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

{**TODO TODO TODO:** añadir: Para el correcto funcionamiento, en Andorid, de la aplicación con el modelo entrenado son necesarias dos librerías. La primera es la librería de tareas de TensorFlow Lite, que cuenta con un conjunto de bibliotecas específicas de tareas potentes y fáciles de usar por los desarrolladores para crear experiencias de aprendizaje automático. Esta funciona varias plataformas y es compatible con Java y C++. La segunda librería necesaria es la de compatibilidad, esta facilita la integración de modelos en la aplicación. Proporciona API de alto nivel que ayuda a transformar los datos de entrada en el formato requerido por el modelo, además de interpretar su salida.}

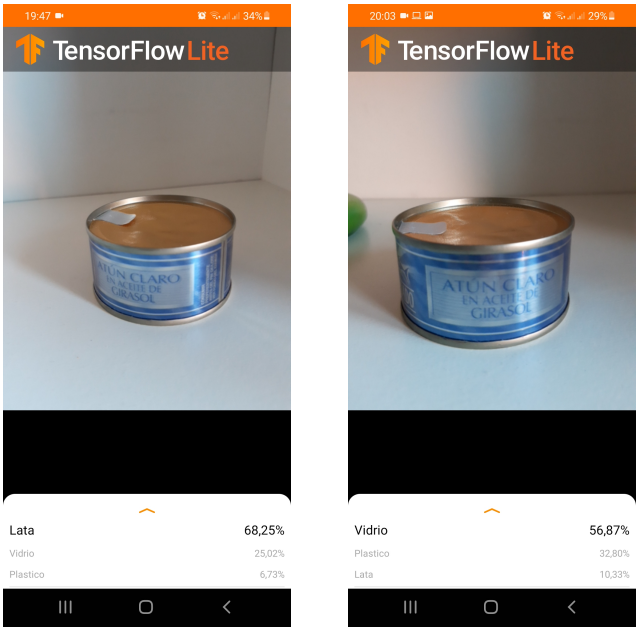
Para realizar la identificación se ha incorporado TensorFlow Lite y sus librerías al proyecto en Android Studio. Esto permite importar un intérprete, el cual carga el modelo y permite ejecutarlo ofreciéndole una serie de datos de entrada, finalmente, tras ejecutarlo, se muestran por pantalla los resultados 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 **??**).

Durante el flujo de la aplicación se hace uso de la cámara del dispositivo, leyendo *frames* desde esta y los convirtiéndolos a imágenes. Estos 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 rápida[1]. 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 más 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

[1] `https://www.tensorflow.org/lite/performance/best_practices?authuser=1`



(a) (b)

Figura 5.1: Aplicación de identificación desarrollada

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 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, estos permiten acelerar la ejecución de los operadores. Sin embargo, el aumento de 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.

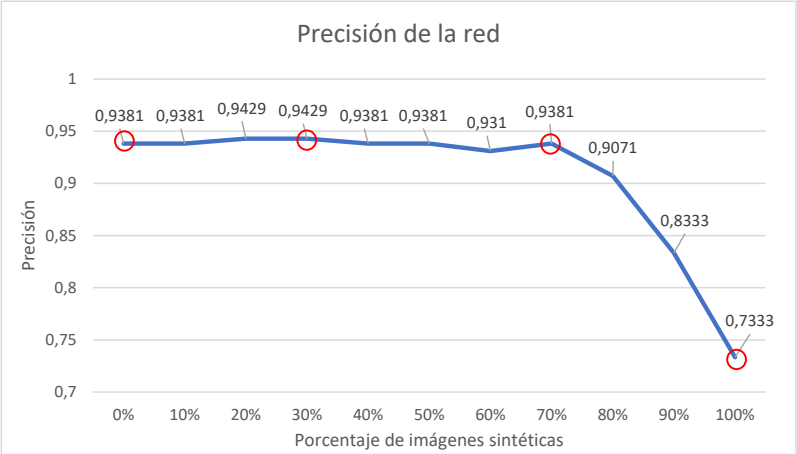


Figura 5.2: Variación de la precisión de la red neuronal con los modelos seleccionados destacados

5.3. Pruebas

Con todo lo necesario desarrollado, se llevan a cabo varias pruebas para comprobar el funcionamiento de la aplicación de identificación. Tomando los resultados en el apartado 4.2 se decidió que el mejor de los modelos obtenidos era para el que en las latas se habían utilizado un 70 % de imágenes generadas, con un 93 % de precisión. 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 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.

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 se-

gunda posición. Puesto que la aplicación 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 corresponde a cada uno de los modelos, estos se identifican indicando los porcentajes de imágenes generadas y reales usadas en su entrenamiento. Estas se expresan primero con el porcentaje de imágenes reales, acompañado de la letra "R" en mayúscula, y después el porcentaje de imágenes generadas, seguido de la letra "G", ambos separados por un guión entre medias. Es decir, el modelo con todas las imágenes reales sería representado por "100R - 0G". De todas formas, para mayor claridad se ordenan de izquierda a derecha según disminuye la cantidad de imágenes reales.

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. A excepción del modelo entrenado por completo con imágenes sintéticas, el cual los identifica como vidrio. Un caso destacable es el de la lata de conservas con el modelo completamente formado por imágenes reales, el cual la identifica mayoritariamente como vidrio, pero no llega a superar el 50 %. Este resultado se explica con la ausencia de imágenes de objetos de este tipo que contaba dicho *dataset*, lo cual significa que el entrenamiento con las imágenes sintéticas está aportando información útil, e incluso mejorando, la identificación.

Una observación importante que se descubre, es la dificultad, en todas las opciones, para distinguir los objetos de vidrio respecto a los de plástico. Para tres objetos de vidrio diferentes todas los identifican como plástico con más de un 60 % de confianza. Esto no genera mucha preocupación puesto que como seres humanos a veces también es costoso diferenciar plástico translúcido de vidrio a simple vista.

Un problema importante que se ha observado es la variabilidad del resul-

	100R - 0G	70R - 30G	30R - 70G	0R - 100G
Metal	23,58%	58,69%	**58,69%**	9,39%
Vidrio	42,21%	26,84%	**26,84%**	54,42%
Plástico	34,21%	14,47%	**36,20%**	14,47%

(a) Lata de conservas

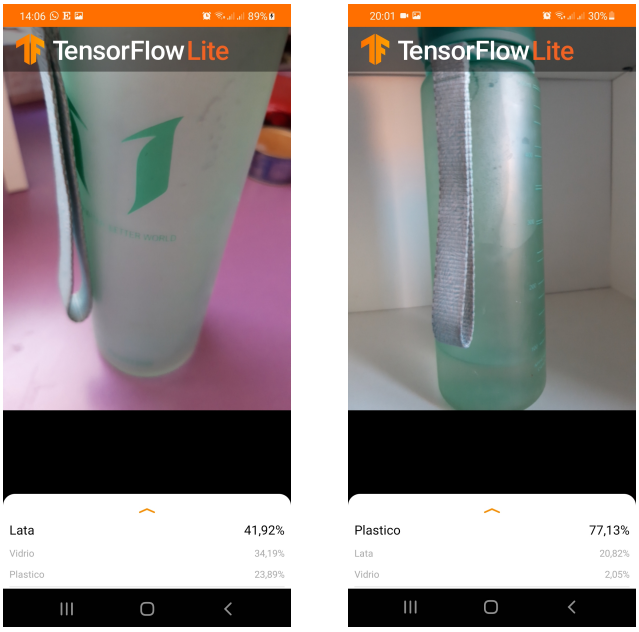
	100R - 0G	70R - 30G	30R - 70G	0R - 100G
Metal	88,78%	89,12%	**89,12%**	4,32%
Vidrio	5,83%	8,14%	**8,14%**	60,56%
Plástico	5,39%	2,74%	**35,12%**	2,74%

(b) Lata de bebida

	100R - 0G	70R - 30G	30R - 70G	0R - 100G
Metal	84,91%	84,19%	**84,19%**	8,36%
Vidrio	8,42%	11,77%	**11,77%**	60,73%
Plástico	6,67%	4,04%	**30,91%**	4,04%

(c) Lata de encurtidos

Figura 5.3: Comparación con distintos objetos metálicos



(a) 100 % imágenes reales (b) 100 % imágenes reales

Figura 5.4: Comparación con una botella de agua.

tado dependiendo de la iluminación. Como se puede observar en la figura 5.4 para el mismo objeto y con el mismo modelo, en este caso el generado a partir de solamente imágenes reales, se obtienen resultados diferentes al cambiarlo de posición.

Figura 5.5: Resultados del modelo 70-30 con una lata de bálsamo labial.

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

[18] 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.

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

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

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

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

[38] 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.

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

[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)*, June 2016.

[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.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 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.

[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. *IEEE Transactions on Cognitive Communications and Networking*, 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*, June 2015.

[52] 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.

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

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

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

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

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

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