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 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 quizá 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) [27]. 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.

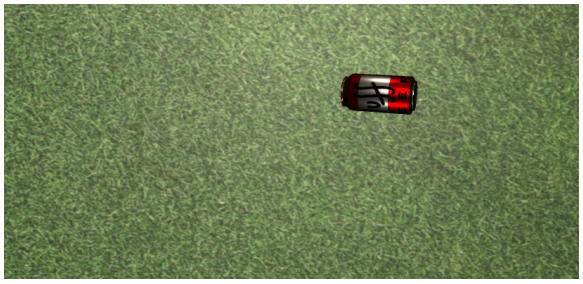
La pregunta que se plantea es si existe un punto medio idóneo, en el que se mezclen ambos extremos. En lugar de utilizar un *dataset* grande de fotografías a objetos físicos, o de imágenes sintéticas realistas, se plantea si es viable utilizar un *dataset* reducido de fotos reales enriquecido con imágenes 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 identificable 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.

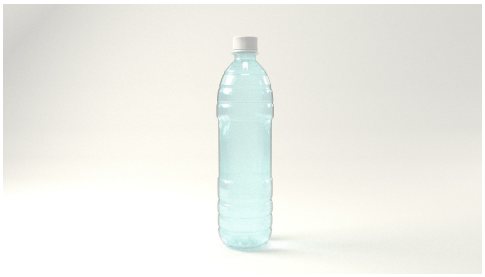
3.2.1. Modelos 3D

Como se ha comentadoen 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 (CGTrader[1], Free3D[2], TurboSquid[3], 3DModelHeaven[4] y Unity Asset Store[5]). 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.

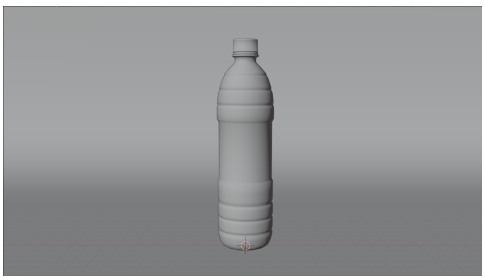
Existen varias dificultades principales con la obtención de modelos tridimensionales. En primer lugar está el realismo del modelo. Puesto que las capturas son para el entrenamiento de una red que se va a usar sobre objetos reales, es importante que los modelos utilizados sean lo más fiel posible al objeto 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 3.2a[6]. 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 Kaggle[7]. 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/`

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. Wwf (2020) living planet report 2020 - bending the curve of biodiversity loss. *World Wildlife Fund (WWF)*, 2020.

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

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

[6] 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.

[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.

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

[9] 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.

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

[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.

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

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

[14] 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.

[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*, 2004.

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

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] R. Karim. *TensorFlow: Powerful Predictive Analytics with TensorFlow.* Packt Publishing, Birmingham, 3 edition, Marzo 2018.

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

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

[27] 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.

[28] R. Liu. Higher accuracy on vision models with efficientnet-lite. *TensorFlow Blog.[online] Available at: https://blog. tensorflow. org/2020/03/higher-accuracy-on-visionmodels-with-efficientnet-lite. html [Accessed 30 Apr. 2020]*, 2020.

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

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

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

[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. Chapter 20 global climate change.

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

[35] S. Morant Gálvez. Desarrollo de un sistema de bajo coste para el análisis de tráfico mediante el uso de deep learning. 2021.

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

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

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

[39] 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.

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

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

[42] 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.

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

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

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

[46] 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.

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

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

[49] 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.

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

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

[52] 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.

[53] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[54] 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.

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

[56] S. C. Wang. *Artificial Neural Network*, volume 743 of *The Springer International Series in Engineering and Computer Science*. Springer US, Boston, MA, 2003.

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

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

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

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