





## Capítulo 4

# Entrenamiento y selección de *dataset*

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

### 4.1. Experimento

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

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

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

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

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

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

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

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

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

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

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

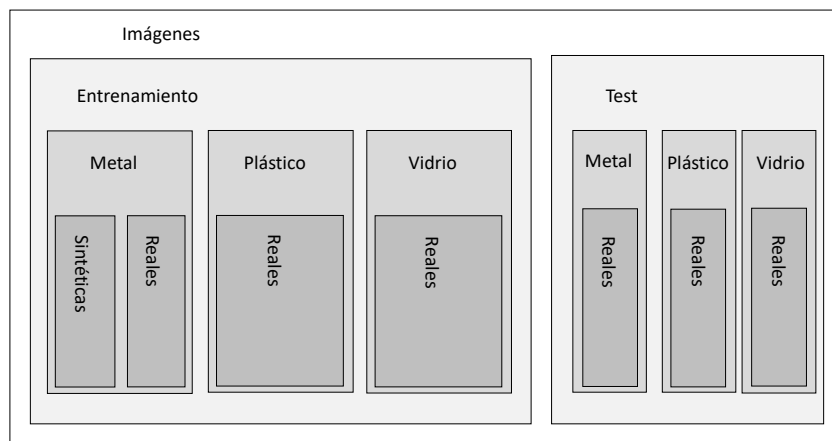


Figura 4.1: Organización de las carpetas con las imágenes separadas.

## 4.2. Entrenamiento

Ya se ha contado qué se va a hacer en el capítulo. Ahora viene el como. Párrafos que tienes ya escritos en el punto 4.1. Código de ejemplo de las flores, épocas, lotes, fichero .tflipe, ... La fase de entrenamiento del proyecto está basada en las recomendaciones para generar modelos de TensorFlow Lite, ya que el modelo que se va a utilizar para la identificación es el de este formato. Para llevar a cabo el entrenamiento de la red neuronal se ha realizado un *script* que se encarga de la lógica de este. Dicho *script* está basado en uno de los ejemplos de Tensorflow Lite y puede usarse desde Colaboratory<sup>1</sup>, plataforma de Google que permite programar y ejecutar en Python

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

desde el navegador. El ejemplo elegido como referencia es un programa de identificación de flores<sup>2</sup>. Se ha seleccionado este debido a su similitud con el proyecto respecto a la identificación del objeto principal en un imagen.

Para generar el modelo usando las imágenes obtenidas se utiliza la biblioteca Model Maker de TensorFlow Lite, que permite realizar el entrenamiento utilizando aprendizaje por transferencia. Para ello se ha utilizado el modelo de EfficientNet-Lite, que aún no siendo el más exacto es el más recomendable para dispositivos móviles por su tamaño y latencia [28] permitiendo ser utilizado por un mayor número de dispositivos independientemente de si son de gama baja o alta. El entrenamiento para este proyecto se lleva a cabo desde la computadora, evitando tener que cargar todo el conjunto de imágenes en Colaboratory. Para ello se tiene un *script* que realiza la carga de las imágenes, el entrenamiento, la creación del modelo y las pruebas de rendimiento.

El entrenamiento se divide en varios episodios. Estos corresponden al número de veces que el algoritmo recorre todos los datos. En cada episodio los datos se dividen en lotes, y en cada lote se recoge una pequeña parte de los datos. Cada lote se recorre en una iteración, por lo que cada episodio tiene las iteraciones necesarias para recorrer al completo el conjunto de datos. Esto significa que el número de iteraciones de cada episodio corresponde al resultado de dividir la cantidad de datos entre el tamaño de lote [57]. Por ejemplo, en los entrenamiento se tiene un total de 3769 imágenes (1000 para metal, 811 para vidrio y 1958 para plástico) y un tamaño de lote de 32, por lo tanto en episodio hay 117 iteraciones.

Una vez finaliza el entrenamiento y el *test* de la red se genera un archivo con extensión *.tflite*, además de un documento de texto plano con las etiquetas de los distintos materiales. Una vez se haya comparado el rendimiento de los once modelos entrenados, el documento de las etiquetas y el archivo *.tflite* correspondientes al modelo seleccionado se trasladan a la aplicación móvil para ser utilizados.

### 4.3. Resultados

Como último punto, queda concluir con qué porcentaje de imágenes sintéticas se obtiene el mejor resultado en exactitud

La figura 4.2 corresponde al crecimiento de la exactitud de la red durante el entrenamiento. La exactitud es el porcentaje de aciertos de la red respecto a los resultados reales. Se encuentra representado para los distintas combinaciones de imágenes generadas y reales. Al ser la exactitud durante el entrenamiento esta se calcula usando directamente las mismas imágenes del *dataset* de entrenamiento, no las de test. En este proceso se le está enseñando

---

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

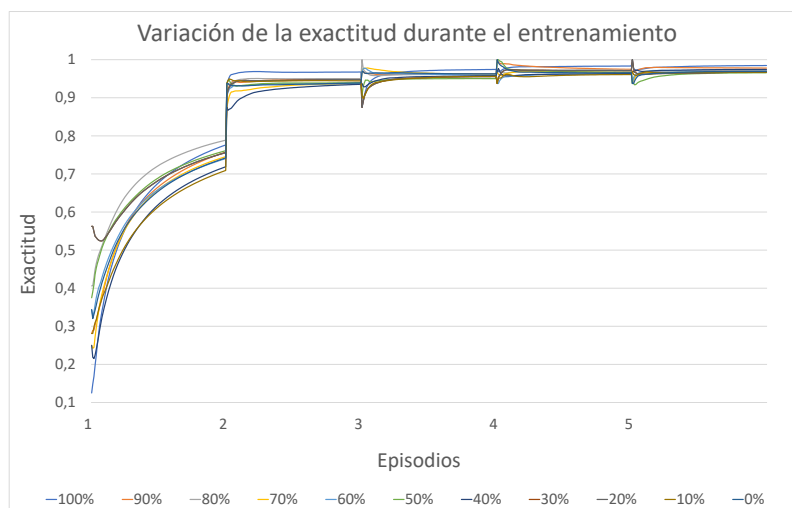


Figura 4.2: Variación de la exactitud de la red neuronal a lo largo del entrenamiento

a la red a categorizar las imágenes, el valor de la exactitud en este caso se obtiene según si la red sabe categorizar las mismas imágenes sobre las que se le está entrenando. Que en todos los casos crezca de manera similar en todos los casos significa que está aprendiendo bien para las imágenes que se le están proporcionando. Es decir, en el caso extremo donde todas las imágenes son de objetos metálicos, la red las está categorizando correctamente como metal. No obstante, si atendemos a los resultados de las figuras 4.3 y 4.4, al probar el modelo entrenado sobre imágenes reales la exactitud se desploma, significando que no identifica adecuadamente los objetos metálicos reales. Como se ha comentado, durante el entrenamiento se aprecia cómo la exactitud de todos los *datasets* tiene un valor superior al 90 % en casi todos los casos a partir del segundo episodio. Aunque dicho valor se acaba acercando al 100 % en ningún momento llega a alcanzarlo, lo que evidencia que no hay sobreajuste. De lo contrario, significaría que los modelos han sido sobreentrenados y por lo tanto que conocen el resultado deseado.

En la figura 4.3, por el contrario, se observa cómo varía la exactitud al probarse el modelo con los datos de *test*, aquí se sacan las primeras conclusiones sobre la red. Se observa que la mayoría de las opciones se mantienen casi todo el proceso en una precisión mayor del 90 %. Los casos en los que se puede contemplar una caída de la exactitud, son aquellos que cuentan con

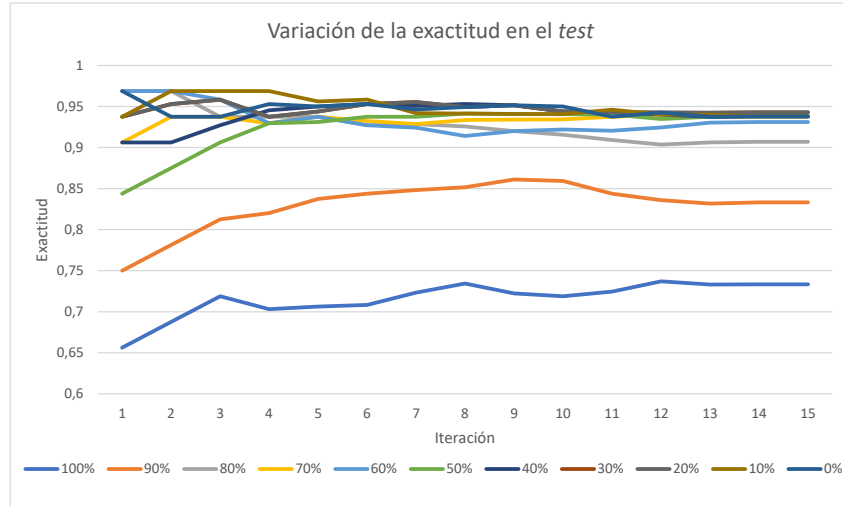


Figura 4.3: Variación de la exactitud de la red neuronal durante el *test*

mayor porcentaje de imágenes sintéticas para el entrenamiento. Esto se debe a que al haber visto muy pocas imágenes reales de este material durante el entrenamiento ahora en el *test* no las identifica como metal provocando que la exactitud decrezca notablemente.

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

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

De cara a elegir cuál de los modelos entrenados utilizar en la aplicación para dispositivos móviles, se tiene en cuenta el rendimiento de la red y la facilidad de obtención del *dataset*. Esto se traduce en que se busca un modelo con un alto nivel de exactitud pero que también cuente con un gran número de imágenes generadas, ya que estas, con lo desarrollado en el proyecto, se pueden conseguir de manera más fácil que las reales.

Finalmente, se considera como mejor opción el modelo entrenado con un 70 % de imágenes generadas en el material de metal. Esta opción es



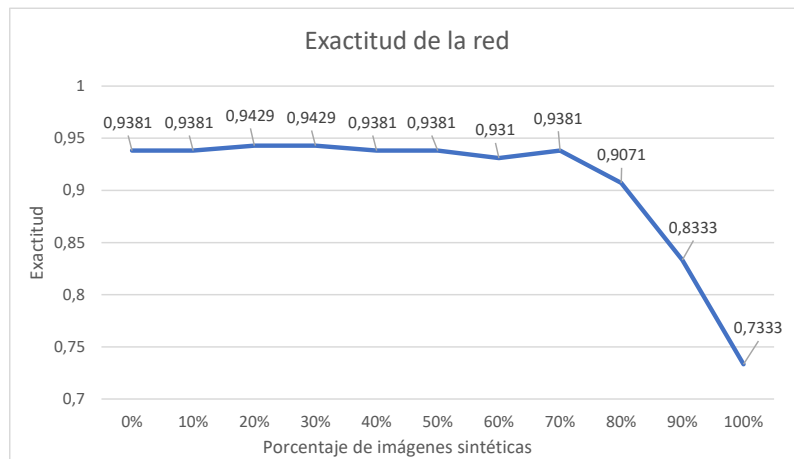


Figura 4.4: Variación de la exactitud final de la red neuronal según la proporción de imágenes reales y sintéticas

de las que mayor precisión tienen y a la vez permite tener un *dataset* que funciona adecuadamente formado principalmente por imágenes sintéticas, lo que facilita su obtención.

# 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/enseñanzas-tecnicas/redes-neuronales-artificiales-y-sus-aplicaciones/Course\\_listing](http://ocw.ehu.es/enseñanzas-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 autónoma. 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. We-  
yand, 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 param-  
eters 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. Ho-  
gins, 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 informa-  
tion 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*, volu-  
me 19. Univ de Castilla La Mancha, 1999.
- [30] M. A. López Pacheco. Identificación de sistemas no lineales con redes  
neuronales convolucionales. *Ciudad de Mexico: Centro de investigación  
y de estudios avanzados*, 2017.
- [31] D. J. Matich. Redes neuronales: Conceptos básicos y aplicaciones. *Uni-  
versidad 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.