación, es un archivo de texto plano con una lista de todos los objetos para los que se ha entrenado a la red que detecte. El segundo tiene extensión *.tflite* y se trata del modelo entrenado de TensorFlow Lite. Ambos archivos deben importarse en la aplicación móvil para su uso. Estos modelos pueden manejarse en dispositivos Android, iOS y Raspberry, desarrollando la aplicación desde Android Studio, Swift u Objective-C; y Python, respectivamente.

2.5. Generación sintética de imágenes

Un problema con el que cuentan las redes neuronales es que necesitan muchos datos para el entrenamiento, y en el caso de que la red esté enfocada a la identificación de objetos es necesario contar con un gran número de imágenes para el aprendizaje. 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 [28].

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 [40] 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 [42]. 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 [9]. 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 [52] 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 diversidad de proyectos, desde simulación de coches autónomos o *smart cities*[17], pasando por demostraciones de recoger y colocar objetos mediante un brazo robótico[18], hasta detención de objetos utilizando únicamente imágenes sintéticas para el entrenamiento.

[17] `https://github.com/Unity-Technologies/Unity-Simulation-Smart-Camera-Outdoor`
[18] `https://github.com/Unity-Technologies/Robotics-Object-Pose-Estimation`

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 [38], creado por Amazon en 2017.

2.6. Unity

Unity[19] ha sido una de las principales tecnologías que se han utilizado para llevar a cabo este proyecto de final de grado, en concreto para realiza la generación sintética de imágenes. Unity es una herramienta de creación de videojuegos desarrollada por Unity Technologies. Pero esta herramienta no se limita solamente al ámbito de los videojuegos exclusivamente. Su uso está mucho más extendido por la cantidad de funcionalidades que ofrece de manera intuitiva y sencilla. Por esto, es utilizado en contenido cinematográfico, transporte y producción e incluso en arquitectura, ingeniería y construcción.

Unity permite la creación de escenas y objetos de manera muy sencilla. Además,la interacción con ellos no sólo se puede realizar a través de código, sino también mediante componentes visuales. utilizando Unity pueden desarrollarse juegos y aplicaciones para un gran abanico de plataformas; desde PC, Mac y Linux, pasando por Android e iOS, hasta consolas de videojuegos como PlayStation4, Xbox One, Nintendo Switch o Google Stadia.

Unity además cuenta con una clase base, `MonoBehaviour`[20], de la que prácticamente todos los scripts de Unity derivan. Esta puede ser activada o desactivada desde el editor de Unity según se requiera. `MonoBehaviour` ofrece numerosos métodos y mensajes, explicados en la documentación, que facilitan el desarrollo de aplicaciones en esta plataforma; los ejemplos más utilizados son las funciones `Start()`[21] y Update()[22]. La primera se llama cuando el script está habilitado antes de que se llame al `Update()` por primera vez. El Update(), en cambio, es llamado en cada frame si `MonoBehaviour` está activo.

Se tenía muy claro que para llevar esto a cabo se quería utilizar algún programa que ya ofreciera muchas de las funcionalidades necesarias ya desarrolladas. Con esto se quiso evitar el tener que crear desde cero escenas, la cámara y la iluminación. Puesto que esta parte del proyecto se basa en tratar de facilitar y automatizar la obtención y generación de datasets para el entrenamiento de redes neuronales, se decidió seguir ese ideal aprovechando aquello que nos facilita el proceso y que está a nuestro alcance, evitándose

[19] `https://unity.com/es`
[20] `https://docs.unity3d.com/ScriptReference/MonoBehaviour.html`
[21] `https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html`
[22] `https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html`

así el desarrollo desde cero de estas funcionalidades más complejas.

Existen varias herramientas que se podrían haber utilizado en lugar de Unity para llevar a cabo esta parte del proyecto. Unreal Engine[23] es otro motor de videojuegos cuyo uso también se extiende por numerosas industrias diferentes. Blender[24] es otro programa en el que se podría haber llevado esto a cabo. A pesar de que es una herramienta de creación de gráficos tridimensionales, Blender permite añadir funcionalidades, denominados *add-ons*, programados en Python; con lo que se podría haber tratado de crear la generación de imágenes.

Puesto que sobre estos programas no se tiene tanto conocimiento y experiencia previa como con Unity, se decidió que iba a ser más costoso de lo necesario, suponiendo que los resultados acabarían siendo similares pero con la posibilidad de que fueran peores al ser algo desconocido.

Para este proyecto, Unity aporta las funcionalidades más necesarias comentadas anteriormente. Además,cuenta con una amplia documentación y los muchos foros donde los usuarios comparten sus problemas, soluciones, experiencias y descubrimientos sobre esta herramienta.

A pesar de las facilidades que nos brida Unity queda todo el trabajo de la aplicación por hacer. El funcionamiento básico de esta consiste en ir cargando los modelos 3D guardados, uno a uno. En cada frame, el objeto que está cargado en la escena cambia su posición y rotación; además, también cambia el fondo.

[23] `https://www.unrealengine.com/en-US/`
[24] `https://www.blender.org/`

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 available from 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.

[3] R. Almond, G. M., and T. Petersen. Wwf (2020) living planet report 2020 - bending the curve of biodiversity loss. *World Wildlife Fund (WWF)*, 2020.

[4] O. Alsing. Mobile object detection using tensorflow lite and transfer learning. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

[5] Y. Amit. *2D Object Detection and Recognition: Models, Algorithms, and Networks*. Mit Press. MIT Press, 2002.

[6] Y. Amit, P. Felzenszwalb, and R. Girshick. *Object Detection*. Springer International Publishing, Cham, 2020.

[7] X. Basogain Olabe. Redes neuronales artificiales y sus aplicaciones. *Dpto. Ingeniería de Sistemas y Automática, Escuela Superior de Ingeniería Bilbao. Open Course Ware.[En línea] disponible en http://ocw. ehu. es/ensenanzas-tecnicas/redes-neuronales-artificiales-y-sus-aplicaciones/Course_ listing.[Consultada 20-09-2012]*, 2008.

[8] 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.

[9] 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, Feb. 2020. SCITEPRESS - Science and Technology Publications.

[10] G. Cortina Fernández. Técnicas inteligentes para su integración en un vehículo autómata. Trabajo de Fin de Grado en Ingeniería del Software, Facultad de Informática UCM, Departamento de Ingeniería del Software e Inteligencia Artificial, Curso 2019/2020., 2020.

[11] B. Cyganek. *Object Detection and Recognition in Digital Images: Theory and Practice*. Wiley, 2013.

[12] 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.

[13] R. Fonfría, R. Sans, and J. de Pablo Ribas. *Ingeniería ambiental: contaminación y tratamientos*. Colección productiva. Marcombo, 1989.

[14] S. Frintrop. *VOCUS: A visual attention system for object detection and goal-directed search*, volume 3899. Springer, 2006.

[15] L. García Rodríguez. *Algunas cuestiones notables sobre el modelo de Hopfield en optimización*. PhD thesis, Madrid, Noviembre 2018. Tesis de la Universidad Complutense de Madrid, Facultad de Ciencias Matemáticas, Departamento de Estadística e Investigación Operativa, leída el 15-12-2017.

[16] 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*, 2004.

[17] G. Guridi Mateos et al. Modelos de redes neuronales recurrentes en clasificación de patentes. B.S. thesis, 2017.

[18] J. R. Hilera and V. J. Martínez Hernando. *Redes neuronales artificiales: fundamentos, modelos y aplicaciones*. 01 1995.

[19] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2008. doi:10.5244/C.22.10.

[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.

[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. Iguarán Guerra, S. Gómez Ruíz, 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. Hogins, 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, Sep 2020.

[25] R. Karim. *TensorFlow: Powerful Predictive Analytics with TensorFlow.* Packt Publishing, Limited, 2018.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[27] P. Larranaga, I. Inza, and A. Moujahid. Tema 8. redes neuronales. *Redes Neuronales, U. del P. Vasco*, 12, 1997.

[28] P. López and J. García-Consuegra Bleda. *Informática gráfica*, volume 19. Univ de Castilla La Mancha, 1999.

[29] M. A. López Pacheco. Identificación de sistemas no lineales con redes neuronales convolucionales. *Cuidad de Mexico: Centro de investigación y de estudios avanzados*, 2017.

[30] D. J. Matich. Redes neuronales: Conceptos básicos y aplicaciones. *Universidad Tecnológica Nacional, México*, 41, 2001.

[31] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 1943.

[32] 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. Chapter 20 global climate change.

[33] M. Minsky and S. A. Papert. *Perceptrons: An introduction to computational geometry.* MIT press, 2017.

[34] L. Moreno Díaz-Alejo. Análisis comparativo de arquitecturas de redes neuronales para la clasificación de imágenes. Master's thesis, 2020.

[35] B. Müller, J. Reinhardt, and M. Strickland. *Neural Networks: An Introduction*. Physics of Neural Networks. Springer Berlin Heidelberg, 1995.

[36] C. Parra Ramos and D. Regajo Rodríguez. Reconocimiento automático de matrículas. *Universidad Carlos III de Madrid*, 2006.

[37] R. Pavón Benítez. Técnicas de deep learning para el reconocimiento de movimientos corporales. Trabajo de Fin de Grado en Ingeniería del Software, Facultad de Informática UCM, Departamento de Ingeniería de Software e Inteligencia Artificial, Curso 2019/2020, 2020.

[38] A. Polacco and K. Backes. The amazon go concept: Implications, applications, and sustainability. *Journal of Business and Management*, 24(1), 2018.

[39] A. Polacco and K. Backes. The amazon go concept: Implications, applications, and sustainability. *Journal of Business and Management*, 24(1), 2018.

[40] 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)*, June 2016.

[41] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[42] A. Rozantsev, V. Lepetit, and P. Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137, 11 2014.

[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088), 1986.

[44] 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.

[45] R. Salas. Redes neuronales artificiales. *Universidad de Valparaíso. Departamento de Computación*, 1, 2004.

[46] M. Sánchez and J. Castro. *Gestión y Minimización de Residuos*. Fundación Confemetal, 2007.

[47] 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.

[48] O. Simeone. A very brief introduction to machine learning with applications to communication systems. *IEEE Transactions on Cognitive Communications and Networking*, 4(4), 2018.

[49] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[50] 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*, June 2015.

[51] A. Terceño Ortega. Análisis de un modelo predictivo basado en google cloud y tensorflow. Trabajo de Fin de Grado en Ingeniería Informática y Matemáticas (Universidad Complutense, Facultad de Informática, curso 2016/2017), 2017.

[52] Unity Technologies. Unity Perception package, 2020.

[53] S.-C. Wang. *Artificial Neural Network*. Springer US, Boston, MA, 2003.

[54] P. Warden and D. Situnayake. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2019.

[55] W. Yu and Y. Bai. Visualizing and comparing alexnet and vgg using deconvolutional layers. 2016.

[56] J. Zamorano Ruiz et al. Comparación y análisis de métodos de clasificación con las bibliotecas scikit-learn y tensorflow en python. 2019.

[57] J. Zurada. *Introduction to Artificial Neural Systems*. West, 1992.