Ensayo introductorio, descriptivo y práctico sobre las redes neuronales

Concepción, historia, técnica y experimentación

Autor: Alejandro Garnung Menéndez

Universidad de Oviedo - curso 2022-2023

Resumen

En este documento se plasman varias palabras que tratan sobre *lo relativo al arte de crear cualidad de discernimiento*, a saber, sobre la Inteligencia Artificial (otras definiciones de varios autores en [1, pág. 16]). Más específicamente, sobre una de las aportaciones foráneas más importantes a esta rama, las redes neuronales artificiales (RNA), que tan en boga se encuentran por virtud de su característica funcional más importante: la capacidad de aprender de manera automática, no persiguiendo la creación formalizada del conocimiento, sino más bien emularlo a través del entrenamiento o aprendizaje [2]. Yuxtapuestamente a los resultados adjuntos a este documento que proporcionan diversas experimentaciones mediante herramientas de MATLAB, en las que se estudian particularmente las redes neuronales (artificiales) convolucionales, se pretende que el lector recoja y absorba de manera precisa numerosos conceptos sobre estos respectos y muchos otros que conforman, es un amplio conjunto, los sistemas inteligentes artificiales. Todo esto teniendo siempre presente la cuestión que, en su día, desató fulgentemente el inicio de este campo de estudio: *¿pueden las máquinas pensar más allá de lo que podemos ordenarles?*, pregunta "cuyas ramificaciones se siguen explorando hoy día".

Índice

Glosario de siglas

ACO	*Ant Colony Optimization*
AI/IA	*Artificial Intelligence/Inteligencia Artificial*
AN/NA	*Artificial Network/Neurona Artificial*
ANN/RNA	*Artificial Neural Network/Red Neuronal Artificial*
ART	*Adaptative Resonance Theory*
BM	*Botzmann Machine*
BN	*Batch Normalization*
BP	*Backpropagation*
CART	*Classification and Regression Tree*
CNN/RNC	*Convolutional Neural Network/Red Neuronal Convolucional*
CV	*Cross-validation*
DFA	*Deterministic Finite-state Automata*
DL	*Deep Learning*
DNN	*Deep Neural Network*
DoG	*Diferencia de Gaussianas*
DT	*Decision Trees*
EA	*Evolutive Algorithms*
EKF	*Extended Kalman Filter*
FCN	*Fully Convolutional Network*
FPN	*Feature Pyramid Network*
FSM	*Finite-state Machine*

GA	*Genetic Algorithms*
GAN	*Generative Adversarial Network*
GCN	*Graph Convolutional Networks*
GHA	*Generalized Hebbian Algorithm*
HMM	*Hidden Markov Model*
ILSVRC	*ImageNet Large Scale Vision Recognition Challenge*
IR	*Image Recognition*
IS	*Image Segmentation*
KNN	*K-Nearest Neighbors*
LMS	*Least Mean Square*
LoG	*Laplaciana de Gaussiana*
LR	*Learning Rate*
LRN	*Local Response Normalization*
LSTM	*Long Short Term Memory*
LVCSR	*Large Vocabulary Continuous Speech Recognition*
ML/AM	*Machine Learning/Aprendizaje Automático*
MLP	*Multilayer Perceptron*
MWC	*Moving Window Classifier*
NAP	*Neural Abstraction Pyramid*
NLP	*Natural Language Processing*
NN/RN	*Neural Network/Red neuronal*
OCR	*Optical Character Recognition*

OD	*Object Detection*
PSO	*Particle Swarm Optimization*
RBF	*Radial Basis Functions*
R-CNN	*Region Based CNN*
ReLU	*Rectified Linear Unit*
RF	*Random Forest*
RGBA	*Red Green Blue Alpha*
RMLP	*Recurrent Multilayer Perceptron*
RNN	*Recurrent Neural Network*
ROI	*Region of Interest*
RPN	*Region Proposal Network*
RUS	*Random Undersampling*
SA	*Simulated Annealing*
SGD(M)	*Stochastic Gradient Descent (with Momentum)*
SMOTE	*Synthetic Minority Oversampling Technique*
SRN	*Simple Recurrent Network (o Elman's RNN)*
SURF	*Speeded Up Robust Features*
SVC	*Support Vector Classifier*
SVM	*Support Vector Machines*
TDNN	*Time delay NN*
TQC	*Total Quality Control*
TS	*Taboo Search*

VAE	*Variational Autoencoder*
VLSI	*Very Large Scale Integration*
XNOR	*Disyunción opuesta exclusiva lógica*
XOR	*Disyunción exclusiva lógica*
YOLO	*You Only Look Once*

Lista de figuras

Lista de tablas

Introducción al documento

En el primer apartado, “Presentación previa del trabajo práctico realizado”, se describe de manera precisa el alcance de los aspectos prácticos a desarrollar planteados que envuelven el trabajo realizado en lo relativo a la programación, implementación y el estudio de las redes neuronales artificiales convolucionales en la herramienta MATLAB, enmarcada de una manera lo más clara y lacónica posible a fin de que el lector tenga una visión completa y suficiente, desde un principio, del apartado práctico que se ha cubierto el presente trabajo.

En el segundo apartado, “Introducción a las redes neuronales artificiales”, se recoge una sucinta reseña descriptiva, histórica y técnica, a la par que curiosa, de la amplia rama del conocimiento que es la inteligencia artificial, con un enfoque especial orientado al ámbito del aprendizaje automático y a múltiples conceptos relacionados con todo lo que pueden abarcar (intemporalmente) las redes neuronales artificiales.

Posteriormente, un tercer apartado llamado “El modelo biológico de las neuronas” trata, de una forma restringida pero relacionada con el propósito de este documento (proporcionar al lector una idea enriquecida sobre los temas que se analizan), acerca del modelo biológico de las redes neuronales y de la asombrosa analogía subyacente y pivotante entre este y el modelo computacional que constituyen las redes neuronales artificiales.

En el cuarto apartado, “Las redes neuronales artificiales convolucionales”, se presenta de una manera más precisa el tipo de RNA que son de mayor interés para la discusión principal de este trabajo y el ulterior análisis práctico de ciertas experimentaciones realizadas con tales redes. En este punto del informe habrían quedado contemplados un amplio conjunto de métodos que se podrían emplear en la práctica más sus bases teóricas y se lograría hacer una composición de lugar de lo que se tratase a continuación.

El antepenúltimo apartado, “Simulación de las RNC con MATLAB”, está subdivido en múltiples subapartados cuyos nombres dan una idea aproximada y concisa sobre su contenido. A lo largo de este se plasman y documentan secuencialmente varios aspectos relativos al trabajo práctico realizado, a decir: el enunciado y la descripción del problema

propuesto, las explicaciones, justificaciones y los detalles de los métodos empleados para extraer los resultados de los experimentos desarrollados, breves comentarios que completan y aclaran el código programado para la simulación de redes neuronales artificiales y diversas validaciones de hipótesis, modelos, cuantificaciones de resultados y conclusiones que completan la solución particular desarrollada para este trabajo con ayuda de la herramienta MATLAB. Varios métodos son empleados para tal labor que, de manera heurística, sirven auxiliarmente para introducir y tratar numerosos conceptos sobre la inteligencia artificial y, específicamente, las redes neuronales convolucionales, como ya se va contemplando en el resto de apartados y los propios subapartados de que consta este, todos contextualizados enfáticamente sobre la comparación entre conceptos teóricos de interés y resultados empíricos (extraíbles y verificables versátil y fácilmente) obtenidos gracias al código proporcionado junto al documento.

Finalmente se presentan varias conclusiones y preguntas abiertas sobre múltiples aspectos tratados (“Conclusión”, aunque mucho de esto abarcan eventualmente los anteriores subapartados) y se citan varias referencias bibliográficas y lecturas recomendadas (“Referencias, bibliografía y lecturas recomendadas”, donde todas y cada una de las citaciones se consideran altísimamente interesantes) que son de valiosa relevancia para profundizar en cualquier ámbito de estudio que se infiera del presente trabajo.

Presentación previa del trabajo práctico realizado

De manera independiente al resto del contenido de este documento, pero reforzándose enormemente con todos los conceptos que se presentan y discuten, una parte preponderante del trabajo realizado consta de la realización de distintos experimentos, en base a varios *datasets* de imágenes aplicados a una red convolucional, para detectar o clasificar objetos defectuosos y sin defectos y racionalizar los resultados que proporcione la evaluación de la clasificación, extraer ventajas y desventajas de cada variación experimental y relacionar (heurísticamente) todo ello con conceptos típicos e interesantes de la inteligencia artificial.

Se comienza analizando varios datasets y escogiendo uno solo, con objetos de carácter industrial (piezas metálicas) para clasificarlos definiendo y comprendiendo una arquitectura de RNC sencilla e introductoria. Se experimenta (en todo momento) con la partición de datos de entrenamiento, validación y test para optimizar la combinación que resulte en una mayor métrica de precisión; también se prueba a variar el número de capas ocultas y neuronas en cada una, además de añadir tipos especiales de capas o modificar el tamaño del filtro de convolución comparando ciertos resultados que brindan varias pruebas de clasificación (un test solamente con imágenes sin defectos, otro únicamente con imágenes defectuosas y un tercero con muestras de ambas clases [elegidas aleatoriamente]) programadas para validar regularmente los distintos modelos que se prueban. Se hacen modificaciones en el tamaño del lote de los conjuntos de imágenes que sirven de datos de entrenamiento y también en el tipo de defectos con que se entrena; con todo esto, se van presentando secuencialmente varias arquitecturas de redes que agregan ciertas mejoras a las previas (capas de normalización, funciones de activación más potentes, aumento de la profundidad de aprendiza, etc.).

Extendiendo los experimentos, en una segunda parte se propone agregar tres lotes más de imágenes (transistores bipolares, botellas de cristal en plano cenital y cables eléctricos) para implementar, modificar ligeramente y entrenar una estructura AlexNet “vacía”. El entrenamiento se complica y el tiempo de computación se alarga, así como surge la necesidad de adaptar de cierta forma la red y los datasets entre sí. Se vuelve a probar a entrenar la red mediante otros experimentos específicos con lotes de diferentes tamaños

y, ahora, escalando en mayor o menor grado el tamaño (la resolución) de las muestras para ver si puede existir correlación entre este (u otros parámetros) y la precisión de clasificación.

En una tercera etapa, se estudia la implementación de algunas técnicas y varios métodos tradicionales de visión artificial en aplicación al preprocesamiento de las imágenes de prueba y entrenamiento para tratar de buscar "alguna forma" de la que se pueda mejorar el comportamiento de la red para enseñándole a "generalizar" mejor el discernimiento de categorías y, de cierta forma, aprender mejor ante la detección de los patrones y las características "que realmente importan". Se proponen e implementan técnicas como el filtrado no lineal de mediana, de moda, la normalización o el ajuste del nivel de intensidad de los canales de color de la imagen u otros tipos de filtros con el propósito de resaltar y extraer, específicamente, "lo determinante" para la clasificación; además se comprueban los resultados de implementar estos métodos y se discute su validez, en una comparativa con predicciones (pruebas) similares que proporcionan redes anteriores (todo esto acompañado de múltiples reseñas, investigaciones y artículos o comentarios relativos al estado del arte del estudio y cantidad de conceptos teóricos y prácticos considerados fundamentales). Junto con este informe se adjunta el fundamental código fuente desarrollado que, velando por una modularidad en la estructuración versátil de información (algo que proporcionan los visuales *scripts* realizados), recoge diversos comentarios ilustrativos, instrucciones y descripciones de lo que se ha implementado en la práctica con el presente trabajo.

Finalmente se logran analizar los resultados de todas las redes desarrolladas, a los sumo, mediante métricas como la precisión de entrenamiento, validación, convergencia de los pesos y parámetros de la red a lo largo de las épocas, decrecimiento de la función de pérdida o matrices de confusión, para extraer resultados inteligibles y sumamente útiles para dar paso a potenciales ampliaciones de los modelos propuestos, pues lo trabajado en este documento puede servir perfectamente como una iniciación introductoria a este campo de estudio tan extenso y sorprendente que es el aprendizaje automático.

Introducción a las redes neuronales artificiales

La inteligencia artificial (IA, que engloba al *Machine Learning*, que a su vez engloba al *Deep Learning*) es toda una ciencia y rama del conocimiento en la que se busca desarrollar herramientas que ayuden a mejorar la claridad de compre(he)nsión del estudio de los agentes inteligentes (entidades que conciben su entorno a través de sensores y responden, en concordancia, con una serie de actuaciones racionales) y la mejora de la cualidad de aprendizaje en el marco de los sistemas computacionales, cualidad que con particular enfoque e interés hacia las redes neuronales se puede definir, de acuerdo con [3], como "el proceso por el que los parámetros libres de una red neuronal (artificial) son adaptados en concordancia con los estímulos que capta del entorno en que la red está integrada; siendo el tipo de aprendizaje determinado por la manera de la que toman lugar los cambios en los parámetros" [4].

El bagaje histórico de que se dispone sobre la evolución de la inteligencia artificial, desde sus raíces y si se logra encontrar, es tal que no debería ser de asombro el hecho de que muchas de las más novedosas técnicas de la computación informática inteligente (algoritmos evolutivos, aprendizaje automático, lógica difusa, [meta]heurísticos…) estén basadas inherentemente en ideas heredadas de la profundidad del pensamiento creativo y reflexivo a la hora de tratar de comprender metodológicamente los fenómenos naturales y la misma inteligencia humana; véanse los archiconocidos tratados filosóficos y matemáticos de Platón (427-347 a.C.) y Sócrates (384-422 a.C.) sobre la relación natural del pensamiento racional y lógico partiendo de conocimiento codificado a través de lenguaje interno [121], las discusiones intemporales sobre el proceso mental que mantuvieron Descartes (1596-1650) [5] y los filósofos empiristas del siglo XVIII [6] o diversos escritos de Leibniz (1646-1716) [7], Asimov (1920-1992) [8], Dreyfus (1929-2017) [9], Penrose (1931-) [10], Searle (1932-) [11, 12]… acerca de la dubitativa entre si es más probable o incierta la posibilidad de concebir una máquina como un mecanismo que emula al cerebro humano por medio de pensamiento artificial.

Es preciso y necesario tener una retrospectiva general de lo acontecido en cualquier ámbito del conocimiento en el que la inteligencia artificial haya dejado cierta huella, entendiendo esta como un sinónimo indeterminado de otros conceptos como computación

neuronal, procesos, sistemas o máquinas autónomas o inteligentes… con los que, al hablar de ellos, se tiene siempre en mente su relación (pues todos se tratan de su conceptualización) con la capacidad del sistema nervioso biológico de controlar el comportamiento sensorial, motor y, a más bajo nivel, cierto proceso interno denominado "pensamiento" [6, pág. 1]. El estudio de reproducir muchas de las funciones biológicas humanas o animales artificialmente ha sido tema de interés para científicos y eruditos a lo largo de la historia. Desde cantidad de siglos atrás ha habido intentos (frustrados y exitosos) de replicar el comportamiento humano (como sus acciones y movimientos) mediante algún tipo de mecanismo (al principio, mecánico) que simulase sus funciones, denominado típicamente "autómata" (del griego *autómatos* [que se mueve por sí mismo]); por nombrar unos pocos ejemplos relevantes [13]: un autómata hidráulico ("máquina de vapor") de la mano de Herón de Alejandría (ca. 10-70 d.C.), varias figuras humanoides automatizadas mediante cadenas mecánicas (Al-Jazari, ca. 1200), el robot humanoide mecánico en forma de caballero de Leonardo da Vinci (1495), varios sistemas complejos de la mano de Jacques Vaucanson y Pierre Jaquet Droz (s. XVIII) o, sumamente importante [14], el teleautómata sumergible de Nikola Tesla (presentado en 1898) operado mediante sus transistores de aumento o, de muy especial interés, las redes neuronales artificiales, que ya desde su nacimiento tan adecuadamente han progresado hasta la actualidad junto al acelerado desarrollo de la ciencia y tecnología de la información y computación en una convergencia de materializar sintéticamente tareas cognitivas en las que, "por lo general y hasta la actualidad, los humanos son mejores" [15].

Anecdótica y sorprendentemente, en el nexo de los siglos XII y XIII el polifacético estudioso Ramón Llull ideó una máquina lógica mecánica orientada a la combinación de conceptos para generar otros distintos, o sea salidas deseadas (ver siguiente figura). Llull daba a entender que, aun con todo el ingenio depositado artificiando su mecanismo este pudiera efectuar razonamiento, no llegaba a alcanzar la perfecta emulación de la inteligencia humana (todo esto con cierto carácter teológico), compartiendo significancia con las tardías palabras de Winter: "en dominios como la visión artificial, el discurso y los procesos motores la inteligencia humana es más poderosa que mil supercomputadores,

sin embargo, para tareas simples como la multiplicación, esta es menos capaz que un microprocesador de cuatro bits" (a pesar de que los eventos en las puertas de silicio ocurran en lapsos de nanosegundos y en las redes neuronales en milisegundos) [16] [17, pág. 38].

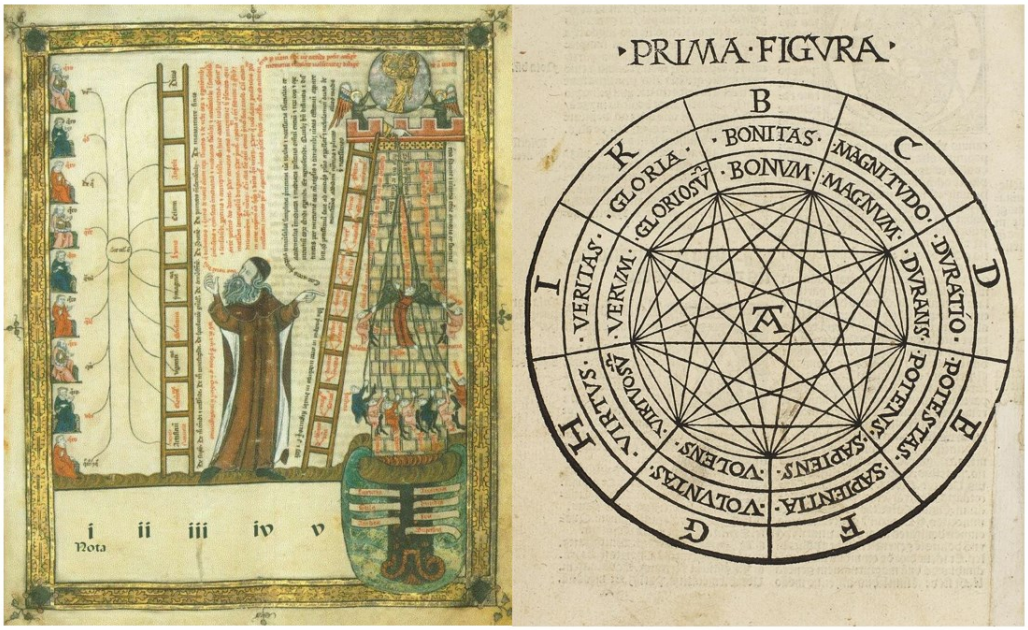

Fig. 1. A la dcha. el Ars Magna (o Ars Generalis Ultima [Arte General Último]), en el que sujetos y predicados (postulado que se quisiera probar) se organizan con cierto sentido para generar posturas positivas (ciertas) o negativas (falsas), a fin de sistematizar el conocimiento por medio de un cómputo (aunque mecánico) riguroso. A la izda. Ramón Llull observando a través de su "torre de confianza" a nueve filósofos haciendo preguntas de distinta índole; según Llull, los "principios" que sostiene la torre, cuando estén bien definidos y distinguidos mezclándose con reglas y cánones, harán que aparezca la verdad de los efectos. Se podrá ver la asombrosa analogía de lo mostrado con las redes neuronales artificiales.

Ya sobre las décadas de 1830 y 1840, Lady Ada Lovelace (cuyo nombre sirvió, de manera honorífica, para bautizar al lenguaje de programación orientado a objetos, concurrente y fuertemente tipado Ada) y Charles Babbage trabajaron acerca de la "máquina analítica", que se puede considerar actualmente como el primer ordenador mecánico de propósito general (se pretendía usar para automatizar ciertas operaciones relativas al campo de la matemática analítica). Este aparato sugirió algunos comentarios ciertamente adelantadas a la época, en los que se remarcaba (p. ej.) que el dispositivo no estaba pensado para "crear ideas", sino responder a órdenes de las que se sabe proporcionaría respuesta, es

decir, hacer disponible lo que ya se conocía (comentario al que muchos años más tarde Alan Turing se opondría, pues opinaba que sí que se puede programar la creación espontánea de ideas si se le ordena a la máquina que lo haga de manera impredecible y se le puede "enseñar" a ello [18]).

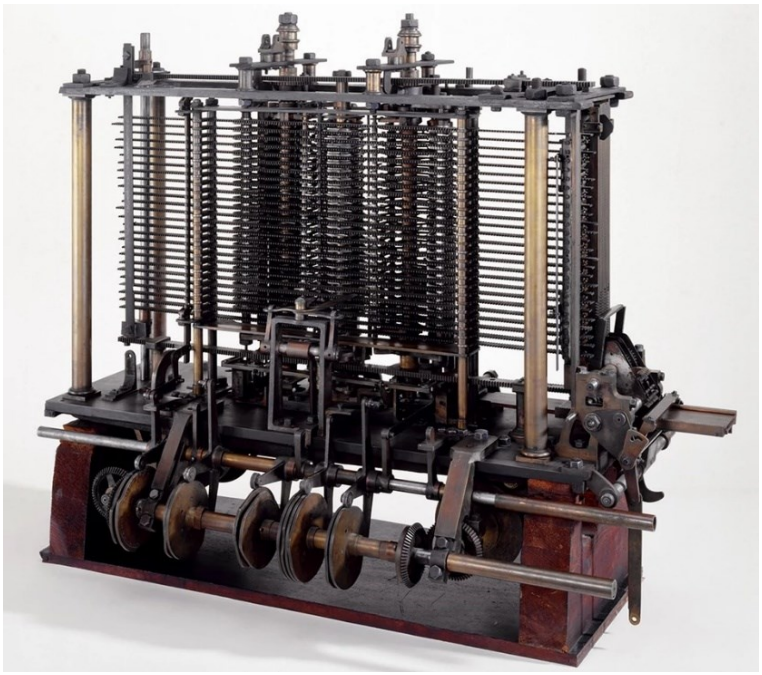

Fig. 2. Reproducción de un residuo componente de la máquina analítica de Charles Babbage y Lady Ada Lovelace, que en una visión fugaz de futuro se pretendía que "modificase su cálculo mientras estuviera funcionando" mediante una parada instantánea para redirigir sus posibles soluciones a una mejor convergencia, utilizando los valores que ya hubiera determinado para elegir entre dos posibles pasos subsecuentes (altísima semejanza con el aprendizaje autónomo).

Aunque en 1956, John McCarthy (quien contribuyó vastamente a esta rama del conocimiento con la computación en la nube, multiprogramación y su lenguaje LISP) acuñó responsablemente el término "inteligencia artificial", la IA ha crecido a partir de un cúmulo de reflexiones propuestas por tan múltiples como dispares ramas del saber, de las que cabe destacar la filosofía, psicología, neurociencia, economía (teorías de decisión, de juegos, procesos de Markov, control de la calidad total [TQC]…), ciencia de la computación, lógica matemática o formal e incluso lingüística (autores como Chomsky asentaron las bases de la lingüística moderna sobre la representación del conocimiento [19]); no tarda en vislumbrarse la importancia de esta última si se atienden a los intensos

trabajos de criptografía por parte del pionero de la "informática" Alan Turing, por ejemplo. Se puede decir que la IA cambió el paradigma tradicional al que se había habituado la ciencia de la computación; en donde antes se "juntaban" masivamente reglas y datos para proporcionar respuestas útiles se pasaron a juntar datos y respuestas para proporcionar reglas válidas, las cuales pudiesen servir incluso para aplicarse a nuevos datos y generar resultados sorprendentes (se evolucionó desde una formalización literal del contexto hacia un entrenamiento [mucho más] inteligente).

Tratar de emular el comportamiento de ciertos sistemas naturales o biológicos mediante algoritmia tradicional con IA simbólica (a decir, sistemas secuenciales, máquinas de estados finitos, tablas de búsqueda, codificación entrópica o aritmética, redes semánticas y lógica de predicados, booleana o máquinas de von Neumann…) resulta ser una tarea complicada si no se cuentan, por lo menos, con las herramientas adecuadas. Tratar de restar dificultad a la tarea de modelar estos escenarios equivale a realizar una búsqueda de métodos innovadores casualmente nunca contemplados que prioricen menguar las imprecisiones e incertidumbres típicas que aparecen en los problemas nacidos del mundo real (búsqueda y reconocimiento de patrones, identificación de imágenes y conjuntos, toma de decisiones, etc.); como se va a ir entendiendo a lo largo de los ulteriores párrafos, la experiencia rechaza aferrarse a los métodos tradicionales antiguamente predominantes con los que primaba la construcción explícita de conjuntos de reglas preprogramadas "suficientemente amplios" como para aproximar la virtud del conocimiento (IA simbólica), de suerte que realmente se desee hacer escalar profusa y conjuntamente la dificultad del problema a modelar o solucionar y "la capacidad de poder hacerlo" autónomamente.

Las décadas de los 40 y 50 fueron detonantes en cuanto al estudio científico de la computación motivada por el desarrollo de agentes racionales artificiales (coincidiendo y colaborando, causalmente, con los comienzos del estudio de las redes neuronales reales [20, 21]); véase el inicio de la cibernética neuronal (modelización de sistemas de conducta y automatismos a cualquier nivel semejables a estructuras celulares nerviosas) gracias a Ashby, Drischel, Wiener [22-24], la emulación del pensamiento humano y "método de loci" en el dispositivo de almacenamiento de datos Memex (1945) por parte de Vannevar

Bush, los estudios de Alan Turing (1950) y su test homónimo en "*Computing Machinery and Intelligence*", en parte motivados por el análisis del exacerbado y creciente número de datos relativos a la economía global y la cuestión de si una máquina puede presentar la cualidad de discernimiento, el nacimiento de los denominados *chatbots* surgido del énfasis en la redacción de sistemas de inteligencia artificial lingüísticos (robot ELIZA, 1955), el programa GPS (*General Problem Solver*) realizado por Newell y Simon creado con el propósito de separar la información empleada para definir el problema de la solución heurística generada para resolverlo (estaba limitado, pues *a priori* solamente podía resolver problemas simbólicos), etc.

El interés del humano en replicar su comportamiento (típicamente) racional y lógico en las máquinas computacionales se logró fundir con la sugestión producida por la compresión de los principios de funcionamiento del cerebro y su estructura; Feigenbaum, McCulloch o Rosenblatt [26-28] [25] fueron de capital importancia para establecer sólidas y revolucionarias ideas y teorías fundamentales con este respecto: analogías entre las redes neuronales biológicas y los circuitos lógicos digitales, el “perceptrón” como modelo matemático y computacional simplificado de una neurona, paradigmas y experimentos sobre el aprendizaje autónomo en la informática o computación y los sistemas expertos (como el analizador espectrométrico de masas Dendral, años 1960), etc., ya difícilmente adaptables a los métodos tradicionales, pero no a las prósperas nuevas vertientes.

A la vista de esta cobertura notablemente general sobre los acontecimientos que tejieron el entorno y marco de la inteligencia artificial, no debería de presentarse confusa la afirmación de que las redes neuronales artificiales, aunque se consideran un pilar fundamental de esta rama del conocimiento son, simultáneamente, tan modernas como antiguas. Es claro, además, que un sistema de aprendizaje automático no se “programa” de manera explícita, sino que más bien se entrena, pues se le presentan conjuntos de datos específicamente relacionados con un problema particular o una tarea y el sistema trata de encontrar en ellos un patrón estadístico que sirva para resolver (automáticamente) la tarea [29, pág. 29].

La primera aportación formal al estudio de las redes neuronales artificiales restringida a un campo denominado entonces como "cibernética" puede deberse al trabajo de McCulloch y Pitts, quienes realizaron la primera modelización descriptiva computacional simple, aislada y lineal de una neurona biológica (homónima) mediante circuitos electrónicos digitales y un estudio de sus posibilidades computacionales, que en esencia se pudieron ver como una implementación de funciones lógicas (tratando con valores lógicos o binarios) con las que se podían realizar clasificaciones simples de categorías linealmente separables mediante hiperplanos (subespacios afines de dimensión n - 1 dado un espacio afín de dimensión n) ajustando manualmente ciertos "pesos" fijos (los hiperplanos lineales son suficientes cuando la separabilidad es lineal, si no la cuestión se complica[1]) [20, 21]. Similarmente, a partir de los datos sobre la fisiología del cerebro conocidos en su época, en 1961 Donald Hebb [30] estableció la teoría antecesora del aprendizaje por refuerzo y no supervisado (que junto al supervisado componen tres grandes áreas de clasificación del aprendizaje automático) en aplicación a las redes neuronales, defensora de que la intensidad sináptica (peso) de un fenómeno neuronal útil se incrementa siempre que su entrada y salida se activen simultáneamente, desarrollando conexiones cada vez más fuertes (sincronía de activación y acción conjunta; asimismo se plantea para la transmisión de información por dos neuronas unidas entre sí). Esto sirvió *a posteriori* para estudiar y perfeccionar métodos de actualización de los pesos asociados a las neuronas constituyentes de una RNA ("aprendizaje hebbiano") [31], además la teoría fue tan relevante que supuso la posibilidad de idear mecanismos (apoyados en los mencionados fenómenos que están correlacionados positivamente [por tanto tienen relación directa] entre sí) para fortalecer los modelos de neuronas y sus redes [32].

[1] Véase el denominado "truco del kernel" de las máquinas de soporte vectorial para sobrellevar esta limitación, con el que se mapean los datos en un espacio de más dimensiones donde se trata de alcanzar una separabilidad lineal. Otra manera de resolverlo es más acorde a la esencia de este documento: asociar múltiples neuronas simples en una red (RNA) multicapa que, de suerte, pueda superar tal limitación.

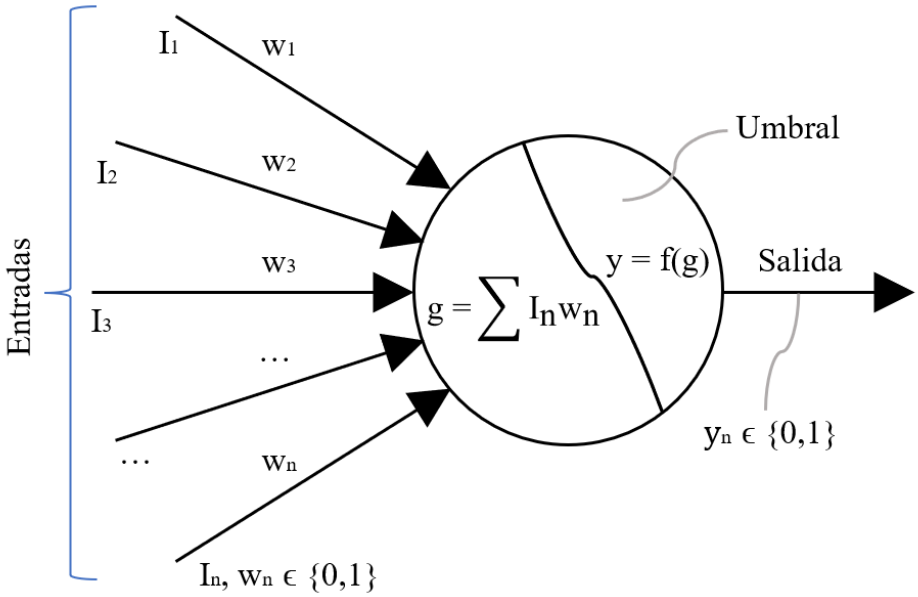

Fig. 3. Neurona de McCulloch-Pitts (M-P); primer modelo computacional de neurona propuesto, se divide en dos partes: la primera (izda.) recibe las entradas y las agrega "para tener cada una en cuenta" a través de cierto procesamiento amplificador o atenuador (o de los pesos [valores constantes] de cada conexión entrante), la segunda (dcha.) toma la decisión de "disparar" o no la salida bajo la influencia de la primera parte. Una primera capa de neuronas M-P podría ocuparse de localizar contornos generales en una imagen, una subsiguiente de los colores, luego una tercera de conglomerar las partes y finalmente la puerta de salida decidiría de qué objeto se trata el visualizado, etc. (procesamiento jerárquico). Rosenblatt y otros autores propondrían otros modelos más precisos de neuronas artificiales en los que pesos, valores umbrales e incluso conexiones se "aprenden" a lo largo del tiempo a partir de ejemplos.

Recientemente comentado, el perceptrón (nacido en 1958 de la mano de Rosenblatt como un esfuerzo por automatizar tareas intelectuales normalmente realizadas por humanos, concretamente reconocimiento visual de patrones sencillos [33]) se considera como un antecesor de muchas técnicas modernas de machine learning (ML) precedentemente englobadas dentro del aprendizaje supervisado que, aun habiendo sorprendido fugazmente a la comunidad científica en la resolución de multitud de problemas desde su creación, se trata de la primera y más simple modelización del cerebro humano. Algunos autores establecen tipológicamente al perceptrón, siendo un paradigma de técnica que "busca alejarse" de lo tradicional, como una de las primeras instancias dentro del tipo de representación "subsimbólica" o "conexionista" (opuesto a los métodos clásicos de la inteligencia artificial) ("un paso más allá" en la cibernética [34]) [30, 35], que trata de

desarrollar sistemas con capacidad de aprendizaje a nivel de individuo (algoritmos genéticos) o de especie (algoritmos evolutivos) en los que la información que se maneja es masiva, distorsionada e imprecisa y no simplemente representa de manera simbólica reglas explícitas manipulables de modo sucinto [36]. Dicho con otras palabras, es claro que para emular las funciones cognitivas de la inteligencia humana un sistema simbólico trasiega en el conjunto de enunciados lógicos contemplados en el momento de su ejecución, mas uno conexionista persigue aprender, adaptarse y organizarse autónomamente asemejándose a la configuración biológica del cerebro [37, pág. 105]. Ligado con esto, es interesante comentar que el perceptrón también se "nutre" sustancialmente de su salida (señal deseada) o de ejemplos de entradas de cada categoría a clasificar y lo aprovecha, característica que no era fundamental en las aproximaciones procedentes.

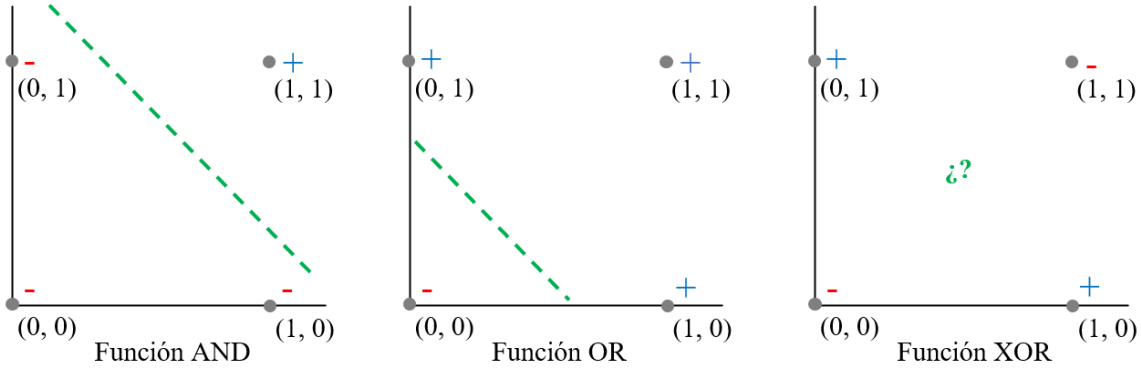

Fig. 4. Ejemplos de funciones lógicas cuyo comportamiento podían, o no, emular sistemas neuronales simples y primigenios como la neurona M-P o el perceptrón. Las dos primeras gráficas muestran visualmente la separabilidad lineal (un hiperplano lineal [en este caso] de separación delimita clara y determinando los dos "conjuntos" de datos) característica de las funciones AND (conjunción lógica) y OR (disyunción lógica), respectivamente, cualidad modelable por este tipo de sistemas inteligentes. En contraposición, la tercera gráfica muestra el clásico problema XOR, donde ningún hiperplano puede separar correctamente los conjuntos (no separabilidad lineal que tales sistemas neuronales soportarían pésimamente) (recogido de [38, pág. 610]).

Cabe mencionar que hoy día numerosos algoritmos comúnmente empleados en resoluciones de todo tipo de problemas de inteligencia artificial se basan en las formas más básicas y fundamentales de métodos subsimbólicos, como la optimización por colonia de hormigas (ACO) o cúmulo de partículas (PSO), los algoritmos de red inmune

(sistemas inmunes), recocido simulado (SA), búsqueda tabú (TS), etc. [39] [1, pág. 18]. Más aún, el término "sistema conexionista" puede hacer las veces de sinónimo de RNA, aunque algunos autores [6, pág. 18] remarquen una somera diferencia entre estos sistemas subsimbólicos (donde se realizan agrupaciones locales de sus nodos para representar conceptos) y las redes neuronales (donde cada nodo funciona corporativamente dentro de una distribución o agrupación general [en ambos casos cada nodo se compone de una neurona]).

Un perceptrón es, por tanto, una intento exitoso en la mejora de los antiguos modelos simples de neuronas (y otros) que sí consigue aprender del entorno y adaptarse a este, en cierta medida, para desarrollar capacidades de clasificación superiores; está formado por varias neuronas lineales que se suplen de un número arbitrario de entradas procedentes de sensores externos ("retina") y devuelven en su única capa de salida, bajo ciertas reglas de comportamiento y habiendo pasado a través de cierta área "de asociación", un resultado adecuado para distinguir si la entrada proporcionada pertenece a una de las dos clases que el modelo es capaz de discernir. Las entradas están ligadas a un número igual de pesos que influyen, ponderadamente, en la "relevancia" que se considere atribuir a cada entrada sobre la salida común. De esta forma, un parámetro umbral sirve para determinar si, según el valor de la ponderación en la suma acumulada de cada entrada por su peso ("el sumatorio ponderado de las características"), este resultado es mayor o menor que el valor con el que se define (la salida es cierta si tal suma es menor que el umbral y falsa en caso contrario [o viceversa]). El modelo de neurona artificial básica que se muestra en la siguiente figura (dcha.) es bastante (sino sumamente) similar a lo que se usa típicamente en las simulaciones de redes neuronales actuales.

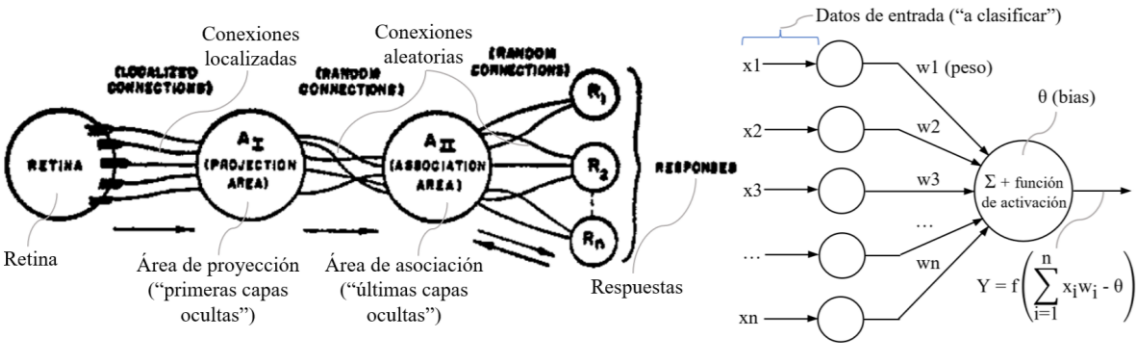

Fig. 5. El perceptrón clásico de Rosenblatt en su organización original (izda.) y la representación simplificada por Minsky y Papert en su versión actual (dcha.) (recogido de [33]); el nodo de salida del último realiza la suma ponderada de sus entradas, computa su diferencia con un desplazamiento (bias) y la hace pasar por una función de transferencia o de activación (de tipo escalón, sigmoide… [representación no lineal de una neurona]), a fin de responder positivamente (+1…) si la entrada se supuso perteneciente a una clase o negativamente (0, -1…) si lo fue a la dual.

Si un perceptrón representase al semejante de una neurona, una red multicapa de perceptrones (o perceptrón multinivel) se parangona con una red de más de una neurona. Así nace el concepto de capas de entrada, ocultas y de salida, pero el sistema se restringe aún al determinismo binario de sus entradas y salidas; cabe mencionar que la diferencia primordial entre el perceptrón y la neurona M-P es que en el primero los pesos pueden variar (se pueden ajustar de manera automática), mientras que en el otro caso son fijos. La labor de parametrizar los pesos y el valor umbral de cada neurona crece (si no exponencialmente) con el número de las que se componga la red, por lo que se intuye un motivado interés por empezar a agilizar y facilitar este proceso para poder plantearse la escalabilidad y evolución de estos algoritmos. También se columbra que la capacidad computacional disponible en aquella época puso ser una de las principales razones que mermaran el estudio y avance imperioso de estas técnicas, al menos hasta mediada la década de los 90 (si no terminada la de los 2000 [40]). La regla introducida por Rosenblatt que denota el carácter supervisado del aprendizaje autónomo de estas estructuras pseudointeligentes se resume en el hecho de que los pesos asociados a cada "neurona" de la "red neuronal" se actualizan en tanto en cuanto el error (diferencia entre la señal producida por el sistema y la de entrada) se hace más pequeño, lo que desemboca en que sus valores se recombinen sinápticamente y se dirijan, de suerte, hacia una convergencia

en la que se produzca la salida buscada, es decir, donde la red haya "aprendido lo suficiente".

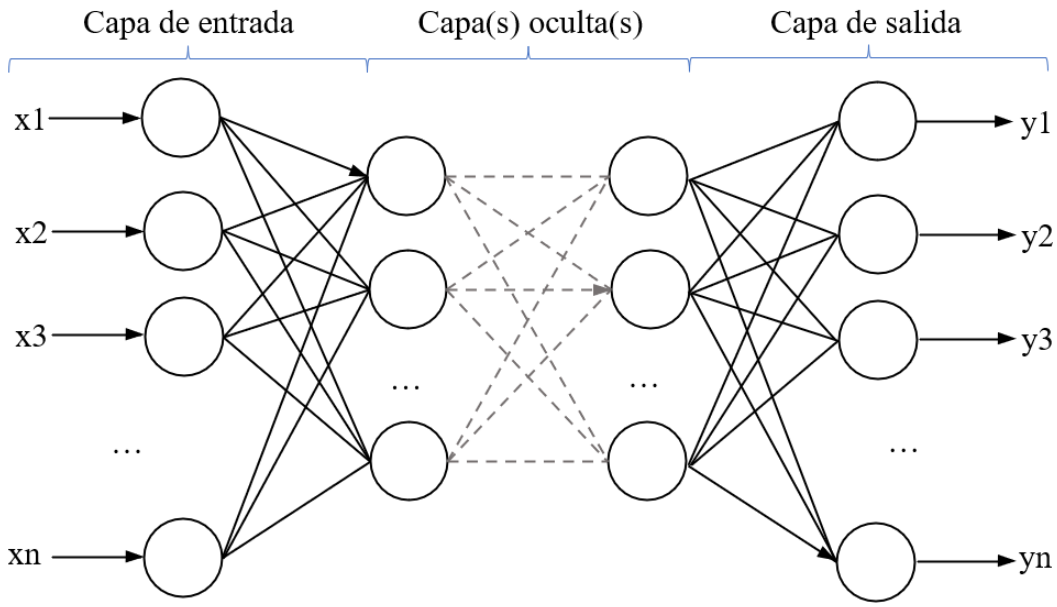

Fig. 6. El perceptrón multinivel de Rosenblatt; cuando el aprendizaje se ve mermado por los niveles ocultos del problema, ya que no proporcionan una salida conocida a priori, las redes de tipo feedforward como esta son una solución válida. Ahora la región de decisión (caracterizada por la estructura de las capas ocultas y su número, de acuerdo con la profundad de aprendizaje requerida) es mucho más completa (se simplifica el trazado de las conexiones del "grafo"; todas van "hacia delante" unidireccionalmente); se intuye que esta red de neuronas podría clasificar "n" clases.

El programa conexionista pareció proseguir creciendo prioritaria e incesantemente encabezando a los métodos tradicionales hasta que, poco tiempo después, comenzaron a aparecer problemas especialmente dificultosos de computar que aminoraron egoístamente las promesas de este campo de estudio, como muestran Minsky, Papert y otros autores [35]. En estos problemas se veían involucradas ciertas clases de funciones consideradas como no computables por los perceptrones, que tenían la cualidad común de no ser linealmente separables, como la función XOR (disyunción exclusiva lógica) y ciertos conjuntos de datos y patrones ambiguos.

Tras el discreto estancamiento mencionado [41], desde comienzos de los 80 surgen algoritmos y técnicas que no solo combinan coordinadamente múltiples perceptrones (o sus variantes) para formar una red conjunta, sino que lo hacen de maneras especialmente

inteligentes; basándose en estas técnicas fundamentales se desarrollaron a posteriori conceptos como funciones de propagación (para calcular cuál es "el valor total que recibe" cada neurona, dependiente de los pesos de cada una) y activación (sigmoides aplicadas a la salida de las neuronas) (en textos como [37, pág. 110] se enfatiza en diferenciar ambos conceptos mientras que en otros como [42, pág. 107] se opta por mezclarlos), la red neuronal ADALINE (que se puede considerar como la primera red en emplear la técnica estocástica de "descenso de gradiente" para su entrenamiento [algoritmo *Least Mean Square*, más específicamente]) y MADALINE (red multicapa compuesta por varias unidades ADALINE) (ambas tuvieron éxito como filtros de eco en señales telefónicas, ecualizadores de frecuencia como filtros adaptativos o implementaciones en circuitos VLSI) (Widrow y Hoff, 1959, 1980), la "representación distribuida" (captura de estructuras y factores estadísticos que explican la variación de los datos, abstracción de la información importante y el aumento en la eficiencia del proceso de entrenamiento) [43], el algoritmo de aprendizaje *backpropagation* (que junto con las funciones de activación permiten que, p. ej., los perceptrones multinivel superen holgadamente los problemas de clasificación causados por las funciones no separables linealmente) [44, 45], las redes simétricas [46], Hopfield [47] (con una arquitectura autoasociativa muy diferente de las anteriores que incluía nuevas reglas de aprendizaje, estableciendo semejanzas con las sinapsis simétricas y el modelo de Ising[2] de la física estadística [48]) y Kohonen [49, 50] (también sumamente especial y diferente del resto de redes antiguas, pues se basa en la hipotética capacidad inherente del cerebro humano en crear mapas topológicos de la información recibida desde el exterior, dotando a la red de una cualidad autoorganizativa), redes estocásticas (Máquina de Boltzmann y Cauchy…), de base radial (curiosa alternativa a los perceptrones multicapa o multinivel) [51] y de variable compleja [52, 53], aprendizaje por refuerzo [54], etc. Cabe mencionar

[2] Cuenta S. Alexander en [120] cómo el modelo de Hopfield (que ilustra el funcionamiento de la memoria asociativa) se relaciona con la física cuántica del magnetismo, concretamente con el modelo de Ising (modelo matemático que describe el ferromagnetismo); el fenómeno de interacción entre espines atómicos (el cual dicta que cuando la energía de interacción es elevada debido a espines discrepantes [cada átomo tiene un espín — cuya matemática trata de explicar cómo se comporta la energía de interacción entre los átomos— binario; si dos átomos tienen espines opuestos son discrepantes], los "pequeños campos magnéticos" generados por átomos individuales quedan anulados, lo que provoca que el metal no tenga ningún comportamiento magnético) condujo (a Hopfield) a su reinterpretación como un escenario de interacciones comunicativas entre neuronas cerebrales (una sorprendente e inquietante analogía).

que, como el perceptrón simple y multinivel, la mayoría de estas redes se denominan redes "hacia delante" o *feedforward* y se caracterizan por arquitecturas en niveles y conexiones entre neuronas dirigidas estrictamente hacia delante (no tienen ciclos o bucles, al contrario que las redes "recurrentes") (de acuerdo con [107], las redes convolucionales profundas han supuesto un gran avance en el procesamiento de imágenes, vídeo, voz y audio, mientras que las redes recurrentes han arrojado luz sobre datos secuenciales como texto y voz). Los arreglos que combinaban estos destacados modelos y sus teorías parecían superar a las redes simples de perceptrones en cuanto al número y tipo de problemas de aprendizaje que podían resolver, mas la dificultad de "configurar" o "sintonizar" adecuadamente las redes podía ser cuantiosamente elevada y, en ocasiones, signo de que el problema sería "demasiado costoso" de resolver mediante tales métodos (pero no mediante otros potentes procedimientos de clasificación como máquinas de soporte vectorial, clasificadores lineales [p. ej.] o sencillamente no basados en machine learning).

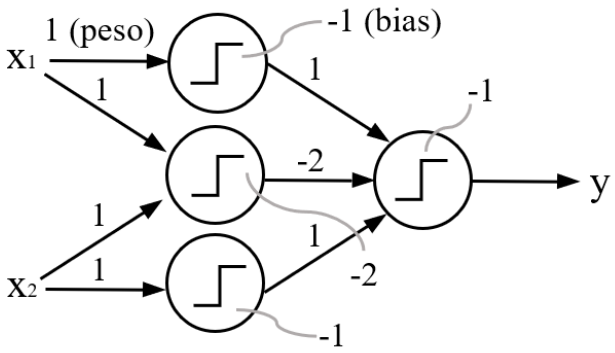

Fig. 7. Ejemplificación visual esquemática del MLP de Rosenblatt (red feedforward de dos capas ocultas); se ve cómo esta configuración tan simple de red neuronal resuelve el problema de la función lógica XOR. Este tipo de ANN sería uno de los principales antecesores de otras redes de aprendizaje profundo de gran repercusión, como las RNC. (adaptado de [38, pág. 610]).

En este punto temporal y de nuevo, la explotación (meramente tecnológica, pues se siguió profundizando en la teoría tanteando la modelización de sistemas neuronales concurrentemente con los avances neurobiológicos que inspirarían los esfuerzos precedentes [103, pág. 200]) de las redes neuronales artificiales de más alto alcance sufre (de manera general) un período de "estancamiento" que se alargaría casi décadas hasta

que, con la capacidad computacional más moderna (remarcada mejora en RAM y memorias no volátiles, GPU con numerosos núcleos…), aparecen de nuevo posibilidades mucho más accesibles (que antes) a la hora de crear, entrenar, desarrollar y validar redes neuronales artificiales funcionales de manera extendida y a una velocidad o rapidez sencillamente (más) practicable.

Es ciertamente remarcable (dado el propósito de este documento) una red neuronal artificial de aprendizaje no supervisado desarrollada por Kunihiko Fukushima entre 1978 y 1984 denominada "neocognitrón", que fue inspirada por el trabajo de Hubel y Wiesel años antes [55] y se ideó para el reconocimiento de caracteres (japoneses) manuscritos, cuya arquitectura jerárquica multicapa fue sugerida por hallazgos neurofisiológicos en el sistema visual de ciertos mamíferos. En este modelo se presentaron los novedosos conceptos de capa convolucional (reúnen sucesivamente información contenida en píxeles adyacentes [p. ej.]) (autoorganizada en un sistema jerárquico de capas) y de muestreo descendente (ayudan a una mejor clasificación de objetos incluso en escenas donde están desplazados [el neocognitrón es invariante a la traslación, rotación y escala]). Sus propios creadores propondrían mucho después algoritmos de aprendizaje supervisado y no supervisado para mejorar el entrenamiento de los pesos del neocognitrón [56]. Esta red serviría como avance para el desarrollo de las redes neuronales artificiales convolucionales (RNC, formal y primigeniamente presentadas en 1989 [*Neural Information Processing Workshop*] para el reconocimiento del habla y análisis en tiempo real de señales mediante su convolución en el tiempo [104]), primeramente con la técnica *max-pooling* ("computar sucesivamente el máximo de las activaciones sinápticas") en una red denominada como "cresceptrón" [57-59], luego mediante la incorporación de un método de aprendizaje "hacia atrás" (backpropagation, que recurrentemente se suele utilizar hoy día para actualizar todos los pesos realimentando el error en la salida) con descenso de gradiente más funciones de coste/pérdida u otras técnicas de optimización para entrenar más eficientemente al sistema [60], la ayuda inestimable de los trabajos de Hinton sobre el desarrollo de métodos más eficientes de entrenamiento de modelos de DL utilizando aprendizaje por capas [61] y posteriormente refinándolos e implementándolos en GPU para conseguir resultados y aprendizajes sorprendentes [62], entre otros.

El modelo biológico de las neuronas

Es manifiesto que el cerebro humano es el agente responsable de la inteligencia humana y parece lógico que los investigadores trataran de simular y plasmar sus funciones en las máquinas inteligentes, por eso también es el modelo de referencia principal de cuyo funcionamiento y estructura se ilustran los modelos de redes neuronales artificiales (u otros paradigmas de la IA como la lógica difusa, que permite la computación de variables expresadas con etiquetas del lenguaje natural) y de los que estas se intentan replicar, manteniendo en todo momento una analogía razonada con la neurona como elemento fundamental que es del cerebro (sistema nervioso), cuya clasificación rigurosa, entendiéndola como célula nerviosa en tipos, tamaños y funcionalidades es un apartado correspondiente a textos neuroanatomistas [63-65].

El cerebro humano (órgano más voluminoso del encéfalo) es una compleja, no lineal y paralela computadora biológica que no por pocos motivos (capacidad de memorizar, aprender, corregirse, computar paralelamente diversas tareas…) motivó intentos forzados de su modelaje, que hoy día siguen siendo aproximaciones quiméricas. El cerebro compromete la actividad combinada de, aproximadamente, cien mil millones de células nerviosas denominadas neuronas que eventualmente "mueren" y se auto(re)generan, cuyo modo de intercomunicación y "estrategia" sináptica es lo que determina la variabilidad de las capacidades humanas particulares y no (principalmente) su número absoluto. Por tanto, más interesantes parecen ser son los llamados "espacios sinápticos", que se tratan de los puntos de conexión (caminos de recepción y envío de información) entre diferentes neuronas, pudiendo estar una sola asociada a múltiples otras y viceversa, habiendo un total de, aproximadamente, cien a mil billones de estos en el cerebro (cien millones de millones). Más aún, en cada espacio sináptico se pueden dar lugar a miles de eventos en donde se involucran partículas, moléculas y fenómenos de distinta calaña como los neurotransmisores ("segundos mensajeros") [66], que son biomoléculas segregadas desde la neurona para transmitir información que sea captada por otras neuronas, células o glándulas mediante las sinapsis (siendo estas últimas conformemente definidas como estructuras que permiten transmitir impulsos eléctricos o señales químicas desde una

neurona [o célula nerviosa] a otra [o célula efectora] en el espacio que ocupa un espacio sináptico [aprox. 50 ÷ 200 Å —10^{-10} m—)]).

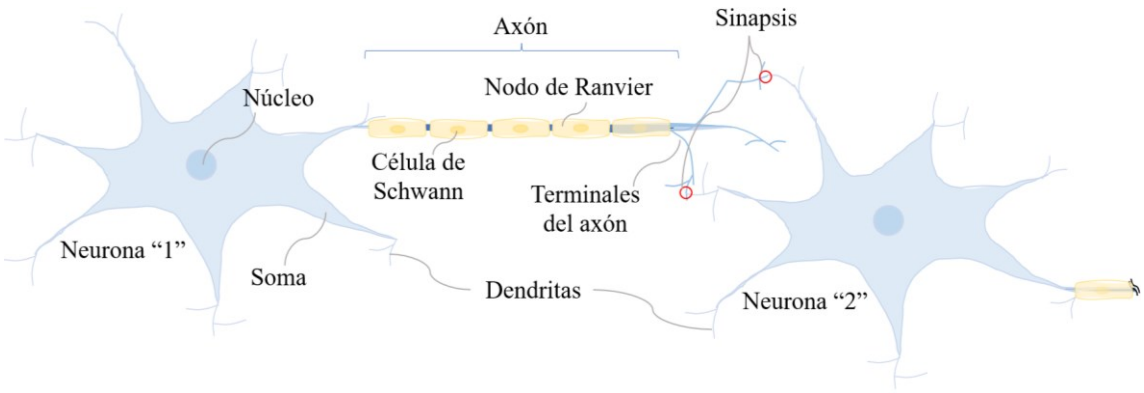

Fig. 8. Representación simplificada del modelo de una neurona biológica. La discusión y comprensión de distintos respectos sobre una neurona biológica no es nada fútil si se tiene como propósito comprender los intrincados mecanismos de las RNA.

Dado que muchas partes componentes de una neurona biológica presentan un relativo semejante en una neurona artificial (debe tenerse en cuenta que una neurona es, indudablemente, una célula viva [diferenciable de otras células por su cualidad de comunicación] semejante en su conjunto al nodo de una neurona artificial), pueden comentarse cualitativa, descriptiva y simplificadamente los elementos que se consideran de mayor interés:

- Dendrita y sinapsis: las dendritas son estructuras o redes arborescentes compuestas de fibras nerviosas; están conectadas al soma de la neurona como ramificaciones secundarias (pueden alcanzar los milímetros de longitud). Por otra parte, las sinapsis o placas sinápticas (como se ha comentado antes) son estructuras o contactos ciertamente especiales que tienen una función más bien receptora y permiten el traspaso de información entre neuronas. Las dendritas o sinapsis son semejables a las conexiones de entrada y los pesos de una neurona artificial, respectivamente.

- Soma o pericarion: es la estructura principal de que se compone de la neurona (esencialmente su cuerpo, de 5 ÷ 10 micrómetros de diámetro) y lleva consigo su núcleo (que contiene casi todo el genoma [información genética] de la célula). El

soma o pericarion puede ser semejable a los nodos (cada nodo sería una neurona biológica en su conjunto) de una RNA, especialmente a los de la capa de entrada.

- Axón: es una conexión única relativamente extensa (alargada, bifurcada en múltiples terminales finales), conectada al cuerpo de la neurona como ramificación principal y por la que se transmiten las señales nerviosas, desde la propia neurona y hasta otras (pueden alcanzar los cientos de milímetros de longitud). El axón es semejable a la salida de los nodos de una RNA.

Las dendritas de la neurona reciben señales de entrada provenientes de otras neuronas (potenciales postsinápticos), las cuales combinan y luego se "procesan" en el soma para transmitir señales de salida a través de su axón y sus terminales hacia otros conjuntos de neuronas; estas señales son tanto de naturaleza eléctrica (las transportadas a lo largo del axón) como química (las recibidas a través de las dendritas, mediante las sinapsis "colocadas" entre estas y los axones previos). A causa de la disparidad en la concentración de iones de sodio y potasio alternativamente entre el líquido que alberga el soma y el exterior, se produce una diferencia de potencial eléctrico entre cada medio de aprox. 70 mV (potencial en reposo) (negativo en el interior de la célula debido al exceso en electrones por efecto de la mayor concentración en iones de potasio [en comparación con su defecto en el exterior, donde predominan los iones de sodio]). Con la recepción de neurotransmisores a través de las dendritas se acumula potencial negativo en el exterior de la neurona, lo que causa que la permeabilidad de la membrana (vainas de mielina) que define la estructura celular de la neurona se altere una vez rebasado un valor límite y, en ese momento, se desate un flujo masivo de iones de sodio dirigido al interior, invirtiendo el signo de la diferencia de potencial (polaridad de la membrana) ("potencial de acción") y, tras décimas de milisegundo, volviendo a evolucionar hacia el estado de equilibrio con un flujo inverso de iones. Se establecen así, en función de la cantidad de neurotransmisores recibidos, trenes de impulsos que viajan entre neuronas a través de los axones, por consiguiente, se establece una comunicación (de una función ponderada de los valores o estímulos de entrada, como en las RNA) desde los terminales del axón a las dendritas de otras neuronas.

Por añadidura, existen dos tipos de sinapsis, excitadoras (sus neurotransmisores aumentan negativamente el potencial eléctrico de la membrana, aumentando la velocidad de generación de impulsos) e inhibidoras (sus neurotransmisores compensan el desequilibrio de la membrana acercando su potencial al de reposo); la suma de los efectos excitadores o inhibidores a los que se ve sometida una neurona provoca que esta se active (emita un tren de impulsos a cierta velocidad o frecuencia) o permanezca en reposo. Estableciendo una analogía, se puede decir que los neurotransmisores presinápticos serían las entradas de la neurona artificial y la acumulación de los efectos debidos a la naturaleza de la sinapsis existente (que tiene su propio "peso") sería la ponderación de los pesos asociados a cada nodo (se amplifican o atenúan las señales de entrada, pues una señal de entrada puede ponderar para activar una neurona [sinapsis excitadora; peso positivo] o desactivarla [sinapsis inhibidora; peso negativo]). Es decir, si la suma ponderada de las entradas por sus pesos ("de la información proveniente de sinapsis que pueden ser excitadoras o inhibidoras" [casi todas las neuronas biológicas reciben neurotransmisores de ambas]) es mayor o igual que un umbral definido para tal neurona artificial entonces esta se activa (proporciona una salida cierta), así que se vislumbra un comportamiento binario (todo/nada) tanto en el modelo nervioso biológico como en el artificial.

Como cabe suponer, la operación de las sinapsis es susceptible de factores externos (que incrementen o decrementen el peso ["huella"] que define su comportamiento y por tanto el grado de activación de las neuronas [aunque estas huellas se puedan mantener a largo plazo por virtud de la memoria —aprendizaje—]) y también internos, pues los mecanismos de aprendizaje regulan el comportamiento de las redes neuronales a través de modelos equivalentes a funciones umbral que comprometen la salida con la "energía" de sus entradas y su estado de activación actual. Téngase en mente que hay muchos detalles relativos al modelo biológico de las redes neuronales y funcionamiento del cerebro que se ignoran en las RNA (al menos tradicionales), por ejemplo, no se tienen en cuenta los fenómenos temporales y de propagación espacial (a lo largo del axón) asociados a las señales neurotransmisoras.

Existen, por añadidura, tres tipos de grupos neuronales: receptores (permiten que entre al cerebro la información sensorial), de salida (transmiten las señales eléctricas desde sus

axones a los órganos actuadores a través de sinapsis especiales) e intermedias (transforman la información obtenida de los receptores [que son como transductores eléctricos acoplados a los sentidos] y forman las señales que controlan a los actuadores [que son como transductores "mecánicos"]); juntos conforman el sistema nervioso central. Las neuronas de la capa de entrada, oculta y de salida de las RNA establecen una analogía directa con estos conceptos; por ejemplo, de la misma forma que las neuronas de salida desembocan en células musculares para producir movimiento, las neuronas artificiales más sofisticadas en vez de devolver valores ciertos o falsos pueden desatar la ejecución de un programa que produzca acciones reales directas.

Los investigadores han esperado siempre, por lo general, encontrar un "patrón" simple que unifique las complejas y diversas observaciones que atañen al desglose de la estructura bioquímica del cerebro humano; no es casualidad que las máquinas de computación, al igual que el cerebro, "trabajen" con señales eléctricas, se compongan de elementos con cierta lógica propia y ejecuten tareas, de alguna manera, computacionales. De todas formas, ambos modelos son particularmente adecuados (y están especialmente optimizados) para resolver tareas o problemas de naturaleza perceptiblemente diferente; así pues, para que las redes neuronales artificiales se acerquen asintóticamente al cerebro humano, en términos de emulación de la inteligencia, es aún vitalmente precisada la comprensión profusa de la organización y el funcionamiento del susodicho [103].

Es interesante, para concluir este apartado, atender brevemente al estado del arte de las redes neuronales en el campo de la biomedicina; merece la pena mencionar un estudio [113] que tiene una particularidad especial con respecto de otros similares y de vanguardia, pues fusiona sorprendentemente ideas pertenecientes a los campos de la visión e inteligencia artificial, además, así se introduce más precisamente el hilo conductor de este trabajo, que lo conforman las redes neuronales convolucionales. Las RNC, como se comenta frecuentemente a lo largo de este documento, tienen una arquitectura que se puede aprovechar eficaz y eficientemente para aplicaciones en el ámbito de procesamiento de imágenes, pues estas redes aprovechan la información espacial y las mismas y son idóneas para tareas de clasificación. Su estructura no solo sirve un propósito principal, sino que está inspirada en datos biológicos extraídos de

experimentos biológicos realizados estudiando el córtex visual, que (como bien se aprovechó para desarrollar el neocognitrón) se divide en múltiples niveles donde cada uno reconoce cierta información de manera jerárquica o estructurada; por ejemplo, primero se distinguen píxeles individuales, luego se reconocen formas geométricas sencillas y finalmente elementos más sofisticados como caras, cuerpos otras entidades abstractas. En el trabajo citado se propone el desarrollo de un modelo de segmentación automática versátil de células cancerígenas una red neuronal convolucional (concretamente un modelo U-Net, que es un tipo de RNC desarrollada expresamente desarrollado para la segmentación de imágenes biomédicas [115]), emulando así tareas englobadas dentro del alcance típico de la visión artificial y velando por una robustez que podría no alcanzarse empleando otros métodos tradicionales. Se ve, por tanto, cómo la arquitectura de las RNC puede ampliarse, componerse y (re)combinarse para resolver tareas tan complejas como clasificación (cómo no), segmentación semántica, localización, detección de objetos o segmentación de instancias [114], que se podrían llevar a cabo independientemente mediante técnicas y algoritmos de visión por computador “no inteligentes” (presumiblemente más restrictivos y condicionantes en lo relativo al complejo “artificio” de métodos interdependientes con el problema particular); esto se retoma someramente en el último subapartado del antepenúltimo apartado.

Las redes neuronales artificiales convolucionales

Habiendo revisado superficialmente algunos de los tipos de redes y conceptos relacionados con esta sección de la IA más importantes histórica (por su difusión) y científicamente (por su evolución) (incluso pincelado el modelo biológico de las neuronas), se considera altamente interesante conocer cómo enmarcar las redes neuronales dentro de una clasificación general aunque restringida, a fin de establecer un marco de referencia conciso e ilustrativo para centrarse en un tipo específico en los siguientes apartados (consultar lecturas recomendadas para distintos puntos de vista sobre esta clasificación y más detalles [17, pág 40] [67, 68]). Se puede hablar de redes neuronales monocapa (redes Hopfield…), multicapa feedforward (redes backpropagation…), temporales (redes recurrentes simples de Jordan y Elman, TDNN…), autoorganizadas (redes Kohonen…), recurrentes (las entradas no van directamente a la salida, sino que pueden escindirse entre neuronas de entrada), LSTM o de memoria a corto y largo plazo (de las más importantes RNA recurrentes, tratan de simular cierta memoria en cada neurona, contenedora de información ligada a las entradas [se usan cuando, p. ej., se deben tener en cuenta no solo valores actuales sino también pasados]), profundas (informalmente referidas a las que presentan sobre 10, 30 o más capas ocultas) supervisadas y no supervisadas más la combinación de ambas, entre muchas otras.

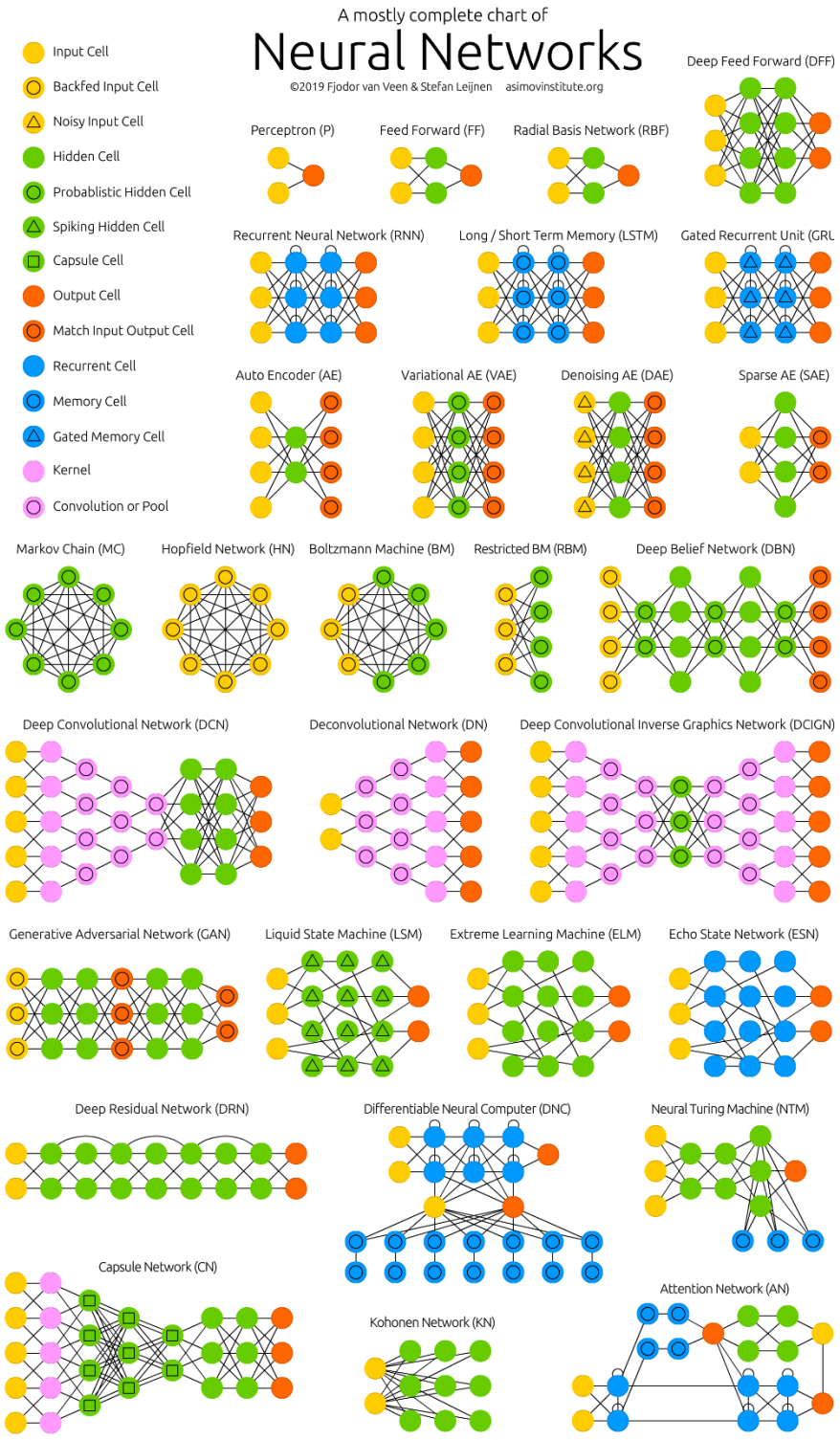

Fig. 9. Taxonomía general de las RNA[3].

[3] Recogido del sitio web https://www.asimovinstitute.org/neural-network-zoo/ (Fjodor van Veen, 2016, *The Asimov Institute*).

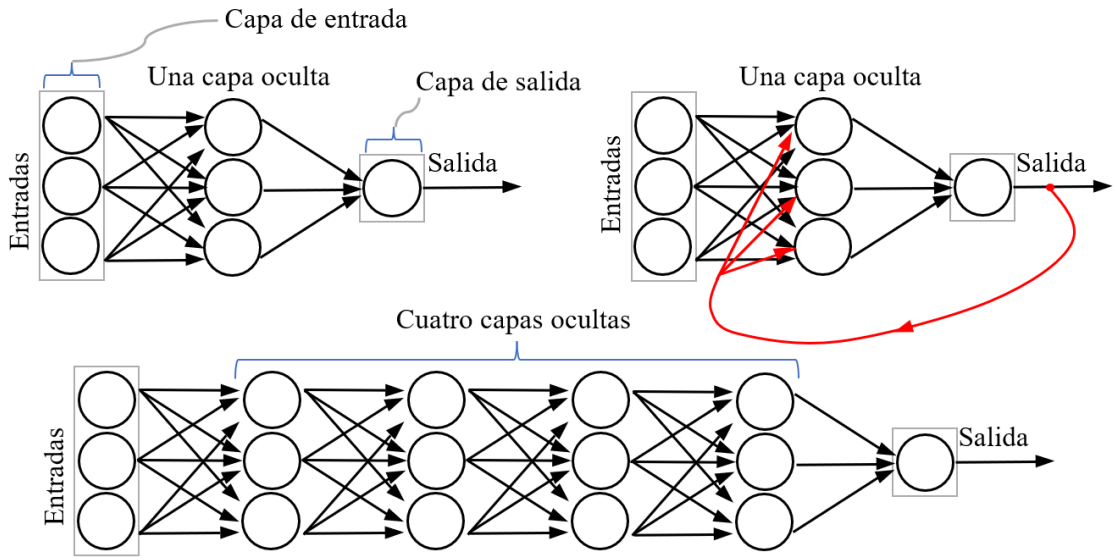

Fig. 10. Representación esquemática y simplificada de algunos de los tipos básicos de RNA típicamente utilizados: red feedforward de una capa oculta (tres capas en total) (arriba a la izda.), recurrente de una capa oculta (arriba a la dcha.) y convolucional de cuatro capas ocultas (densas o totalmente conectadas) (abajo) (adaptado de [38, pág. 610]).

A la neurona artificial ingresa un vector de señales de entrada que se supone son salidas de otras neuronas, además cada una de estas señales se multiplica por el peso correspondiente a su conexión con cada nodo ("eficacia de la sinapsis"; peso positivos para conexiones de excitación y negativo para inhibición) y todas ellas se dirigen al "cuerpo" de la neurona artificial, más exactamente un sumatorio o integrador donde se realiza la ponderación acumulativa y se computa el nivel de excitación correspondiente. Seguidamente, una función no lineal (*piecewise-linear*, sigmoide, tangente hiperbólica, ReLu [y sus distintos tipos], *softmax*, etc.) decide el nivel de activación final de la neurona en base a su nivel de activación actual y un desplazamiento (*bias*) constante semejable al potencial de activación o escalón que ocurría en cierto instante de tiempo en algún punto de la membrana biológica y que tiene el efecto de incrementar o disminuir la entrada a la función de activación mediante la realización de una transformación afín al vector de entradas (una transformación lineal más una traslación, geométricamente).

Toda red neuronal artificial, de modo similar a como están estructuradas las neuronas biológicas, está constituida de elementos bien diferenciados [37]: una neurona (N_i) es su elemento básico, una capa (C_i) es la agrupación de varias neuronas que rinden un mismo propósito (el de la capa) y realizan una misma función, una red (R_i) agrupa distintas capas conectadas entre sí y, finalmente, un sistema (S_i) agrupa distintas redes neuronales.

De acuerdo con [69] las RNA son "redes de elementos simples (usualmente adaptativos) interconectados masiva y paralelamente con organización jerárquica, las cuales tratan de interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico"; por otra parte, en [70] se propone una definición similar: "[…] un sistema de computación constituido por un gran número de elementos simples, elementos de proceso altamente interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas". Definitivamente una red neuronal artificial es, teóricamente y como se presupone desde hace largo rato, una red de capas de neuronas artificiales anidadas; la RNA más simple consiste en una capa de entrada, otra oculta y una más de salida, de forma que las neuronas de una capa pueden estas enlazadas con una o varias neuronas de la capas subsiguiente de manera no necesariamente biyectiva (aunque también pueden atenderse a configuraciones retroalimentadas en las que se consideren conexiones bidireccionales).

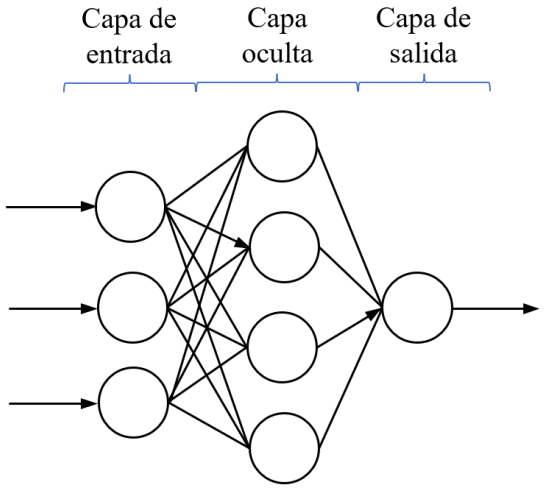

Fig. 11. Red neuronal artificial en su forma más simple. Cada nodo representado se corresponde con una neurona de cierta capa de las que se compone la red; en la práctica la estructura, arquitectura y el funcionamiento de las RNA son arbitrariamente más complejos de lo que se extrae de esta figura.

Una clase de redes neuronales que funcionan de manera ciertamente distinta a las tradicionales son las redes neuronales convolucionales, pues incorporan técnicas de aprendizaje profundo (aprendizaje automático en donde se mejora el rendimiento proporcionalmente al número de datos que se aprenden debido a su mayor número de capas) que se emplean típicamente para el reconocimiento y la clasificación de imágenes (son redes con "más experiencia"). De manera completamente análoga al resto de redes neuronales, también se basan en procesos biológicos en los que se involucra la actividad del cerebro; las RNC modelan el fenómeno jerárquico que tiene lugar, por ejemplo, al observar un objeto de la naturaleza: las neuronas corticales (las células nerviosas de los dos hemisferios de la corteza cerebral) de la corteza visual de un ser vivo con sentido de la vista (típicamente mamífero [56]) se activan individualmente de forma primera al recibir la luz procedente del objeto al que se mira, estando cada una de estas neuronas asociada a una de ciertas capas superpuestas a diferentes profundidades en la retina. De manera similar, una RNC consta de una capa de entrada y otra de salida más varias capas ocultas intermedias, mediadoras de la información que recibe "del exterior" la red y de cada cual se obtiene un nivel de complejidad semántica distinto al de las anteriores. A medida que un modelo se hace más profundo se pueden encontrar características más complejas, abstractas y distintivas, aunque tiende a producirse un sobreajuste cuando simplemente [sin atender a modificar razonadamente la estructura de la red a medida que se escalan los datos de entrada] se añaden parámetros al modelo haciéndolo más profundo [101, pág. 6].

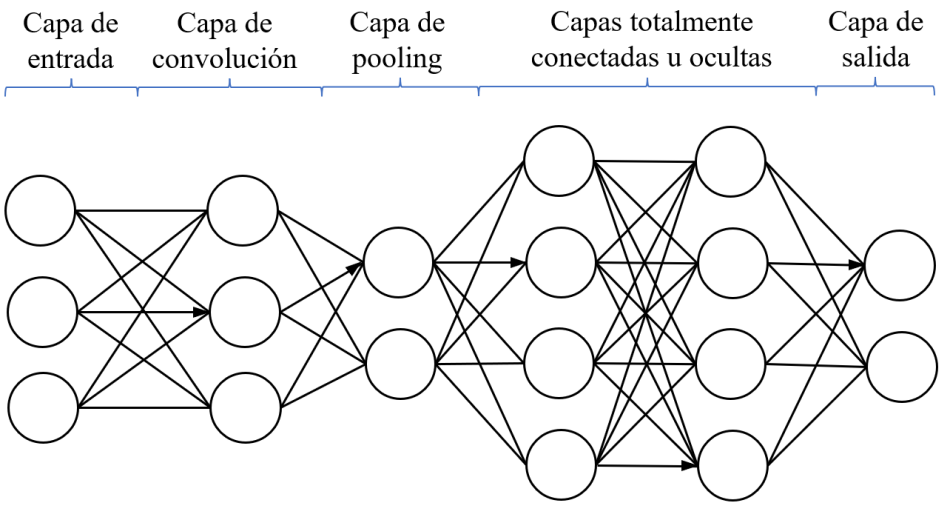

Fig. 12. Ejemplificación estructural arquetípica de una RNC, que "no es más" que una RNA compleja, es decir, que tiene más de una capa oculta. Por ejemplo: a partir de una imagen de entrada, se crean mapas de características mediante convolución y se redimensionan mediante operaciones de max-pooling, reduciendo la resolución pero profundizando en los mapas de características hasta que llegan a un clasificador (capas densas más capa de salida), una de las piedras angulares de la arquitectura.

A lo sumo, se puede decir que las RNC se diferencian de otros tipos de RNA en que en vez de focalizarse en la entidad total del problema se centran en explotar las características más determinativas del tipo de entradas que el problema proporciona, dando lugar a estructuras de redes más simples pero no necesariamente menos precisas [71]. Las capas necesarias para construir una arquitectura de RNC pueden tomar numerosas formas de acuerdo con los requisitos y características que son condicionados por el problema aunque, en base a la experiencia, existen algunos tipos concretos (algunos anteriormente mencionados) a los que se suele recurrir para tareas específicas casi siempre dentro del procesamiento y análisis de imagen, visión por computador o NLP (más otros campos específicos y extrapolables a problemas de índole industrial) [72-74]; particularmente, las RNC son ideales para estas tareas debido a que tienen capas que convolucionan sus entradas para "dar una idea" a la siguiente capa de en qué medida la función con que convolucionan a la entrada afecta a la misma. La convolución (operación matemática entre matrices), en contraposición con la simple y limitada multiplicación entre matrices que utilizan otras RNA, otorga la posibilidad a estas redes de poseer, desde muy pronto y paulatinamente, un conocimiento general de la manera en que se agrupan los píxeles, contornos u otras características determinantes de las imágenes (obtención de

características abstractas más "estrechas" a medida que la información se propaga a través de capas ocultas cada vez más profundas), consiguiendo tareas que otras ANN no son capaces de llevar a cabo, pues el supuesto más importante sobre los problemas que resuelven las RNC es la independencia de la localización del objeto en el entorno en que se presenta (por ejemplo, en una aplicación de detección de rostros no debe preocupar la situación espacial de estos en la imagen, sino simplemente su detección independientemente de la ubicación). Sin embargo, para un problema que sea suficientemente abordable por redes poco profundas puede ser igual de recomendable, o más, resolverlo mediante tales de estructuras y no recurriendo a otras con cantidades sobredimensionadas de capas.

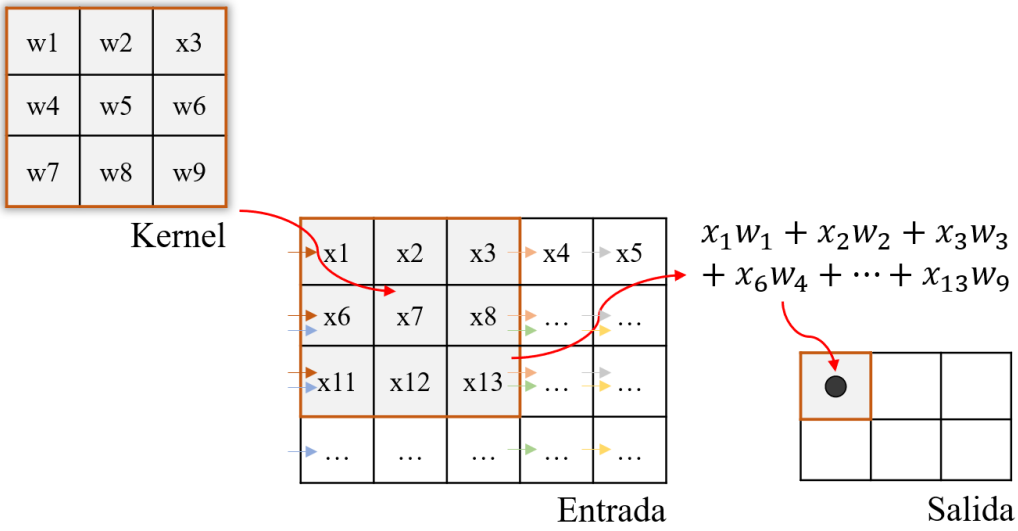

Fig. 13. Ejemplo general de la operación de convolución que implementan las capas convolucionales (las capas "ocultas" más importante de una RNC). Las matrices de convolución (inicializadas de antemano dando valores a sus pesos) se entienden como filtros de un tamaño prefijado que se convolucionan con los datos de entrada de cada capa convolucional (proporcionando matrices de menor tamaño a su salida). El tamaño de los filtros y el número de neuronas (y de filtros) que incorporan son algunos parámetros secundarios que determinan la estructura y función de estas capas e incluso de la red en su conjunto (adaptado de [83, pág. 177]).

Retomando el concepto teórico de las RNC, se puede decir compendiosamente que su capa de entrada recibe vectores de características procedentes del entorno de la red, la capa de salida emite una predicción también al entorno (una única capa de entrada y salida, como las redes feedforward) y las capas "ocultas" (generalmente hablando;

número variable) se corresponden con los conjuntos de neuronas que están interconectadas en el interior de lo que abarcan los otros dos tipos de capas y (sus pesos y configuraciones) definen la transformación que sufre la información que pasa "de capa a capa" [72]. Ciertas capas "no totalmente interconectadas" (capas convolucionales) en las que algunas de sus neuronas no reciben conexiones entrantes de todas las neuronas de la capa anterior, sino de un número limitado de estas, favorecen que cada neurona pueda "especializarse" en una región o un subconjunto restringido de los datos de entrada y se reduzca notablemente el número de operaciones a realizar en el proceso de cómputo. Por ejemplo, una RNC dedicada a una aplicación de reconocimiento de objetos puede tener varias capas iniciales ocupadas de detectar contornos, después algunas otras para buscar formas a partir de los contornos, luego algunas más centradas en atender a la posición de esas formas y, finalmente, capas (más la de salida) que combinen de manera lógica el resultado de todas las anteriores para proporcionar una predicción muy concreta (conjunción de múltiples predicciones sencillas para logar tareas más complejas velando por una constante escalabilidad en los datos de entrada).

El ajuste de los parámetros principales de la red (las variables "más principales" cuyo valor se estima durante el proceso de entrenamiento con los conjuntos de datos o entradas), que son pesos y bias, se realiza típicamente mediante backpropagation [75], pero la existencia de hiperparámetros (variables que influyen en el comportamiento de las posibles configuraciones de la red durante el proceso de entrenamiento; no son explícitos a los datos pero sí modelables) como el número de capas ocultas, de neuronas en cada capa, la función de activación para cualquiera de las anteriores, la cantidad de épocas (el número que determina el máximo de veces que se "permite" actualizar los pesos de la red), el tamaño de las entradas o su formato, la tasa de aprendizaje (LR) para el algoritmo de retropropagación… hacen que sea ineludible ajustarlos con tal de optimizar el rendimiento de la red; así también pueden definirse métricas de validación de referencia, técnicas empíricas, basadas en la experiencia… para llegar a buen puerto en la selección de valores óptimos para estas variables.

Las RNC más complejas tienen realmente capas a niveles de abstracción más profundos y detallistas, incluyendo capas convolucionales, totalmente conectadas (densas, *fully-*

connected o también denominadas, directamente, ocultas) (estas dos tienen parámetros directamente ajustables mediante aprendizaje), *dropout*, *batch normalization*, de activación, no linealidad o *pooling*, entre otras. Por otra parte, el hecho de recurrir a la convolución para conectar las capas se debe a que la cantidad de variables involucradas en la mera conexión de un nodo a una entrada que sea una imagen que tenga (p. ej.) 256 píxeles en cada dimensión y una profundidad de 3 canales RGB se eleva a un total de 196608 (el número de pesos asociados a la conexión de cada bit de información con la neurona) (la RNC debería tener una forma de entrada de 256x256x3), así que buscando un método más eficiente para procesar los datos se llegó a la conclusión (entre otras simplificaciones) de que conviene analizar regiones locales de la imagen, en lugar de la escena general de forma "cruda" [76-80], además de limitar el número de conexiones de las neuronas que una capa recoge de las previas o fijar ciertos pesos de una capa o a la siguiente. Esto equivale a que las capas (especialmente las ocultas) solo reciben la información (local) que ha sido anteriormente procesada por las capas precedentes, de suerte que el número de conexiones a las que se permitan conectar secuencialmente algunas capas decrezca con la profundidad y desde un valor razonablemente bajo. Los beneficios de tales simplificaciones son reducir drásticamente el número de parámetros de la red (a causa de la compartición de conexiones y pesos entre neuronas) o, más interesante, suponer que el hecho de fijar los pesos de las conexiones asociadas a una ventana local de la entrada es analizar esa ventana y mapear en ella los resultados de la red, estableciendo así una total independencia del reconocimiento de patrones de interés circunscritos (en relación con su posición) con el resto de la imagen [81, 82].

Así, se puede ver la interesante analogía entre las matrices de convolución[4] [83, 205] en el campo del procesamiento de imagen o el filtrado digital de señales (*kernel*) y los pesos de las RNC que se fijan inicialmente a fin de analizar "ventanas" de información restringidas contenidas en los datos de entrada. Para "personalizar" estas redes y adecuar su estructura a una tarea particular se pueden asociar múltiples capas a sendos filtros,

[4] Es curioso mencionar que ciertos algoritmos o librerías, al ser preferible a la hora de implementarse en *software*, emplean la operación matemática de correlación (cruzada) en vez de la convolución, pues es lo mismo pero sin "darle la vuelta" al kernel (la convolución de las RNC es someramente distinta a la "estándar"). Sitios web de interés: https://setosa.io/ev/image-kernels/, https://zhang-yang.medium.com/convolutional-neural-networks-conv-cnn-correlation-or-convolution-5840b91c46a6.

donde cada uno se ocupe de extraer diferentes características de la entrada (incluso analizando o atendiendo a la misma zona de la imagen) (las matrices de convolución son máscaras especificadas mediante filtros, cuyos coeficientes se deben aprender durante el entrenamiento). En [100-102][5] se pueden ver ejemplificaciones visuales convoluciones bidimensionales entre filtro e imagen que se dan en las capas de convolución de una RNC (así como arquitecturas internas típicas muy interesantes), donde se extraen características a través de varios niveles de representación gracias de las distintas capas ocultas de este tipo de redes profundas.

Una técnica comúnmente implementada en las RNC denominada *stride* permite no solo reducir progresivamente los parámetros ajustables de la red a medida que evoluciona el procesamiento de información y aprendizaje, sino mermar en la medida de los posible diversos efectos secundarios. En una operación de convolución tradicional, la matriz de convolución se desplaza a lo largo de la imagen "píxel a píxel" (conexión a conexión), pero también es posible realizar "saltos" mayores que resulten en la obtención de imágenes, a la salida, de menor tamaño que a la entrada, lo que reduce la cantidad de datos a procesar computacionalmente entre una capa y otra, controlando el solapamiento en los datos de entrada de nodos vecinos, no necesariamente menoscabando la solución.

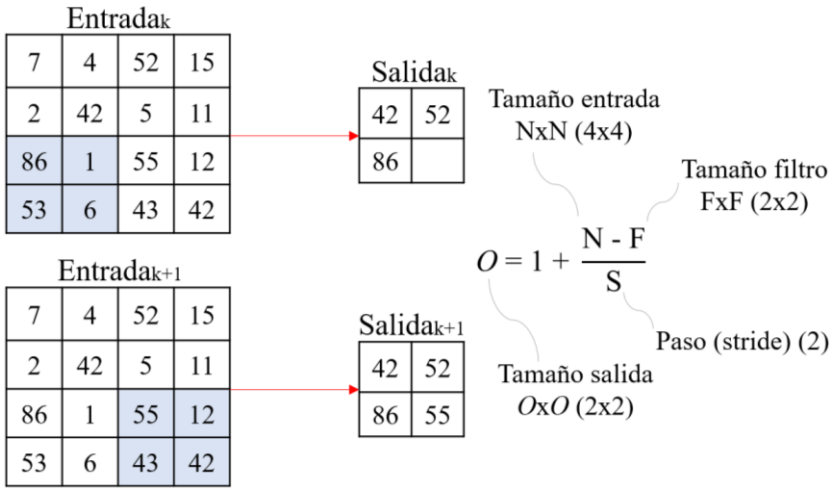

Fig. 14. Muestra visual de la técnica de stride junto con la de max-pooling (en vez de convolución; ver siguientes párrafos), en cierto instante "k" anterior y "k+1" posterior. A la derecha se ve la ecuación que formaliza esto matemáticamente.

[5] Además del sitio web https://numerentur.org/convolucionales/.

Otra conocida técnica es la llamada *padding* (semejable a lo utilizado en el procesamiento digital de señal), que solventa sencilla pero eficazmente problemas que puedan aparecer al procesar los bordes de una imagen gracias a la agregación de píxeles de valor nulo alrededor de la misma (*zero-padding*), por ejemplo, además de que también puede servir para mantener el tamaño de la imagen de entrada y salida a cada capa, lo que es útil para evitar el fenómeno de que el tamaño de los datos disminuya entre cada iteración (de manera descontrolada) o con la profundidad de la red (en esto se vislumbra el hecho de que es posible que las redes profundas [como las RNC] tengan cualquier número de capas ocultas).

Un tercer concepto de suma importancia, y con el que se asocia otro tipo de capas de las RNC (homónimas), es el *pooling*, que supone la reducción en el muestreo (submuestreo) de datos de entrada para agilizar la computación en capas posteriores y reducir su complejidad (algo similar a reducir la resolución de la imagen que sirve de entrada a las neuronas). El método más utilizado de pooling es el llamado max-pooling (y usualmente con un tamaño de 2x2, con un paso o stride también de 2 si se quiere evitar el solapamiento [la eficacia puede mejorarse caracterizando de manera particular la red en función del problema tratado]), con el que se divide la imagen de partida en múltiples subregiones y se conserva únicamente el valor más alto de cada una de estas, para formar una imagen de menor resolución (un vector de entradas de menos elementos) pero que reúne las características más significativas de cada región. Dado que el submuestreo provoca que no se preserve la posición exacta de los datos de entrada, es importante que no se implemente (a no ser que se haga deliberadamente) si la característica importante es la posición de los objetos en la escena y no sencillamente su presencia. En conclusión, en la operación de pooling que implementan las capas homónimas es mandatorio determinar el tamaño de la región sobre la que se quiere aplicar esta operación y el tipo de submuestreo a realizar (max/min/average-pooling…), pudiendo reducir el coste computacional de subsiguientes capas, el número de parámetros a determinar u optimizar, otorgar cierto grado de invarianza a la traslación y ayudar a controlar el sobreajuste de los (hiper)parámetros de la red.

También suele existir una primera capa denominada *flatten* (totalmente conectada), cuya labor es convertir la salida (su entrada) tras las etapas de convolución, pooling y padding en un vector unidimensional que iría directamente conectado a la primera de las siguientes capas totalmente conectadas del modelo de red, ocupadas íntegramente de la determinante clasificación final. El hecho de que estas ideas puedan aplicarse no solo a imágenes concretas, sino a conjuntos de datos de distintas dimensiones como series temporales o señales de audio, imágenes RGBA, vídeos (mediante convoluciones unidimensionales y tridimensionales) y conjuntos de datos arbitrarios de más dimensiones para aplicaciones estadísticas…, hace que las RNC sean especialmente codiciables en el campo de la visión artificial y reconocimiento de patrones, clasificación de objetos y etiquetado de imágenes, entre otros.

Tras la presencia de las capas que se "ocupan" de la convolución de los datos de entrada, aparece la figura de otros tipo de capas denominadas "no lineales", empleadas (de manera pareja con las funciones de activación de los modelos de neuronas y RNA clásicos) para ajustar o sesgar las salidas producidas por las capas antecedentes. Funciones típicamente empleadas son tanh o sigmoide (ya comentadas junto con otras anteriormente), pero la que más se suele emplear actualmente es la función ReLU (Unidad Lineal Rectificada), por varios motivos: su definición matemática (tanto la de su función como la de su gradiente) es relativamente simple (y aunque no sea diferenciable esto se puede ignorar en la implementación práctica), no causa problemas de retropropagación debido a su gradiente constante para valores positivos (otras funciones sí debido a que su gradiente es cercano a cero en casi todas partes menos en el origen ["gradiente de fuga"]) y el entrenamiento de la red puede mejorar en cuanto a rapidez en la convergencia a causa de que gradientes nulos (entradas "inhibidoras") equivalgan a respuestas nulas de la función no lineal (restricción de no negatividad) [72].

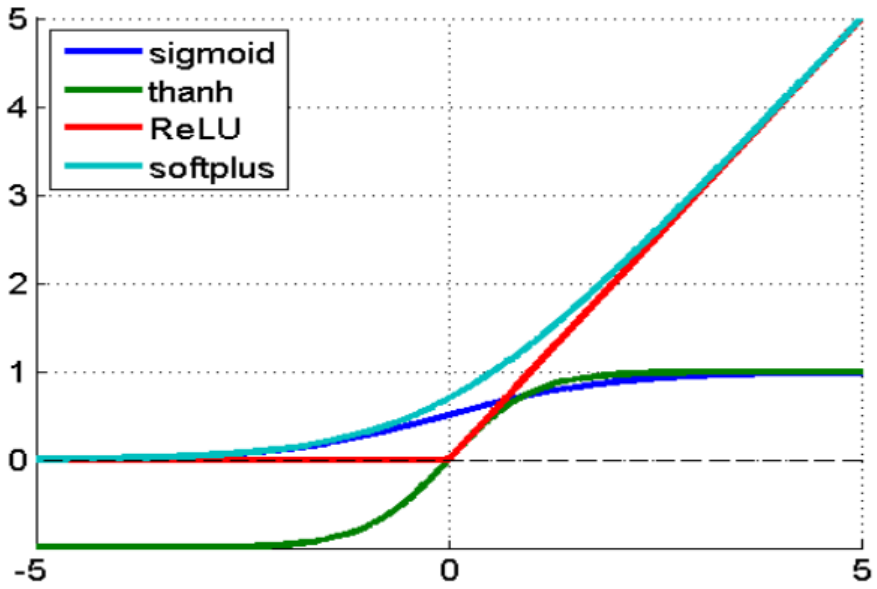

Fig. 15. Tipos más comunes de funciones de activación no lineales. Estas se suelen usar para problemas de clasificación, mientras que las lineales para casos de regresión recogido de [72]).

Finalmente, se puede mencionar la presencia de una última capa principal bastante relevante, la capa "totalmente conectada", que recuerda a la arquitectura que tomaban las RNA tradicionales, pues cada neurona de esta capa está conectada con todas las neuronas de sus capas adyacentes (sean o no capas totalmente conectadas). Aunque es elevada la complejidad de un óptimo diseño de una de estas capas, pues involucra muchos parámetros explícitos e implícitos que precisan ajustarse en el proceso de entrenamiento, con una técnica denominada dropout pueden agregarse capas adicionales que anulen la contribución de ciertas entradas (o salidas de "neuronas ocultas") y dejen intactas el resto con tal de evitar el sobreajuste de los parámetros en el entrenamiento, es decir, que en la primera época de entrenamiento las muestras utilizadas no influyan de una manera desproporcionada, como presumiblemente ocurriría sin la presencia de una capa dropout (sobreentrenamiento; un problema típico que puede aparecer en cualquier solución de IA y ocurre cuando el sistema "memoriza" las respuestas a partir de los datos de entrenamiento en vez de evolucionar realmente mediante aprendizaje).

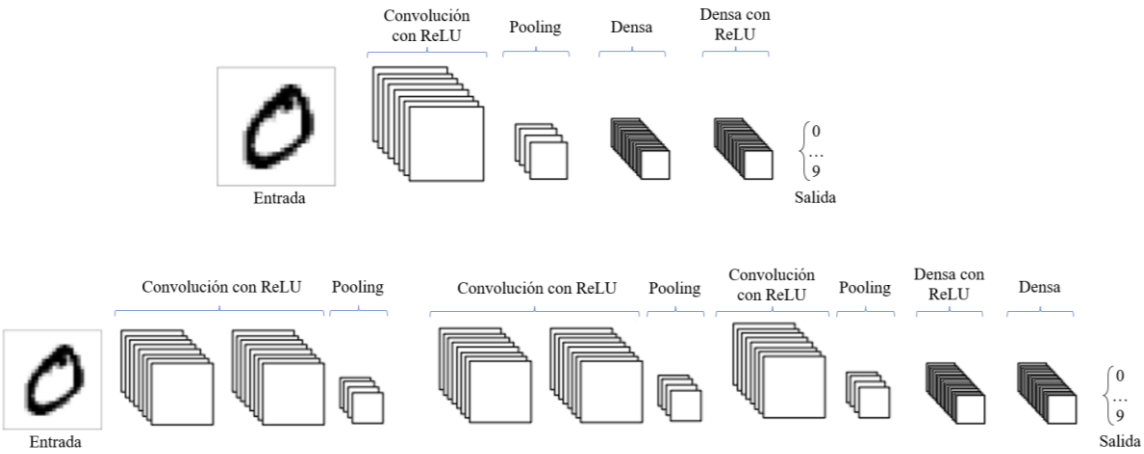

Fig. 16. Dos ejemplos de arquitectura de RNC que se pueden emplear para el reconocimiento de dígitos en imágenes y su clasificación; una red simple con cuatro capas (arriba) y una más completa con unas diez capas colocadas de manera razonable en concordancia con las funcionalidades que brinda cada una (abajo) (adaptado de [71]).

Algunas RNC ampliamente conocidas y que son signo de que las redes de aprendizaje profundo tienen una exitosa cabida en las aplicaciones de visión artificial son: DanNet (la primera RNC pura en ganar competiciones internacionales de visión por ordenador, entrenando mediante GPU y siendo de los primeros éxitos del aprendizaje profundo moderno y potenciadores [desde entonces] del resurgir y progreso tecnológico incansable de la inteligencia artificial [29, pág. 40] [105]) , AlexNet (contiene 5 capas convolucionales con max-pooling después de la primera, segunda y tercera, más 2 totalmente conectadas [650 mil neuronas, 60 millones de parámetros y 630 mil conexiones]) [34, 82], LeNet (7 capas en total [excluyendo la de entrada]: 3 capas convolucionales, 2 de submuestreo y otras 2 totalmente conectadas) [72, 85] y GoogleNet (con la que se introdujo un nuevo concepto llamado *Inception Module*, que redujo en un 93 % el número de parámetros de la red con respecto a AlexNet) [99].

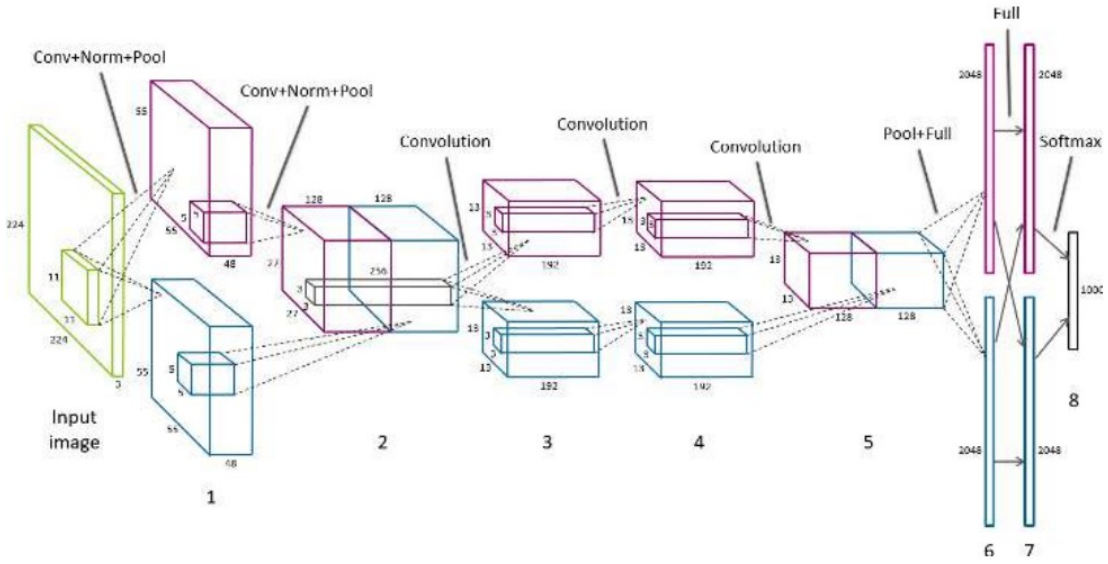

Fig. 17. Diagrama de arquitectura de la red neuronal AlexNet; los distintos tonos de color que se muestran en los elementos de la estructura denotan las capas que se computan en una GPU u otra, pues el artículo original divide el proceso y los datos de cálculo en dos GPU (con excepción en los grupos 2 y 5 de la red, sin contar la entrada y la salida) (recogido de [122, pág. 154]).

Simulación de las RNC con MATLAB

Al inicio de este documento se planteaba hacer un estudio exhaustivo de las redes neuronales (así como de sus tipos) y, de forma práctica, en base a varios datasets de índole industrial aplicados a una red convolucional para detectar objetos con defectos, realizar distintas variaciones de entrenamiento para racionalizar los resultados que proporcionen, extraer ventajas y desventajas de cada variación y relacionar (heurísticamente) todo ello con conceptos típicos de la inteligencia artificial como la imprecisión inherente a los datos de entrada e imperfección de las redes neuronales, las funciones de activación como sigmoide, ReLU, *linear* y tanh, la capa "batch normalization", el efecto de la variación del tamaño de las imágenes de entrada en el resultado de la clasificación, el "sub" y "sobreentrenamiento", la técnica *data augmentation*, el desequilibrio de datos y otros conceptos similares que surgen de interés a lo largo del desarrollo del trabajo y que se pueden ver presentados más explícitamente en diferentes lugares de esta memoria; además, se plantea de forma concluyente y casi anecdótica introducir algunos métodos y algoritmos de visión artificial que puedan ayudar a mejorar, a priori y posteriori, la clasificación y los resultados de la red neuronal.

En el este apartado se plasma el enunciado del problema, las explicaciones y los detalles que completan la solución particular desarrollada para este trabajo con la herramienta MATLAB. Varios métodos son empleados para tal labor que, de manera heurística, sirven auxiliarmente para introducir y tratar numerosos conceptos sobre la inteligencia artificial y, específicamente, las redes neuronales convolucionales, como se va contemplando en cada uno del resto de apartados y los propios subapartados de que consta este, todos contextualizados enfáticamente sobre la comparación entre conceptos teóricos de interés y resultados empíricos (extraíbles y verificables versátil y fácilmente) obtenidos gracias al código proporcionado junto al documento.

Descripción del problema planteado

El caso de uso particular que ejemplifica la aplicación en que serviría un potencial diseño de RNC dice así: "Una empresa fabrica distintos productos en producción seriada. Dicha empresa quiere mejorar sus estándares de calidad y minimizar el número de objetos que llegan al usuario final de forma defectuosa, por ello, necesitan identificar estos casos previamente a la etapa de empacado. Los productos son desplazados con una cinta transportadora, por lo que ha sido posible situar una cámara en la parte cenital y realizar una fotografía de cada uno de ellos. Se busca desarrollar un algoritmo de visión artificial capaz de determinar cuál es defectuoso y cuál no, así una palanca descartará estos productos. Solamente utilizan una referencia de producto simultáneamente, por lo que en las imágenes no se verán alternados distintos productos".

Se decide estudiar las RNC para explorar las soluciones a este caso de uso, como cabe suponer dado el interés de este trabajo; para entrenar "la red" se usan, a lo sumo, los dataset "metal nut", "cable", "transistor" y "bottle", que pertenecen a un conjunto de imágenes realizadas en entorno y ambiente controlado de carácter industrial [87][6]. Se cubren distintas tareas secuencialmente para discutir los "buenos" o "malos" resultados que proporcionen ante distintas imágenes de test las redes con que se experimenta, si bien se discuten los problemas y las bondades que puedan otorgar las características de cada una. El motivo de seleccionar estos dataset es que son similarmente válidos que otros conjuntos de la misma compañía que los proporciona u otros conjuntos de datos independientes que, por el contrario, son notablemente más "pesados" computacionalmente; es remarcable además el interés que tienen estas imágenes en el ámbito de una aplicación industrial.

Se definen con ayuda de MATLAB, razonadamente, varias RNC de distintos números de capas convolucionales y densas, neuronas, estructuras, funciones de activación, etc. (arbitrariamente en base a lo que se "va obteniendo") para discutir los resultados que proporcionan. Cabe mencionar que, en una aplicación real, la manera última en que se

[6] Recogidos del sitio web oficial de la empresa MVTec: https://www.mvtec.com/company/research/datasets/mvtec-ad (bajo licencia Attribution-ShareAlike 4.0 International [CC BY-SA 4.0] de Creative Commons).

evaluaría la validez de los resultados experimentales (en programación) de la red neuronal convolucional desarrollada y entrenada sería a través de la detección de los objetos defectuosos (de las carpetas de test de los dataset específicos o imágenes nuevas tomadas "en línea"), pues idealmente se buscaría alcanzar una precisión del 100 %, ya que a posteriori un falso negativo significaría un producto válido desechado. Dado el alcance de este trabajo se observan diferentes métricas y se analizan más aspectos que la especificación funcional de la aplicación, aunque también se tiene muy en cuenta la persecución de resultados que sean todo lo óptimos posible. Como se comenta en la presentación, junto con este informe se adjunta el fundamental código fuente desarrollado que, velando por una modularidad en la estructuración versátil de información (algo que proporcionan los visuales scripts desarrollados), recoge diversos comentarios ilustrativos, instrucciones y descripciones de lo que se ha implementado en la práctica con el presente trabajo.

Experimentación general y comparación de diferentes arquitecturas

Dentro del campo del aprendizaje profundo, una RNC es una potente técnica de aprendizaje automático, concretamente para ser entrenada utilizando grandes lotes de imágenes de diversa índole y aprender extensas representaciones de características para una amplia gama de imágenes. En este subapartado se procede a diseñar y entrenar arquitecturas de redes neuronales convolucionales (razonablemente) sencillas para la detección de defectos en imágenes, es decir, clasificación de imágenes en "defectuosas" y "no defectuosas" en concordancia con las muestras de partida con que se entrena; como estas imágenes son instancias específicas de clases de objetos pertenecientes al ámbito industrial (cables, roscas de metal, transistores bipolares, botellas de cristal… todo ello en un plano cenital muy específico y controlado) no tendría mayor sentido introducir imágenes de otro tipo (ni estos mismos objetos en perspectivas sensiblemente diferentes de las esperadas), pero podría contemplarse la posibilidad de que esto ocurriese en una implementación "más real".

El primer paso es cargar y explorar los conjuntos de imágenes de que se dispone; para la experimentación se decide que se va a comenzar por entrenar una red compuesta por un solo tipo de objeto, sea el relativo al dataset "metal_nut", así tras experimentar largo rato con él se podría pasar a tratar e integrar otros dataset para ver cómo varía la red antes los distintos tipos de entrada y, más aún, se podrían entrenar diversas arquitecturas de red para computar y clasificar varias imágenes de diferentes categorías al mismo tiempo. La razón por lo que se decide no hacer lo último en primera instancia es que las imágenes en los datasets originales no tienen las mismas dimensiones, por lo que para aprovechar al máximo la fotografía (aunque podrían directamente normalizarse los tamaños con sencillas funciones de manipulación de píxeles) se comienza tratando los lotes independientemente y, si acaso, posteriormente experimentar de la otra manera.

Así pues, se comienza tratando un solo dataset, como se puede comprobar concurrentemente en el código adjunto. La idea en este primer experimento es considerar la red como un filtro o control de calidad enseñado con "lo mínimamente buenos o malos que se deben considerar los objetos para definirlos como defectuosos o no defectuosos", pues clasifica el elemento presente en cada imagen operando de una manera adaptada (y

ciertamente inusual) al problema particular enunciado; algunos pasos tenidos en cuenta en la implementación práctica de la RNC son: la organización jerárquica en el sistema de archivos del sistema para el tratamiento y manejo flexible de las imágenes (definir la[s] carpeta[s] donde se encuentran las imágenes preclasificadas), la carga de la base de datos en el sistema de forma categórica (definiendo un nombre para cada tipo de imagen de entrenamiento, típicamente el nombre que identifica a los objetos y defectos), la especificación del tamaño de las imágenes en la capa de entrada de la red (así como a posteriori se puede especificar el número de clases en la última capa totalmente conectada de su red con el argumento "OutputSize") o la división en conjuntos de datos y de validación (por ejemplo, el 90 % para entrenamiento y el 10 % restante para validación [los datos de test se simplifican a veces semejándolos por estos o se escogen de conjuntos disjuntos independientes —*transfer learning*—]).

Cabe mencionar que son distintos los propósitos de los (sub)conjuntos de datos de entrenamiento, validación y test para una red neuronal artificial (es recomendable que todos procedan de la misma distribución, siendo lo usual realizar una partición de los datos disponibles en tres porciones); a lo sumo:

- Los datos de entrenamiento se usan para computar el gradiente y actualizar los pesos y "bias" de la red. Son los datos principales que se usan en el proceso de entrenamiento y el error obtenido de estos es típicamente una de las métricas de aceptación de los modelos de predicción o clasificación; es decir, estos datos "de ejemplo" se usan para realizar el ajuste del modelo predefinido.

- Los datos de validación tienen que ver con la monitorización del error durante el proceso de entrenamiento a fin de "comparar" diferentes modelos (o variaciones de uno solo durante el entrenamiento) y, aparte, discernir cuál es el "más adecuado". Los datos usados para hacer *early stopping* (evitar el sobreajuste del modelo) están contenidos dentro de los de entrenamiento, los datos de validación disjuntos se usan para comparar modelos. Su error suele decrecer al inicio del entrenamiento, al igual que lo hace el de los de entrenamiento, pero a medida que evoluciona (posible inminente sobreajuste debido a sobreentrenamiento) el error de validación empieza a crecer en algún punto. Los pesos y el bias de la red

neuronal se "guardan" en la iteración correspondiente al mínimo del error de validación. Es relevante mencionar que si el mínimo del error de entrenamiento se alcanza en una iteración significativamente distinta a la relativa al de validación, puede significar una mala elección en la división de los subconjuntos de datos. A lo sumo, puede decirse que estos datos (los contenidos en el entrenamiento) sirven para detener el proceso de entrenamiento y evitar el sobreajuste y además, alternativamente (datos de validación "canónicos" o independientes), evaluar el modelo para buscar el de "mejor" o que "mayor capacidad de generalización", que típicamente es el de menor error de validación.

- Los datos de test se pueden ver como el conjunto de datos que "sobren" habiendo determinado el tamaño de los otros dos, si es que se decide extraer del mismo lote. No se usa durante el entrenamiento (al contrario que los demás) pero sí después ("algo más fiable" que evaluar con los otros) para comparar distintos modelos y obtener una referencia cuantitativa global del comportamiento y rendimiento de diferentes modelos o redes. De manera general, si la precisión del test es cercana a la del entrenamiento significa que el modelo generaliza "bien" la predicción de datos, en caso contrario (si son demasiado dispares) puede ser que el modelo no haya entrenado bien y no sea realmente útil, lo que es indicio de una mala selección de muestras, partición de conjuntos u otras consideraciones.

```
layers = [
    imageInputLayer(size_img)

    convolution2dLayer(3, 8, 'Padding', 'same')
    reluLayer

    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 16, 'Padding', 'same')
    reluLayer

    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 32, 'Padding','same')
    reluLayer

    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer];
```

Fig. 18. Definición de arquitectura RNC I.

Se puede seguir definiendo una arquitectura para la RNC, tal como muestra la anterior figura; se decide implementar una arquitectura típica y sencilla para poder experimentar en base a ella en posteriores pasos (de nuevo, se recomienda atender al código adjunto para una visión más específica); varios comentarios se pueden hacer sobre las capas implementadas:

- En la capa de entrada *imageInputLayer* se especifica el tamaño de la imagen de entrada (en este caso 700 píxeles de alto y ancho por 3 canales de color [RGB]); si la red se quisiera entrenar con imágenes de distintos tamaños (distintos lotes) habría que escalarlas, realizar el procesamiento de colores adecuado y organizar correctamente la información de cada una para que cumplan con el "prototipo" definido. Esto es una RNC; no hay 700x700x3 neuronas de entrada que analizan cada píxel y aprenden de cada uno individualmente, sino que hay las "suficientes" para recibir la información entrante, definidas mediante filtros (que se convolucionan a lo ancho y alto del volumen de entrada, produciendo un mapa de activación bidimensional de cada uno) que tienen un campo receptivo pequeño pero se extienden por toda la profundidad del volumen de entrada, resultando en que la red aprende a través de filtros que se activan cuando detecta alguna característica específica en alguna posición espacial en la entrada [125, pág. 448].

- *convolution2dLayer* se refiere a la capa convolucional, en la que se especifica el tamaño del filtro utilizado (ancho y alto del kernel o matriz de convolución [3x3, indicado mediante el primer argumento]) cuando se "escanea" o "barre" cada imagen en el proceso de entrenamiento, por ejemplo. El segundo argumento indica el número de filtros, es decir, el total de neuronas que se conectan a cada región de entrada (8, en este caso). Con el tercer argumento se indica que se implemente la técnica homónima, garantizando que el tamaño de entrada sea el mismo que el de salida gracias al cuarto argumento; se podrían personalizar más parámetros como la tasa de aprendizaje para los pesos y el bias de la red, pero por ahora se decide mantener cómodamente sin ninguna especificación más.

- La capa *reluLayer* implementa la función de activación no lineal ReLU que, como ya se ha comentado múltiples veces a lo largo de este documento, suele ser

la función de activación más recurrida en este tipo de modelos de aprendizaje automático. Otras funciones de activación que se pueden probar son la sigmoide, tangente hiperbólica e incluso una lineal; se comprobaría empíricamente que la función ReLU es la que mejor hacer converger el aprendizaje de la red neuronal, aunque las otras puedan adaptarse correctamente a problemas simples y su definición matemática también sea relativamente sencilla.

- *maxPooling2dLayer* se refiere a la capa de pooling o submuestreo que reduce el tamaño de su entrada (espacialmente [en píxeles, no en bits de intensidad]) para simplificar el mapa de características que se "esté analizando" en particular, pudiendo así ser útil para eliminar información redundante en la imagen. Más aún, los efectos del submuestreo permiten que se pueda incrementar el número de filtros en capas convolucionales más profundas sin aumentar la cantidad necesaria de cálculos por capa (agregando más capas de pooling cuantas más capas densas se quieran utilizar). Se usa la técnica más común de pooling, el max-pooling, que devuelve los valores máximos de las ventanas rectangulares que se escanean (primer argumento de la función, tamaño de ventana 2x2). El segundo y tercer argumento indica el salto o stride que da la ventana en cada "escaneo" consecutivo (stride de 2); cabe mencionar que se yuxtaponen varias etapas similares para conseguir resultados más meticulosamente y "poco a poco".

- *fullyConnectedLayer* se refiere a la capa totalmente conectada, en la que todas sus neuronas se conectan con todas las neuronas de las capas adyacentes. Se podría decir que esta capa combina todas las características (patrones de mayor interés) aprendidas por las capas previas de la red; así, la última capa totalmente conectada (podrían agregarse algunas más previas para "recoger minuciosamente más características aprehensibles o aprendidas") sería la encargada de poner en conjunto la totalidad de las características de la imagen consideradas para clasificarla. Así, el parámetro OutputSize en la última capa totalmente conectada es igual al número de clases en los datos objetivo; como las imágenes que entren a la RNC pueden tener o no defectos y la red detecta esto mismo (en esta configuración particular), se elige 2 como el número de salidas de esta última capa.

- softmax se corresponde con una función de activación que normaliza la salida de la última capa totalmente conectada; su salida consiste en números positivos que suman hasta 1, los cuales se pueden utilizar posteriormente, en la capa de clasificación, como valores relativos a probabilidades de clasificación, es decir, de pertenencia a ciertas categorías o clases u otras.

- La última capa, *classificationLayer* (capa de clasificación) utiliza las probabilidades proporcionadas por las salidas de la función softmax (dirigidas a cada entrada de esta) a fin de asignar la imagen de entrada a una de las clases mutuamente excluyentes y calcular la pérdida, es decir, determinar "cuánto de segura está la red de que la entrada pertenezca a cada clase" y computar y devolver una decisión concluyente.

Antes de entrenar la red, se debe realizar un análisis y una decisión de las opciones de entrenamiento, valiéndose de las útiles facilidades que proporciona MATLAB como herramienta que facilita una capa de abstracción notablemente superior de cara a la introducción versátil y directa de múltiples campos de estudio e investigación. De entre las opciones de entrenamiento tenidas en cuenta en la especificación mediante la función *trainingOptions*, las más relevantes son: entrenar la red mediante un método de descenso del gradiente estocástico con momento (SGDM) (optimización o minimización de la función de coste), tasa de aprendizaje inicial (una diezmilésima, en este caso), número máximo de épocas (igual a 4) (ciclos de entrenamiento completos [en los que se completa el procesamiento de todo el conjunto de datos de entrenamiento] al final de cada cual se actualizan los parámetros de la red) y monitorizar la validación y el entrenamiento durante el proceso a través de la definición de una frecuencia para su control y muestra por pantalla (valores empíricos; parece que entre una quinceava y veinteava parte [o alrededor de una quinta parte si el dataset es relativamente complejo] del ratio entre el número de iteraciones y las épocas es razonable [o se puede calcular como el número de muestras en total dividido por el tamaño de los lotes]), entre otros aspectos más detallados en el código y la documentación de las funciones y la *toolbox*.

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate', 1e-4, ...
    'MaxEpochs', 20, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', imdsValidation, ...
    'Verbose', false, ...
    'Plots', 'training-progress', ...
    'ExecutionEnvironment','gpu', ...
    'MiniBatchSize', 8);
```

Fig. 19. Opciones de entrenamiento (típicas) en que se basa el primer entrenamiento de prueba. Se ve que se escoge un tamaño de lote de 8; es decir, en vez de utilizar todas las imágenes de entrenamiento "de golpe", se hace más lentamente en subconjuntos de 8 imágenes. Esto aligera la computación en lo que se refiere a memoria RAM.

Considerando que las condiciones para entrenar la red con MATLAB ya están cumplimentadas, puede antojarse recomendable hacer uso de la unidad GPU (lo que requiere tener instalado *Parallel Computing Toolbox*) para agilizar enormemente el proceso, pues de lo contrario MATLAB recurre a la directamente disponible en el equipo y puede provocar severos retrasos en otros procesos activos remanentes (aunque si hay GPU disponible y se cumplen los requisitos la función la utiliza por defecto, pero [entre otras opciones] se puede indicar explícitamente el entorno de ejecución a través del argumento "ExecutionEnvironment" [introducido desde la versión R2016b de MATLAB]). La GPU de que se dispone, en un portátil ASUS TUF Dash F15 con procesador Intel(R) Core(TM) i7-12650H (duodécima generación) a 2,30 GHz y 16 GB de RAM, es la NVIDIA GeForce RTX 3050 Ti (7,8 GB de memoria RAM en total), que en concordancia con la documentación específica es compatible con la computación "en paralelo" de MATLAB con un valor de "ComputeCapability"[7] (indica de cierta forma la capacidad de computación de la GPU con respecto a la posibilidad de ser utilizada de este modo en MATLAB) de 8,6 y, por tanto, se puede utilizar sin problema para entrenar la red (extraído gracias a los comandos *gpuDeviceTable* más *gpuDevice*).

Parece lógico suponer que debería entrenarse la red, en lo relativo a las imágenes que se deben considerar como defectuosas, con unos datos distintos a los que se utilizarían para hacer el test, aunque no sería tampoco completamente necesario. Como se decide mantener esta suposición, se entrena la red con la "mitad inferior" de las imágenes

[7] Consultar la web oficial de MATLAB: https://es.mathworks.com/help/rtw/ref/compute-capability.html.

defectuosas que proporcionan los datasets en las carpetas particulares de test (se ha personalizado la lógica y organización de las carpetas adjuntas del trabajo con este documento por una mayor comodidad) y utilizar la "mitad superior" para establecer los lotes que se usan en las pruebas de precisión de la red.

Antes de nada, cabe remarcar un aspecto de cierto interés… puede deducirse de manera intuitiva que si se elige un tamaño del kernel (filtro de convolución) pequeño, se extraerán más detalles que con uno mayor, lo que puede conducir al sobreajuste y también a aumentar la potencia de cómputo. Asimismo, un kernel demasiado grande equivale a un número proporcionalmente reducido de neuronas en las capas de entrada (por lo menos), lo que puede conducir, por otra parte, a un ajuste o una calidad de entrenamiento insuficiente. Dada una entrada concreta a la red neuronal, todos los elementos de las capas de esa red constituyen una "transformación" de esta entrada en una salida prevista; así, la medida de la variación entre esta salida prevista y la salida real (computada por la red en cada época) se define como "pérdida". El valor de esta pérdida se "retropropaga" hacia cada filtro (o matriz de convolución) (algoritmo backpropagation) y se utiliza para ajustar los parámetros a fin de minimizar la diferencia entre el resultado previsto y el real (optimizar el entrenamiento); de este modo, el valor de los filtros se ajusta durante el entrenamiento y, por tanto, se puede decir que el sistema neuronal converge cuando la "función de pérdida" queda minimizada (los filtros de red también pueden inicializarse a partir de los pesos de otra red para llegar a acelerar la convergencia óptima del entrenamiento [esto se conoce como "aprendizaje por transferencia"]).

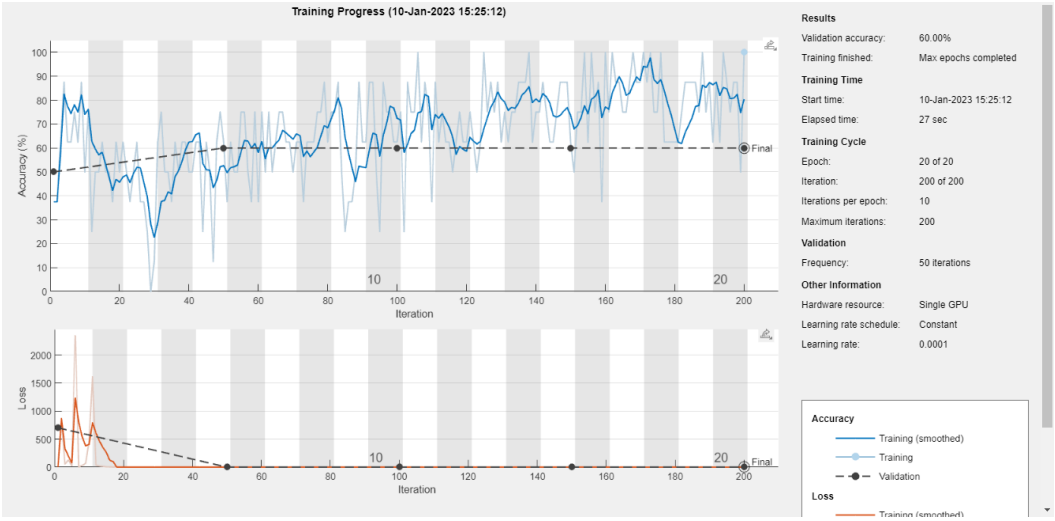

Fig. 20. Gráfica de progreso del entrenamiento I. Muestra la pérdida (de entropía cruzada [medida usualmente utilizada como función de perdida en múltiples arquitecturas de RNA, relacionada con la "entropía de Shannon" [112]]) y la precisión (porcentaje de imágenes clasificadas correctamente) de los datos de validación y los lotes (varios lotes si se usa normalización por lotes) del entrenamiento[8]. Se ve que el resultado, por ahora, no es muy prometedor (60 % de precisión de validación) pero, por lo menos, el proceso de entrenamiento ha sido sumamente veloz.

Habiendo entrenado la red, esta ya debería de ser capaz de etiquetar (clasificar como defectuosas o no defectuosas) las nuevas imágenes (o modificaciones directas de las de entrenamiento) que se le proporcionen, pudiéndose calcular también la precisión de cada clasificación. Se ve que aunque se usa una arquitectura relativamente simple y el conjunto de imágenes abarca un solo tipo de objeto se consigue una precisión "no muy buena", aun habiendo especificado 20 épocas de entrenamiento; esto se puede deber a que las imágenes de entrenamiento no defectuosas son "demasiado" similares entre sí, hay un número demasiado escaso dada la complejidad del problema y la variabilidad de los datos de entrada al entrenamiento vienen definidos, en la inmensa mayoría, por las imágenes defectuosas (que priman en diversidad), lo que hace la red "no se entere muy bien" de las características esenciales que hacen que cada imagen sea considerada como realmente corresponda (sobre todo en lo que respecta a las no defectuosas). Es lógico suponer además que otro motivo de la pobre precisión de clasificación se debe a que la arquitectura

[8] Consultar el sitio web de la documentación oficial de MATLAB para más información técnica sobre este tipo de gráficas y otros aspectos relacionados con el tema: https://es.mathworks.com/help/deeplearning/ug/monitor-deep-learning-training-progress.html.

es bastante sencilla y, deliberadamente, se le ha privado de ciertas capas esenciales que mejorarían notablemente el resultado. Se puede ver en el código lo que se programó para validar esta primera red: se crearon distintos lotes de imágenes de test, estando algunos compuestos íntegramente por imágenes de roscas sin defectos, otros con defectos y otros mezclando objetos con y sin defectos, a fin de que la red clasifique (valide) lo que corresponda, proporcionando resultados de los que es fácilmente extraíble el porcentaje de acierto final (mediante una función propia). Cabe mencionar que la medida de "precisión" que se ha implementado es especialmente fidedigna, pues no se cuenta simplemente el número de imágenes que se han clasificado como defectuosas y no defectuosas para realizar las proporciones, sino que se tiene en cuenta que verdaderamente la imagen que se "había supervisado" como defectuosa (p. ej.) coincida correspondientemente a la que hubiese clasificado la red (clasificada necesariamente también como defectuosa para que "aumente" el grado de precisión de la métrica con que se valora la red).

Tabla 1. Resultados de ciertas pruebas para la RNC I.

Lote del test / Resultado	Normales	Defectuosas	Precisión (%)		
			prueba	train	valid.
20 normales	16	4	80	83,72	60
45 defectuosas	27	18	40		
20 normales y 45 defectuosas	43	22	52,31		

La tabla anterior muestra los resultados de cierta prueba a la que se sometió la red comentada; en azul se ve el contenido de los tres tipos de lotes creados ("normal" se refiere a objetos sin defectos) y en verde distintas métricas (se decidió utilizar las más directas conceptualmente y a nivel cualitativo) para validar los resultados. La precisión "prueba" se refiere al número de datos clasificados que realmente están bien clasificados en concordancia con el relativo de cada lote, "train" se refiere a la precisión obtenida con los datos de entrenamiento y "valid.", similarmente, con los datos de validación (o "test", como se denomina en el código [se decide utilizar un lote independiente para las pruebas, por seguridad]). Aunque la tendencia mayoritaria es acercarse a clasificar los datos

correctamente, los resultados no parecen ser muy determinantes con esta RNC (parece que es "demasiado permisiva" con los defectos que se les permite a los objetos; la red tiene que aprender a ser más estricta en cuanto a la determinación de objetos defectuosos [*es hora de presentar un nuevo concepto*]).

Existe una capa ciertamente interesante llamada batch normalization (normalización por lotes) que se agrega a la arquitectura de una RNC cuando se quieren normalizar de alguna manera los datos de entrada, es decir, aplicar una transformación que mantiene la media de los valores de salida próxima a 0 y su desviación estándar cercana a 1 (ajustar los valores medidos referidos en diferentes escalas con respecto a una escala común). Estas capas normalizan un subconjunto de entrada a razón de todas las observaciones para el resto de entradas de manera independiente; para acelerar y optimizar el entrenamiento al integrarlas en la red se aconseja situar estas capas entre las convolucionales y las no lineales (sean las que ejecuten la función ReLU). Tras la normalización, los datos de salida se ven sometidos con respecto de la entrada, tanto en magnitud como en desplazamiento, a escalas en ciertos factores (autoajustables mediante el aprendizaje).

Cabe mencionar que este tipo de capas favorecen y optimizan el entrenamiento de las redes neuronales profundas debido a que ayuda a lidiar con el problema de que la distribución de las entradas de cada capa cambie durante el entrenamiento, a medida que cambian los parámetros de las capas anteriores, lo que usualmente ralentiza el entrenamiento al requerir tasas de aprendizaje más bajas y una cuidadosa inicialización de los parámetros y hace que sea notoriamente difícil entrenar modelos con no linealidades saturantes. De acuerdo con [88], esta capa actúa como regularizador y puede eliminar en algunos casos la necesidad de dropout, además de que con ella se puede conseguir la misma precisión que algunos métodos avanzados de clasificación (entre los que se comenta "aquel" que logró adelantar a AlexNet en la validación "top-5" con el conjunto de datos ImageNet [ILSVRC]) con 14 veces menos pasos de entrenamiento y superando al modelo original por un margen significativo.

```
layers = [
    imageInputLayer(size_img)

    convolution2dLayer(3, 8, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 16, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 32, 'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer];
```

Fig. 21. Definición de arquitectura RNC II.

Habiendo extendido someramente la arquitectura de la red neuronal previa, se puede agregar algún comentario al respecto para ver si esta nueva red proporciona mejores resultados de cara a las diversas pruebas programadas:

- *batchNormalizationLayer* se refiere a la capa de normalización por lotes, que de cierto modo normaliza los gradientes y las activaciones que se propagan a través de la red, haciendo del entrenamiento una tarea de optimización más soportable. Colocando este tipo de capas entre la convolucional y la no lineal (ReLU, tanh…) se puede acelerar el entrenamiento de la red y redecir la sensibilidad a inicializaciones poco apropiadas o muy acusados en sus parámetros, como puede deducirse fácilmente con todo lo comentado.

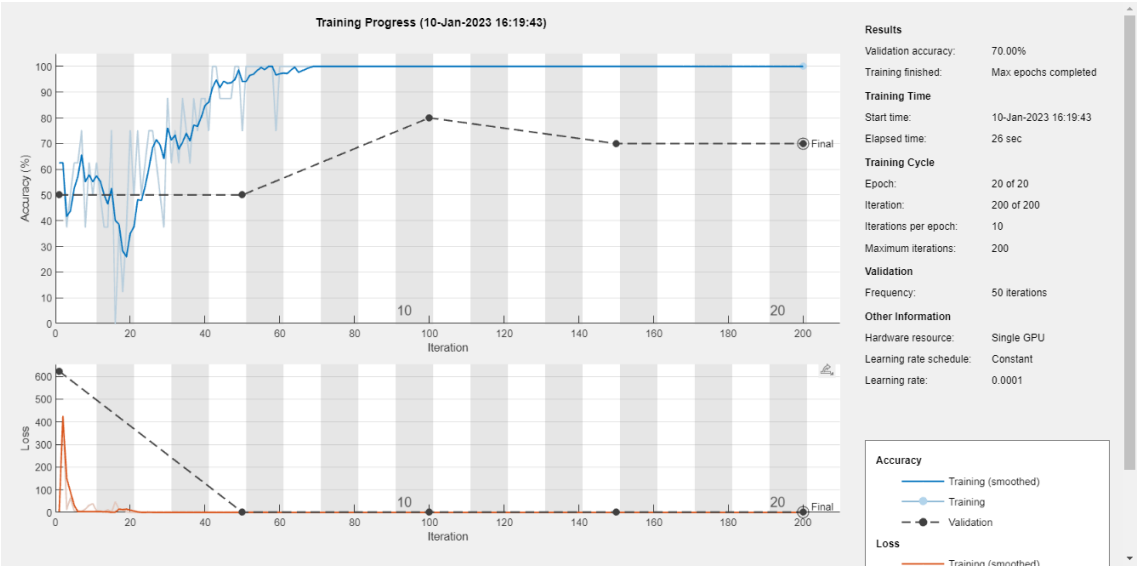

Fig. 22. Gráfica de progreso del entrenamiento II. Aunque la precisión no "despega" todo lo que se quisiera, ciertamente tiende a hacerlo y a una velocidad mucho mayor que antes (70 % de precisión de validación). Lo mismo se puede decir para la "pérdida" del entrenamiento, pues ya no es tan remanente ni extendida como antes; su decrecimiento se ha visto potenciado.

Ahora el resultado es significativamente más prometedor, pues "sobraron" muchas épocas en comparación con el anterior entrenamiento y, aunque la precisión de validación mejora solamente en un 10 %, la red parece que "está mejorando".

Tabla 2. Resultados de ciertas pruebas para la RNC II.

Lote del test / Resultado	Normales	Defectuosas	Precisión (%)		
			prueba	train	valid.
20 normales	11	9	55	100	70
45 defectuosas	19	26	57,78		
20 normales y 45 defectuosas	30	35	56,92		

Aunque los resultados para el lote en que solamente había un tipo de objetos (poca variabilidad) resulta ser algo peor (la precisión baja un 25 %), mejora sensiblemente en caso de que los lotes tengan, además de defectuosos, mayor proporción en estos (se refleja lo comentado en los anteriores párrafos; unas imágenes de entrenamiento diseñadas más minuciosa y atentamente [y sobre todo una mayor cantidad] podrían haber sido de mayor

utilidad, pero está bien mantener los experimentos así para extraer un razonamiento práctico de los resultados obtenidos).

Se pueden añadir algunas modificaciones a la estructura de la red (capas densas, ocultas y su profundidad, número total de neuronas, tamaños de los filtros de convolución, etc.) para ver si es capaz de comportarse aún mejor con modificaciones sencillas y directas.

```
layers = [
    imageInputLayer(size_img)

    convolution2dLayer(8, 8, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    crossChannelNormalizationLayer(1)
    maxPooling2dLayer(4, 'Stride', 4)

    convolution2dLayer(8, 16, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(8, 32, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(8, 64, 'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(2)
    reluLayer
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer];
```

Fig. 23. Definición de arquitectura RNC III.

Habiendo extendido de nuevo y modificado la arquitectura de la red neuronal previa, se puede agregar algún comentario al respecto para ver si esta nueva red proporciona mejores resultados de cara a las diversas pruebas programadas:

- Se ha aumentado el tamaño de los filtros (poniendo “énfasis” particular en los de las primeras capas) y también su número, dando lugar a “extracciones de características” más generales dado el relativo gran tamaño de las imágenes de los datasets.

- *crossChannelNormalizationLayer* se refiere a un tipo de capa que lleva a cabo una normalización de respuesta local para el tamaño de ventana (que controla el número de canales utilizados para la normalización) especificado como argumento (1, en este caso). Según [89], este tipo de capas ayuda a evitar el fenómeno de "gradiente de fuga" o "explosión de gradientes", pues se consigue que estos permanezcan acotados (lo que se traduce en una optimización más eficiente).

- Aunque no se implementa en este sencillo ejemplo, pudo haberse agregado también una capa dropout, que pondría a cero aleatoriamente los elementos de entrada con una probabilidad determinada (*dropoutLayer*, con una probabilidad de 0,5 por efecto) (típicamente en las capas densas finales); el número de conexiones en las capas finales no se considera tan elevado como para que esta capa proporciones mejoras considerables.

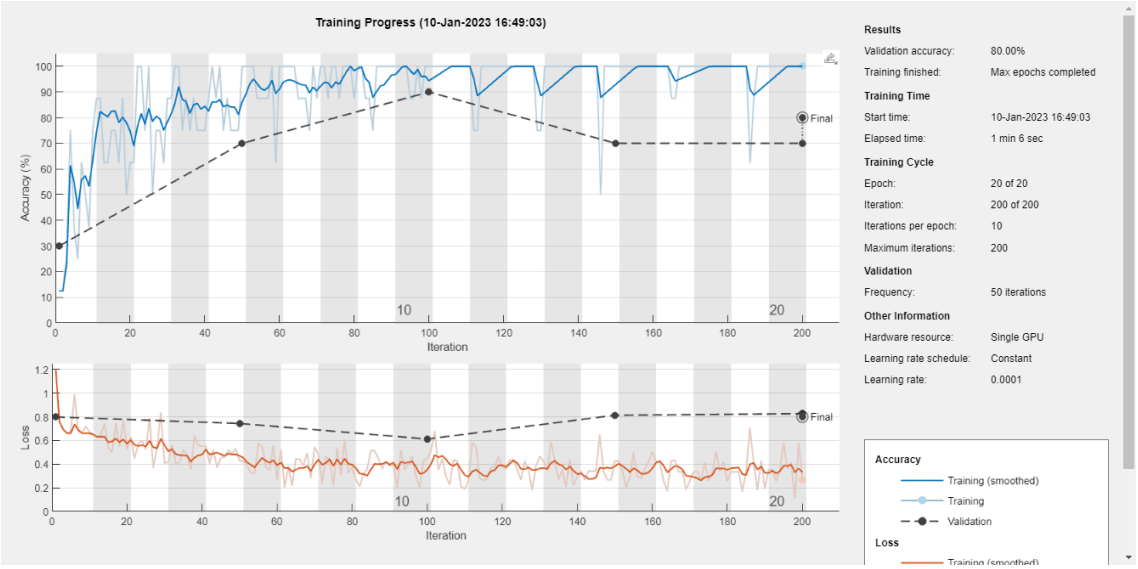

Fig. 24. Gráfica de progreso del entrenamiento III (80 % de precisión de validación). La precisión de validación crece notablemente, acercándose (cada vez más) a lo que parece ser un resultado razonable y aceptable. Aunque ahora la "pérdida" del entrenamiento no cae a cero inmediatamente (al menos su gradiente no es tan pronunciado como el resto de gráficas), la magnitud es inmensamente menor, lo que corrobora el hecho de que esta última reestructuración de la red es de provecho. Parece que se vislumbra cierto efecto de sobreentrenamiento a partir de la época 10, por eso mismo se decide mantener el proceso tal y como se muestra aunque se podría haber "cortado" llegado ese punto.

Tabla 3. Resultados de ciertas pruebas para la RNC III.

Lote del test / Resultado	Normales	Defectuosas	Precisión (%)		
			prueba	train	valid.
20 normales	2	18	90	100	80
45 defectuosas	23	22	48,89		
20 normales y 45 defectuosas	41	24	61,54		

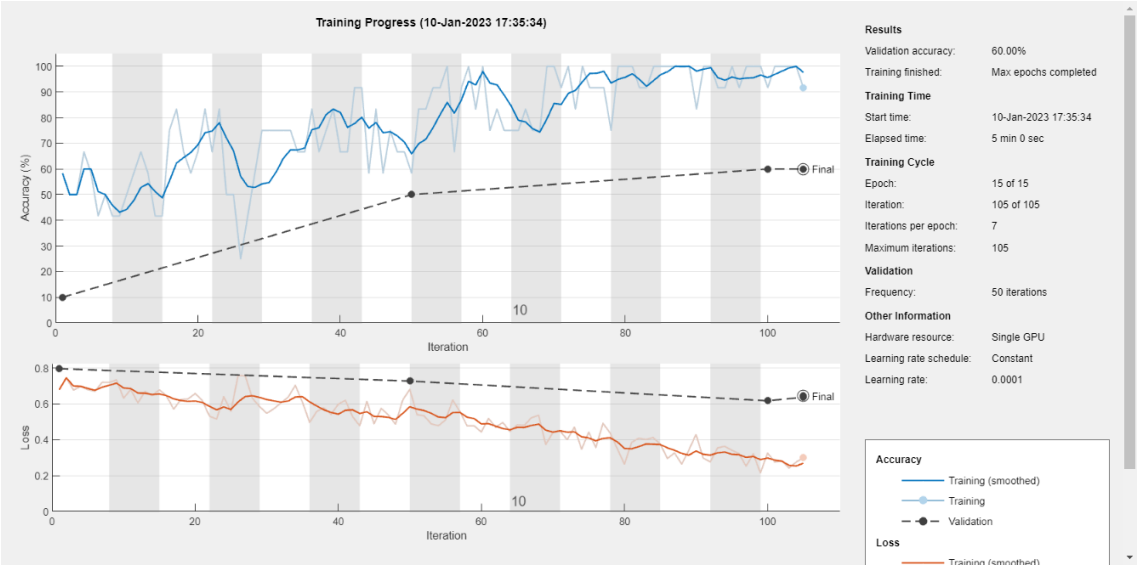

Fig. 25. Gráfica de progreso del entrenamiento IV. Se puede ver y vaticinar que si se le hubiese permitido a la red aprender durante más épocas, con un mayor número (óptimo) de datos y, presumiblemente, a una tasa de aprendizaje lo suficientemente pequeña y un tiempo "ilimitado", el resultado se podría haber ido aproximando (quizás con estructuras de neuronas más "perfectas") asintóticamente a un 100 % de precisión y el error de validación habría alcanzado un mínimo de la mejor manera (aunque el número de capas ocultas y neuronas interconectadas tendrían que escalarse ilimitadamente, así como habría que establecer y agregar mejores capas en la estructura de la red).

Tabla 4. Resultados de ciertas pruebas para la RNC IV.

Lote del test / Resultado	Normales	Defectuosas	Precisión (%)		
			prueba	train	valid.
20 normales	4	16	80	100	70
45 defectuosas	13	32	71,11		
20 normales y 45 defectuosas	29	36	73,85		

Es claro que el tamaño de los filtros con que se caracterizaron las primeras redes no era óptimo; al tratarse de imágenes de alta resolución (dentro de lo que cabe y considerando el tamaño de los defectos de los objetos particulares) el tamaño de las matrices de convolución debería ser (por lo menos en las primeras capas) algo más alto, lo que se verifica comprobando la mejoría que presenta el entrenamiento de la red tras incorporar lo comentado. Aunque la tercera RNC proporcionaba un resultado llamativamente bueno de 90 % de precisión para el primer test, es la siguiente modificación (ver tabla anterior) la que consigue otorgar los resultados más favorables para todas las pruebas (en los que hay imágenes defectuosas) en comparación con el resto; lejos de ser decisorios estos resultados si se tuviera en mente realizar una potencial implementación real, sí indican un comportamiento del que se van induciendo direcciones de mejora en la arquitectura de la red y su entrenamiento o múltiples conceptos inherentes de valiosa utilidad.

Podría hacerse un estudio más profundo y detallista sobre diversas mejoras y reestructuraciones para desarrollar RNC más completas y precisas (incluso variar el porcentaje de datos de entrenamiento y validación, incluir más imágenes en el dataset…), pero el tratamiento aproximado que se hace hasta este punto es suma y suficientemente informativo de acuerdo con el carácter introductorio que se le quiere dar al trabajo. Es interesante probar, a modo de curiosidad, a cambiar ligeramente las opciones de entrenamiento para ver cómo varía la precisión del modelo con la alteración de los parámetros (por ejemplo, aumentando la tasa de aprendizaje o disminuyendo el tamaño de los lotes de entrenamiento parece que el gradiente es más “disperso” o agudo [la red no toma un rumbo “suficientemente fijo”]).

Habiendo experimentado con un lote de imágenes con diversas configuraciones de redes neuronales convolucionales y ensayado con cada una a través de sencillas pruebas de validación, ahora se puede enfocar la práctica en modificar el tamaño de la imagen de entrada bajo distintos supuestos; para ello, en el siguiente subapartado se propone agregar tres lotes de imágenes más a los que se les reduce el tamaño (la escala) para compensar en cierto modo el entrenamiento y sonsacar conclusiones en base a ello, tanto con respecto a la rapidez o velocidad de entrenamiento como a la precisión final de la pruebas.

Experimentación particular con las RNC y validación descriptiva

Como se ha comentado, ahora se prueba a agregar otros lotes de imágenes, correspondientes a objetos de distinto tipo, para ver si el entrenamiento se vuelve más complicado y se pueden sacar resultados más provechosos aunque se puedan volcar (como parece lógico a priori) con menos precisión. Se integran las imágenes correspondientes a los datasets de transistores bipolares, botellas de cristal en plano cenital y cables eléctricos (todo ello con ejemplos de muestras con defectos y sin ellos), recordando que hay que normalizar (igualar) la escala de las imágenes antes de proporcionarlas a la red; como las imágenes de los nuevos lotes son mayores (1024 y 900 frente a 700 "píxeles cuadrados") que las del lote anterior se decidiría reducir las nuevas, pues puede aligerar la computación y se podría no considerar necesaria una resolución tan grande, en principio (otra opción sería hacer padding a la entrada en las imágenes de menor tamaño), pero de acuerdo con lo comentado en el anterior subapartado, se decide reducir el tamaño de todas las imágenes a una misma escala, a decir, 256 píxeles cuadrados por 3 canales de color. Cabe mencionar que se decide establecer un tamaño de lote típico ("MiniBatchSize" [batch size; ver en siguientes párrafos]) de 12 imágenes para cada entrenamiento, a fin de quitar peso a la computación.

En este subapartado se opta a implementar una estructura AlexNet sin entrenar, por practicidad (se requiere instalar "Deep Learning Toolbox Model for AlexNet Network" en MATLAB); en la siguiente figura puede verse una representación esquemática y visual de esta red. Aun así, hay que sustituir las tres últimas capas y la de entrada (como mínimo) para establecer una concordancia con los datos de imágenes de entrada de que se dispone; es decir, hay que extraer todas las capas, excepto la primera y las tres últimas, de la red preentrenada, sustituyendo las últimas tres por una capa totalmente conectada, una capa softmax y una capa de salida de clasificación y además ajustando el tamaño de entrada de la primera capa al que tengan las imágenes que se proporcionen para el entrenamiento. Como recomendación de la documentación de MATLAB, se inicializan los factores de aprendizaje del bias y de los pesos a un valor de 20 (típico); esto hace que el aprendizaje sea más rápido en las capas nuevas que en las capas transferidas.

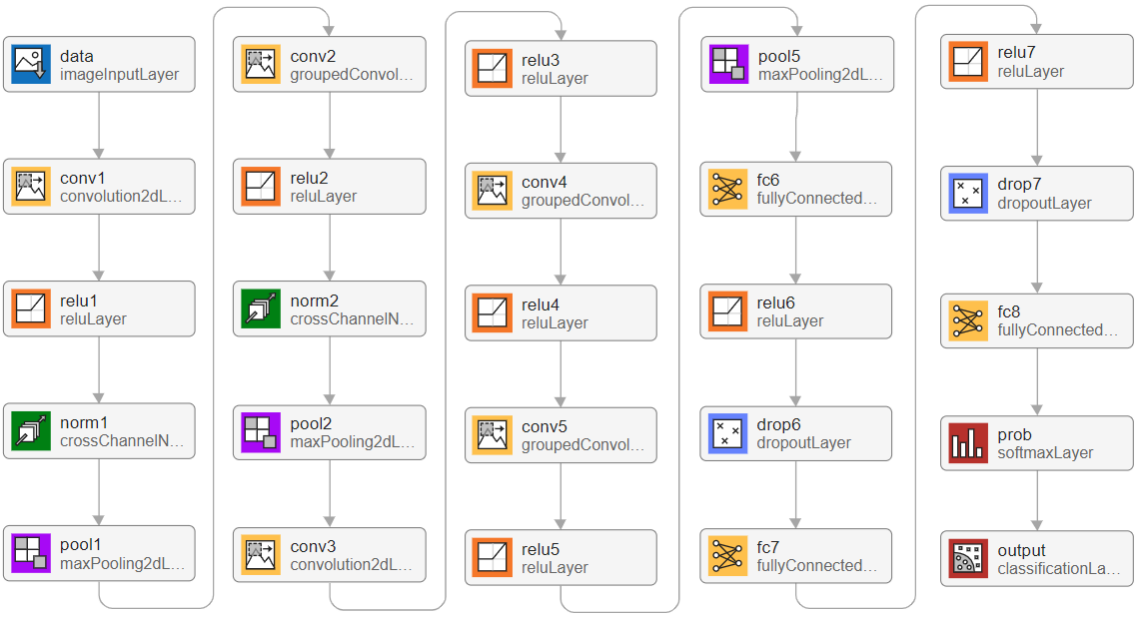

Fig. 26. Arquitectura fundamental esquematizada de la popular red neuronal artificial convolucional AlexNet, la cual se ha decidido utilizar para los experimentos realizados en el presente subapartado (diagrama extraído mediante el comando de MATLAB deepNetworkDesigner(alexnet)).

Para dividir en conjuntos de datos y de validación, por ejemplo, el 90 % para entrenamiento y el 10 % restante para validación, se calcula el máximo conjunto de datos de entrenamiento en base al tamaño mínimo de los lotes disponibles (hay considerablemente menos muestras de imágenes de la categoría defectuosa), dado que se decide no entrenar la red con una mayor cantidad de imágenes de entrenamiento que de muestras existentes de alguna de las categorías. Esto es una simplificación o suposición, hecha simplemente por motivos de flexibilidad en el código y la comprensión de aspectos relevantes de las redes neuronales más que para la búsqueda de una solución máximamente óptima (se podría contemplar de otra manera en una implementación real). Dicho de otra forma, se implementa un sesgo del número de imágenes de entrenamiento por categoría al mínimo número de muestras de la categoría mínima correspondiente.

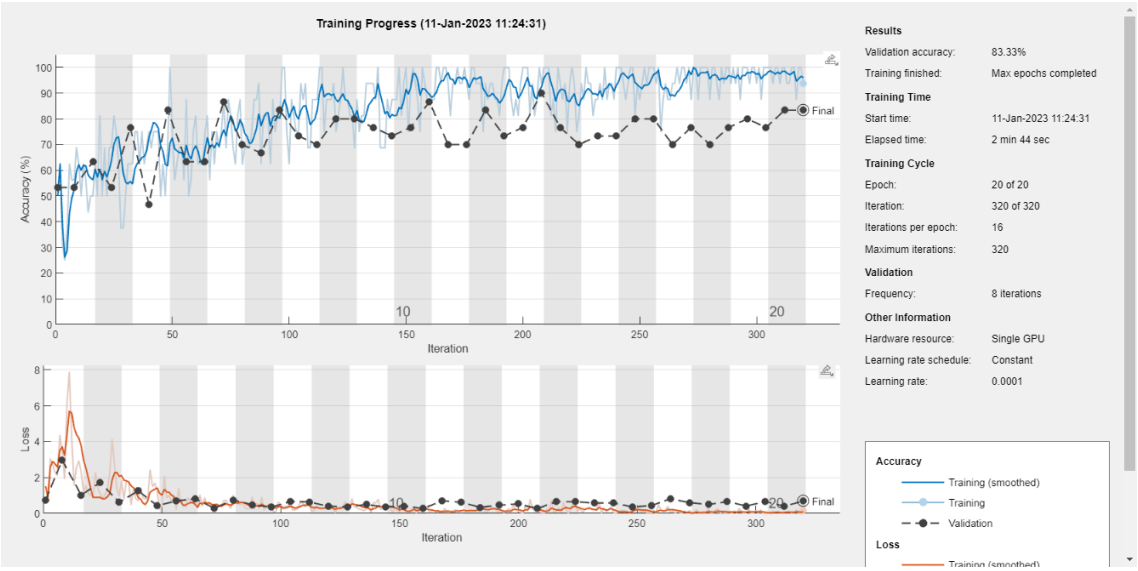

Fig. 27. Gráfica de progreso del entrenamiento con la red AlexNet modificada I, ante imágenes de 256x256 píxeles (83,33 % de precisión de validación).

En comparación con las gráficas vistas en el anterior subapartado, esta vez la red parece empezar a entrenar provechosamente "de verdad", de suerte que la estructura de AlexNet (aunque modificada) es notablemente mejor que la de los sencillos ejemplos previos.

Tabla 5. Resultados de ciertas pruebas para red AlexNet modificada I (clasificación múltiple de objetos pertenecientes a lotes o categorías diferentes).

Lote del test / Resultado	Normales	Defectuosas	Precisión (%)		
			prueba	train	valid.
80 normales	75	5	93,75	100	83,33
141 defectuosas	46	95	67,38		
80 normales y 141 defectuosas	121	100	76,92		

Los resultados se muestran altamente mejorados con respecto a las redes anteriores, aun habiendo disminuido la resolución de las imágenes y cuadruplicado alternativamente los tipos de objeto con que se entrena la red de clasificación.

Ahora se va a probar a disminuir sensiblemente el tamaño de las imágenes de entrada (creando otras estructura de red) para ver si, bajo los mismos parámetros, tiene correlación la variación de la escala de las muestras de entrenamiento con la precisión de la

clasificación. Se tiene cuidado de no disminuir "demasiado" el tamaño de las imágenes debido a que se decide no modificar ciertas capas de la arquitectura base de AlexNet, la cual "espera" recibir, aproximadamente, cierto número de entradas y activaciones propagándose a lo largo de sus capas que quizás sería demasiado bajo con resoluciones muy reducidas (lo que pasa para tamaños menores o bastante mayores que 227 [la resolución a la que red fue realmente entrenada por los autores[9]] [97, 98]; la métrica de validación podría seguir siendo plausible). Cabe decir que el tamaño de los lotes de entrenamiento también influye de manera similar, asimismo finalmente se decide, llegado un punto, ajustar ligeramente las capas más problemáticas (las últimas capas densas) para tratar de menoscabar lo mínimo la inferencia (de la forma más fiel que se pueda) de las conclusiones en base a los resultados.

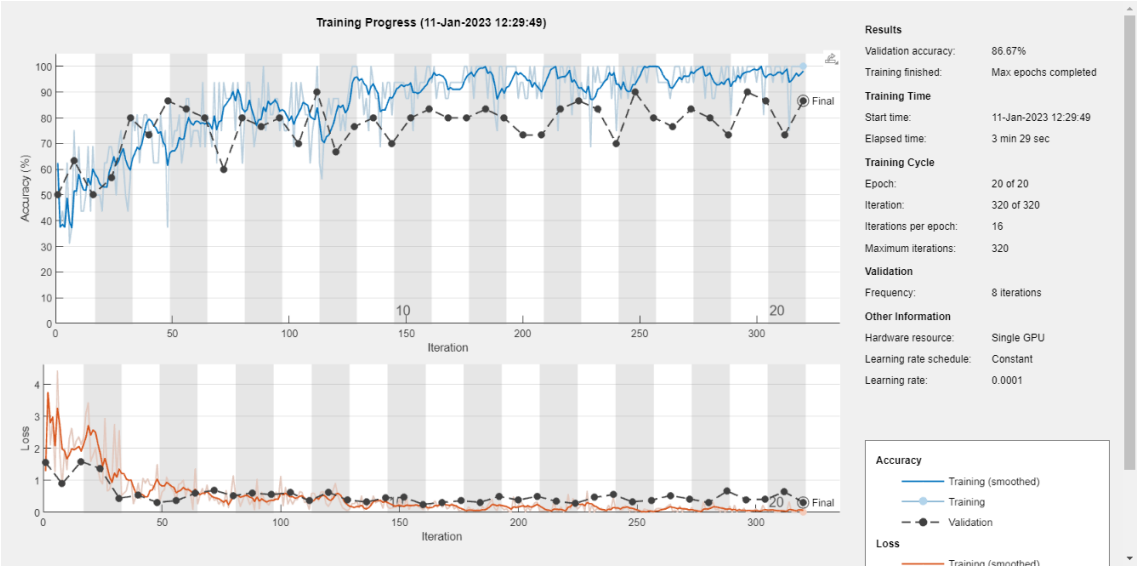

Fig. 28. Gráfica de progreso del entrenamiento con la red AlexNet modificada II, ante imágenes de 227x227 píxeles (resolución de imagen "para la que se entrenó" la red original) (86,67 % de precisión de validación).

De suerte y "ante todo pronóstico", parece que (como mínimo) la precisión de validación en el entrenamiento de la red aumenta habiendo reducido someramente el tamaño de las

[9] El documento original plantea un tamaño de entrada de 224x224x3, pero esto puede dar lugar a dimensiones erróneas después de que las imágenes atraviesen la red, por lo que lo recomendado es realizar muestreos de las imágenes de entrada para redimensionarlas a un tamaño de 227x227x3 (ver lectura nº 9, sobre arquitecturas de las RNC, de la Escuela de Ingeniería de la Universidad Stanford en el sitio web: https://www.youtube.com/watch?v=DAOcjicFr1Y&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk&index=10 [Stanford University CS231n, *Spring* 2017]).

imágenes de entrada. Más adelante se hace un último experimento en el que se aumenta considerablemente el tamaño de las muestras para comprobar si, manteniendo el mismo tamaño de lote en cada iteración de entrenamiento y el resto de opciones específicas y características de la red, la precisión aumenta o disminuye (ahora parece razonable pensar que, si no se aumenta [ni siquiera] el tamaño de los filtros de convolución y las neuronas de las capas densas u otras capas ocultas, entre otros, la red no va a aprender "lo suficientemente bien" como para que la precisión aumente causalmente).

Tabla 6. Resultados de ciertas pruebas para red AlexNet modificada II (clasificación múltiple de objetos pertenecientes a lotes o categorías diferentes).

Lote del test / Resultado	Normales	Defectuosas	Precisión (%)		
			prueba	train	valid.
80 normales	75	5	93,75	100	83,33
141 defectuosas	28	113	80,14		
80 normales y 141 defectuosas	103	118	85,07		

Por último, se procede a aumentar el tamaño de las imágenes de entrada a 512x512 píxeles introduciendo pequeños cambios en el número de lotes y las dos últimas capas totalmente conectadas (excluyendo a la que se ocupa de la clasificación [que tiene 2 salidas, en este caso]) (se aumenta el primero para contrarrestar de cierta forma el efecto del segundo y el cambio de la resolución en sí) pero manteniendo el tamaño de los filtros de convolución y demás aspectos de la estructura "base" de AlexNet, para hacer hincapié de manera concluyente en lo influyente que es el dimensionamiento de estos y otros (hiper)parámetros para un óptimo entrenamiento y comportamiento provechoso de la RNC. Se mantienen en toda instancia los 3 canales (RGB), pues no hacerlo significaría tener que hacer un cambio mucho más austero en la arquitectura de AlexNet y su entrenamiento. Una interesante alternativa para poder calificar imágenes de un tamaño superior al permitido por la red en cuestión es, tras realizar el escalado al tamaño arbitrario decidido (p. ej., 700x700 píxeles, como las imágenes del primer dataset con que se experimentó en este trabajo), asumir y programar que las entradas sean en realidad cultivos aleatorios de tamaño 227×227, con 3 canales de color, de las imágenes mayores

(se decide no hacer esto pero sería interesante ver cómo se comporta la red, aunque parece razonable pensar que los resultados se verían mejorados [casi] únicamente con relación a la detección de fallos relativamente pequeños [en comparación con los objetos con defectos más grandes y ruido o contaminación artificial más prominente]) [122, pág. 154].

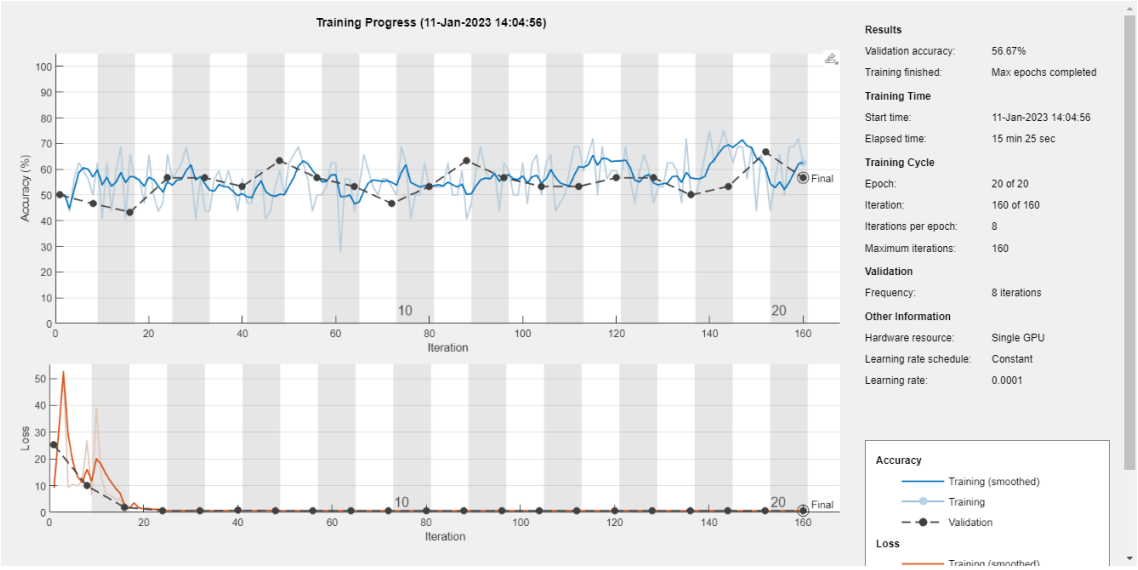

Fig. 29. Gráfica de progreso del entrenamiento con la red AlexNet modificada III, ante imágenes de 512x512 píxeles (56,67 % de precisión de validación). Los resultados de las pruebas a que se sometió esta red son notoriamente peores que los anteriores; proporcionando precisiones de 52,50 %, 54,61 % y 53,85 % (las mismas pruebas recogidas en las dos últimas tablas [en ese orden]) (precisión de 65,15 % y 56,67 % para "train" y "test" [test con datos de validación], respectivamente). Aun así, siendo inferior la cifra de la todas precisiones, parece que los resultados son uniformes, denotando la cualidad de "generalización" que otorga esta red de cara a enfrentarse a una gran variabilidad en las muestras de entrada (proporciones arbitrariamente diferentes de objetos o imágenes de clases distintas).

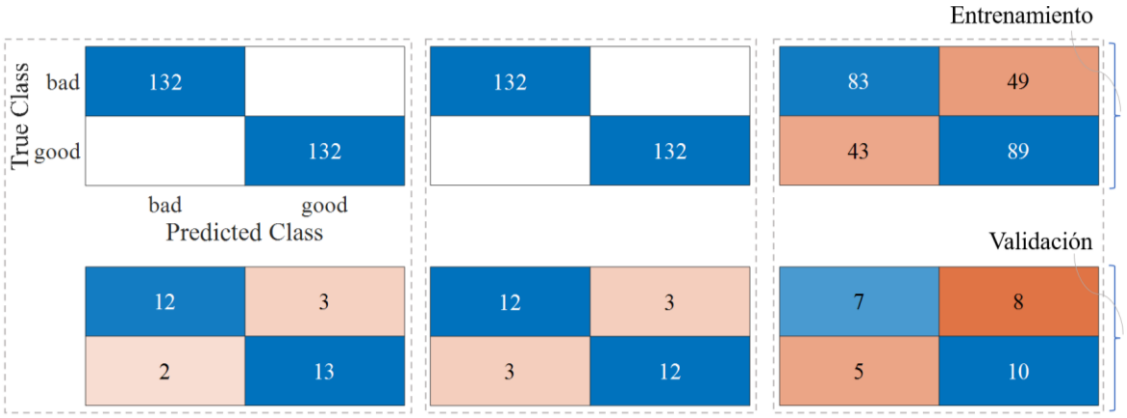

Fig. 30. Comparativa de las matrices de confusión para las redes adaptadas a los conjuntos de datos de 227x227x3 (izda.), 256x256x3 (centro) y 512x512x3 píxeles (dcha.). Aunque los resultados son complacientes para los dos primeros casos, se muestra con la gráfica de entrenamiento del segundo que ahí "tendían" a ser peores. Es concluyente (aunque se tuvo que modificar "mínimamente" la estructura de la red con respecto a los otros dos casos [el corolario se mantendría]), viendo el tercer resultado, que esta red se comporta peor siendo entrenada con imágenes de tamaños cada vez mayores (casi sin haberla inmutado con respecto de las anteriores) ("bad" se refiere a "defectuosa", "good" se refiere a "normal").

Se puede decir (analizando las últimas tres gráficas) que el entrenamiento se ha visto claramente agilizado cuando se utilizaron las imágenes de menor resolución, pues la disminución en el tamaño de la imagen es directamente proporcional a la disminución del número de parámetros que se deben ajustar en el entrenamiento de la red y, por tanto, del tiempo de cómputo dedicado al proceso. De cara a la optimización del tamaño óptimo de las imágenes de entrada se podría variar cuantitativamente la proporción de datos de entrenamiento y de validación (incluso integrar más de test independientemente), aunque realmente se suele recomendar y es típico mantener un conjunto de entrenamiento al menos 3 ó 4 veces más grande que de validación (80 frente a 20, por ejemplo). Se puede comentar también que, aunque por lo general el error de entrenamiento sea menor que el error de validación (y la precisión de validación menor que la de entrenamiento), puede llegar el caso contrario (cuando el modelo tiene muchos casos "difíciles" para aprender o cuando tiene mayoritariamente casos "fáciles" para predecir) y la diferencia entre ambos depende de factores como la validación cruzada, división de datos, métricas o parámetros de rendimiento, etc. Como la "pérdida" se calcula sobre todo el conjunto de datos tanto para entrenamiento como para validación, se puede esperar una diferencia de magnitudes

similares en la evolución; pero por el hecho de que el modelo puede empezar a sobreajustarse puede pasar que las dos pérdidas se acerquen entre sí (que pasa cuando el error de entrenamiento comienza a ser más bajo que el de validación) a medida que avanza el entrenamiento. La solución más típica (sobre todo para problemas de regresión) en caso de este tipo de sobreentrenamiento (en el que el modelo se ajuste excesivamente a los datos y no capture el comportamiento del proceso que los genera [no es capaz de generalizar]) es la regularización, que consiste en penalizar el crecimiento de la función de coste en función de los parámetros (para reducir las posibilidades del modelo de sobreentrenar) y puede servir para disminuir el número de coeficientes o parámetros adicionales por culpa (presumiblemente) del sobreentrenamiento. El motivo es que la regularización ejerce "cierta presión" que hace que ciertos coeficientes se anulen, lo cual supone directamente reducir el grado del polinomio de hipótesis (caso de regresión; no es el tratado pero surge de interés comentarlo).

En vista de estos últimos resultados (y también de los experimentos anteriores), se puede concluir vehementemente en que la precisión no es dependiente únicamente del tamaño de las entradas (ya sea en el proceso de pruebas o entrenamiento), sino que la propia arquitectura de la red, las capas de normalización por lotes, tamaño de lote, tamaño de los filtros, número de neuronas en ciertas capas, las funciones de activación, la cantidad de épocas, el formato de las muestras de entrada, la tasa de aprendizaje y actualización de los pesos, bias y el resto de parámetros, etc. son perceptiblemente influyentes en el comportamiento de la red. De manera intuitiva y cualitativa, en lo relativo al resultado de la última clasificación con imágenes de mayor tamaño, parece que a la red "los árboles no le dejan ver el bosque", pues su arquitectura original es (más o menos) óptima para las imágenes de reducido tamaño para la que fue diseñada y entrenada y a medida que el tamaño de las muestras crece, paulatinamente, el conjunto de neuronas se centra en características "tal vez" demasiado específicas y poco generales, que "distraen" a la RNC de discernir las clases debidamente. De decidir quedarse con una red (de cara a una implementación real lo ideal sería, posiblemente, diseñar desde cero [muy seguramente basándose en las fortalezas de otras redes competentes [34, 72, 82, 85, 99]] una nueva

arquitectura para adaptarse a los tipos de imágenes de que se disponga) la opción sería la red AlexNet con submuestreo de imágenes a tamaño 227x227.

Estableciendo un aparte, merece la pena destacar y comentar una técnica denominada data augmentation (aumento de datos), típicamente utilizada en visión por computador para mitigar el efecto de sobreajuste (*overfitting*), que se trata de un problemático fenómeno causado por la involucración de "ruido" indeseable y características de entrada innecesarias que desembocan en un aprendizaje inexacto debido presumiblemente a subentrenamiento (demasiados datos poco útiles o redundantes), lo que afecta al rendimiento y la precisión de la red en el sentido de que se entrena un modelo cuya salida es impracticable de generalizar a nuevos datos (lo que posiblemente ocurra [sobre todo] en los primeros experimentos). El aumento de datos, como su propio nombre indica, consiste en generar más datos de entrenamiento que los que hubiese al suponerse la presencia de sobreajuste; esto se realiza mediante la agregación aleatoria de muestras a las imágenes de entrada (p. ej.; se pueden "inventar" datos, aplicar filtros o rotar la imagen, desplazar píxeles, difuminar, rellenar bordes…), produciendo datos notoriamente más apropiados a la naturaleza ruidosa del "mundo real". Un propósito que se extrae como corolario de esta técnica es que el modelo no se encuentre con dos conjuntos de datos exactamente iguales en momentos distintos del entrenamiento (conseguir una mayor generalización en la entrada de la RNC). De acuerdo con [29, pág. 153], utilizando (solamente) esta técnica se puede lograr mejorar el comportamiento de una red para alcanzar hasta un 82 % de precisión.

Por otra parte y teniendo en cuenta como premisa que una RNC entrenada para clasificar imágenes de cierto tamaño debería de poder proporcionar mejores resultados solo para imágenes de un tamaño, por lo menos, igual o superior al correspondiente con el que se ha entrenado, se puede decir en base a la experiencia y lo observado que una RNC entrenada con imágenes de más tamaño sí que debería afectar a la precisión y la debería aumentar en la medida inversa a que disminuye la velocidad computacional del modelo. Lo que no influye en la arquitectura de la red con respecto a sus entradas es el "batch size" (ligado a las capas de normalización por lotes) o tamaño de lote (que ciertamente relacionado está con esas) el cual define el número de muestras que se dejan entrar por la

capa de entrada y que se propagan a través de la red; al utilizar un tamaño de lote menor que el tamaño de las muestras de entrada para entrenar la red (con lo que habría que entrenar cierto número de veces, "de lote en lote" hasta cumplimentar todas las muestras [90]) menos precisa y más fluctuante es el gradiente en la estimación de los pesos (la tasa de variación instantánea de los valores de los pesos; se dirigen "menos sutilmente" a la convergencia), pero requiere menos memoria (porque la red entrena [de cada vez] utilizando menos muestras) y normalmente el entrenamiento es más rápido (pues los pesos se van actualizando [con más frecuencia que con el lote completo] tras cada iteración o época), en comparación con el que resultaría de utilizar el lote de muestras completo (donde los pesos solo se actualizan una vez, al final de la única época).

Con todo esto en mente y a causa de que la red solo "ve" en su entrada una imagen de cada vez (ya sea en etapas de entrenamiento, validación o test), parece lógico que la imagen tenga que coincidir en tamaño con la forma de la capa de entrada, independientemente del número de imágenes haya en cada "lote" (este hecho se muestra en la implementación práctica realizada cuando se intenta introducir imágenes más pequeñas que el tamaño mínimo neto [de 227 píxeles] para el que la arquitectura de la red AlexNet estaba diseñada). Si en el momento de entrenar una red con múltiples imágenes o muestras del mismo tamaño o de ponerla a prueba mediante nuevas muestras se amplía (diferente a escalarla) demasiado una imagen (pudiéndose hacer luego padding para ajustarse a la capa de entrada) es probable que se pierdan detalles relevantes para la clasificación y también una visión general (en escena) del objeto, patrón o elemento que se quiere reconocer, pero si la imagen se reduce demasiado también podría causar efectos perniciosos debido a que el objeto se pierda en la naturaleza ruidosa de la escena y se contemplen excesivos detalles innecesarios que "confundan" a la red; todo depende del tipo de problema que se esté tratando o trabajando. Se puede entrever también la manera de la que influye en la precisión de clasificación (validación) de la red la variación de las imágenes con que se entrena, como ya se concluye en las pruebas realizadas tras sesgar el tamaño de los objetos sin defectos (mayores lotes, aunque más pesados computacionalmente, si contuviesen imágenes "verdaderamente útiles" para el

entrenamiento y la clasificación, mejorarían sin duda alguna la precisión) (análogo a la influencia de los lotes en la evolución del gradiente de los parámetros).

Es interesante mencionar la existencia de un concepto más, el fenómeno *Imbalanced data* (datos desequilibrados o no balanceados); este es otro problema común en el aprendizaje automático (que aparece eventualmente más en la rama del *clustering*) que resulta en una traba a la hora de extraer y correlacionar características y patrones, separar datos en categorías o clases y evaluarlos. Aparece cuando el número de muestras o datos de entrada que se asocian a cada clase es muy variable, provocando escenarios en los que, aun habiendo dos clases semántica y claramente diferenciadas, las muestras asociadas a una clase sean extravagantemente inferiores a los de la otra con tan mala suerte de estar además suficientemente "cercanos" entre sí como para que el modelo de clasificación cometa una equivocación al separarlos y mezcle los dos tipos de muestras, estableciendo sus propias clases erróneas (problema que se acentúa cuando las características más representativas de las clases son difíciles de encontrar). Podría haber aparecido en las experimentaciones realizadas si no se hubiera tenido el cuidado de normalizar el número de imágenes de cada clase a un mismo valor, aunque por otra parte se ve ciertamente reflejado al condicionar al lote de imágenes sin defectos a ser la que se "sesgue" y adapte al otro, pues haciendo esto la red puede que "aprenda de manera menos general" a clasificar objetos sin defectos que objetos defectuosos.

Existen varias formas eficaces de desvanecer el mencionado problema: introducir una mayor ponderación en las clases más pequeñas, transformar los propios datos mediante submuestreo (eliminar datos no deseados o repetidos en la clase mayoritaria y sesgar el conjunto a una parte útil reducida) y sobremuestreo (obtener más datos, preferiblemente no repetidos, en la clase minoritaria) o recurrir a alguna métrica de evaluación arbitraria diferentes. Pueden incluso combinarse distintos métodos y someter los datos de entrada a un submuestreo (menguar notablemente la proporción de los datos de la clase predominante) y luego, tras el entrenamiento, agregar un factor de ponderación a la clase submuestreada igual al valor en que se hubiese disminuido primeramente. Existen algoritmos específicamente diseñados para lidiar con este tipo de problemas, como RUSBoost, que integra RUS (submuestreo aleatorio) más el procedimiento de Adaboost

(algoritmo metaheurístico de clasificación estadística utilizado recurrentemente con otros modelos de aprendizaje [“clasificadores débiles”] para mejorar el rendimiento mediante la aplicación a sus salidas de un sumador ponderado que potencia la clasificación [“clasificador fuerte”] [91]) para mejorar la clase minoritaria eliminando muestras de la mayoritaria o SMOTEBoost (similar al anterior pero con un sobremuestreo estadístico único de minorías sintéticas [SMOTE] implementado de tal manera que mejora el rendimiento con respecto a la mejora en los datos minoritarios con cada iteración de refuerzo también gracias a Adaboot [92]). De acuerdo con [93], una clasificación del conjunto de datos “Cover type” (dataset multivariable y categórico de diferentes tipos de bosques [94]) realizada mediante redes neuronales puede mejorar su precisión de clasificación alrededor de un 10 % empleando la técnica RUSBoost.

Breve comentario sobre la relación entre la visión artificial y las RNC

Los sistemas de visión artificial (por computador) y procesamiento de imagen se utilizan cada vez más frecuentemente, desde hace lustros, en aplicaciones dentro del ámbito de los procesos industriales, automatización y robótica (entre muchos otros) debido a que su recurrente combinación con paradigmas y algoritmia de IA (donde destacan las RNA, los algoritmos adaptativos, genéticos y la lógica borrosa) denota una relación muy estrecha entre ambos, dirigida al desarrollo de sistemas de control muy flexibles y adaptables a todo tipo de problemas y tareas, particularmente de cara a la investigación [25, pág. 53].

Es de elevadísimo interés mencionar varias formas de las que un sistema de refuerzo por técnicas de visión artificial y procesamiento de imagen podrían ayudar a esta red con relación a la mejoría en la clasificación de imágenes. Es patente que los filtros de convolución (típicamente bidimensionales) se usen de manera similar en el análisis de imágenes y en las RNC, pues la técnica de convolución es útil para extraer características específicas de los datos de entrada, mas si se quiere dar un "enfoque de visión" al modelo neuronal son ampliamente diversos los tipos de filtros que se pueden emplear: filtros de suavizado Gaussianos, primera (Prewitt, Sobel, Scharr, derivada de la gaussiana…) o segunda derivada (LoG, DoG…), de mediana, adaptativos, media no local, máscara rotatoria, etc. Más aún, considerando que "la convolución en el dominio temporal [o espacial, si se trata con imágenes] es equivalente a la multiplicación en el dominio frecuencial", se puede recurrir a filtros que se apliquen en el dominio de frecuencias de los datos (al igual que se hace en el procesamiento de señales tras aplicar la FFT, DFT…), sean filtros "pasobajos", "pasoaltos", "pasobanda", "rechazo de banda", sinc, Buttherworth, Chebyshev, Lanczos, etc. En definitiva, los filtros de convolución son la clave para la extracción primera, potenciación o el destaque de las características más "acordes" a las imágenes que se quiere clasificar; es seguro que este tema precisa de un análisis particular de cara a implementaciones rigurosas. Es plausible incluso modelizar los datos de entrada de la red neuronal emulando el proceso de percepción visual (en los humanos: recepción visual de información y tratamiento previo hasta la formación de un precepto que revela ciertas estructuras caracterizadoras de lo percibido [108, pág. 249]) para perfeccionar así el sistema inteligente de clasificación, es decir, implementando

técnicas y algoritmos de visión artificial en etapas de formación de imágenes (SURF, submuestreo y sobremuestreo, digitalización y transformación de imágenes planas a partir de otras tridimensionales [109, 110]…), preprocesamiento (realzado o suavizado de bordes o contornos…), segmentación (computación de gradientes, umbralizaciones, extracción de bordes mediante filtros de primera derivada…) u otras extracciones y representaciones de características (determinación de orientación, tamaño o momentos [cartesianos, centrales, centrales normalizados, complejos de Zernike, de Hu…] de regiones o su binarización, descripción de líneas mediante la transformada de Hough…), entre otras.

Es fácil diferenciar entre dos tipos de sistemas de reconocimiento de imágenes, uno que contiene un extractor de características (que transforma la imagen en un vector paramétrico, donde cada uno de sus componentes se corresponde con una característica específica) y un clasificador (sistemas como el perceptrón de Rosenblatt de tres capas, el neocognitrón de Fukushima, la red LeNet de Yan LeCun, o el MWC de Hoque [105]) y otro que contiene solo un clasificador (aunque realmente puedan tener extractores de características pero incorporados en la estructura interna del clasificador). En [106] se propone un clasificador LIRA que contiene extractores de características invariantes (las características no cambian durante el proceso de entrenamiento) (opuestos a los adaptativos) que introduce el vector paramétrico en un perceptrón de una capa para realizar la clasificación. El clasificador LIRA tiene una estructura similar al perceptrón multinivel pero con un rendimiento mejorado, gracias a ciertas modificaciones relacionadas con el tamaño del área de la imagen que se utiliza para extraer cada característica (lo que sería semejable al tamaño del filtro de convolución de las capas iniciales de una RNC) para obtener las características locales, además de la posibilidad de extraer y aprender información particularmente de la escala de grises de la imagen.

Las “reglas” para la elección de filtros de procesamiento de imagen previos a la clasificación, de modo que ayuden como extractores de características, están totalmente abiertas, así pues depende íntegramente del tipo de problema a resolver y de las imágenes a tratar el decidir si se emplean o no unos tipos u otros de filtros. Para el caso concreto que ocupa a este trabajo, puede ser interesante aplicar cierto suavizado a las imágenes de

entrada para facilitar de cierta forma la percepción de los defectos más prominentes, aumentar la intensidad de los contornos de aquellos objetos que tiendan a presentar defectos más discretos o incluso normalizar la escala de grises (o más bien la intensidad de cada canal de color) para no "confundir" a la red con iluminaciones engañosas y perjudiciales. La experimentación con estas técnicas de modificación de píxeles podría aplicarse para los datos de entrada en "producción normal", los de entrenamiento (y/o validación, test…) o para ambos casos; cada opción puede tener sus ventajas e inconvenientes, que habría que analizarlos detenidamente en función de la tarea a resolver y del comportamiento que se le quiera dar a la red neuronal. Más adelante se hace un breve compendio de algunas mejoras en el sistema de clasificación presentado en los anteriores apartados, que integran alguna que otra sencilla técnica de procesamiento de imagen para aunar de manera práctica, puntual y circunstancial parte de lo comentado en este apartado.

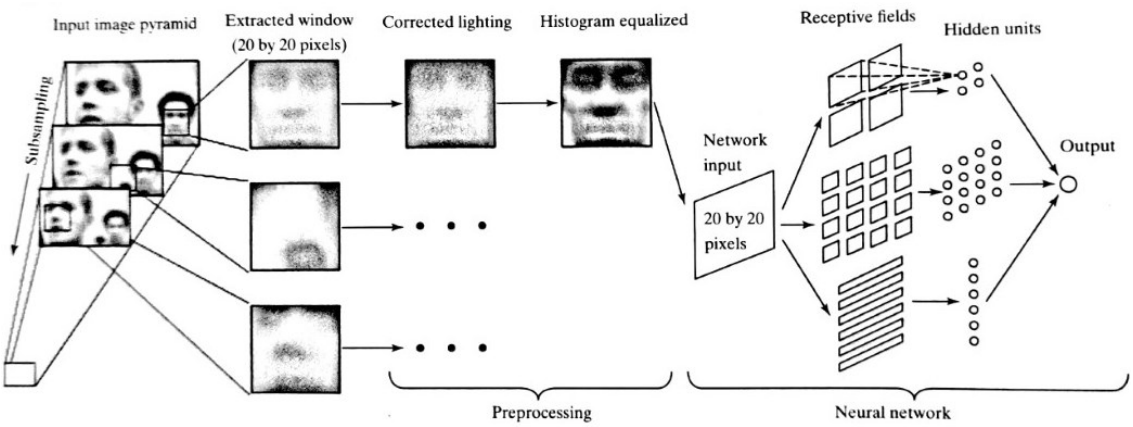

Fig. 31. Red neuronal de Rowley-Baluja-Kanade para detección de rostros [95], que aplica una arquitectura feedforward a ventanas en escala de grises de tamaño 20x20 con corrección de la iluminación (recogido de [38, pág. 610]).

En el ejemplo que se ve en la anterior figura, el preprocesamiento normaliza la escala de grises de la imagen de entrada para disminuir el efecto de variaciones en la intensidad demasiado irregulares y corregir los contrastes bajos, luego las ventanas que recorren la imagen pasan a través de las capas ocultas de la red, que están especialmente diseñadas para adaptarse de manera óptima a varios rasgos y patrones identificativos del rostro humano. El entrenamiento de la red se realiza mediante backpropagation y un algoritmo

denominado *bootstrap*, que consiste en realimentar el modelo con falsos positivos para reducir la frecuencia de tales fallos. Además, el algoritmo detecta rostros a múltiples escalas pues el entrenamiento se habría realizado procesando una pirámide de imágenes sucesivamente submuestreadas partiendo de la original, muestreando con la ventana en todas las localizaciones. La combinación de estos enfoques a la salida de la red proporciona un resultado que en poco se diferencia de bastantes métodos modernos actuales de reconocimiento de rostros, por lo que se puede extraer una conclusión muy determinante en base a esta red: dos de los aspectos más significativos de un detector artificial son los datos de entrenamiento y las características que se extraen de estos, siendo posiblemente más relevantes que el algoritmo particular usado para la clasificación o el entrenamiento [38, pág. 611] [95].

Un enfoque ciertamente más práctico se consigue en [116], donde los autores utilizan una búsqueda selectiva donde se emplean técnicas tradicionales de visión por computador para encontrar áreas en la imagen que puedan obtener objetos. Originariamente, trabajos anteriores y similares propusieron nombrar tales áreas como "Region Proposal"; la red (más básica) utilizada para detectarlas se denomina actualmente R-CNN o R-RNC. En la "R-CNN" original [117, 118] estas regiones se extraían mediante algoritmos externos, se redimensionaban y se introducían en una RNC o un tipo de SVM para obtener vectores de imagen y clasificar los objetos contenidos en cada región, antes de corregir los recuadros delimitadores propuestos por la herramienta externa mediante una red de regresión lineal sobre los vectores de la imagen. Los modelos (posteriores) basados en R-CNN [119] utilizan este tipo de búsqueda selectiva de formas más inteligentes para extraer ciertas ROI, a partir de las cuales se computan una serie de características mediante RNC y se realiza la clasificación (otros modelos derivados como Fast R-CNN [unas veinticinco veces más rápida] o Faster R-CNN [unas diez veces más rápida que la precedente] se propusieron a partir de este). Por ejemplo, la red Faster R-CNN elimina el mecanismo externo de propuesta de regiones y lo sustituye por un componente (sistema neuronal) entrenable (inteligente), denominado RPN[10], dentro de la propia red; luego su

[10] De hecho, de acuerdo con [123], las redes (feedforward) multicapa son una clase de aproximadores universales, pues con tan sólo una capa oculta y utilizando funciones de activación o aplastamiento arbitrarias son capaces de aproximar cualquier función medible de Borel (el predecesor inmediato de

salida se combina con el mapa de características y se hace pasar por un proceso similar al de la red Fast R-CNN (es también interesante la red YOLO).

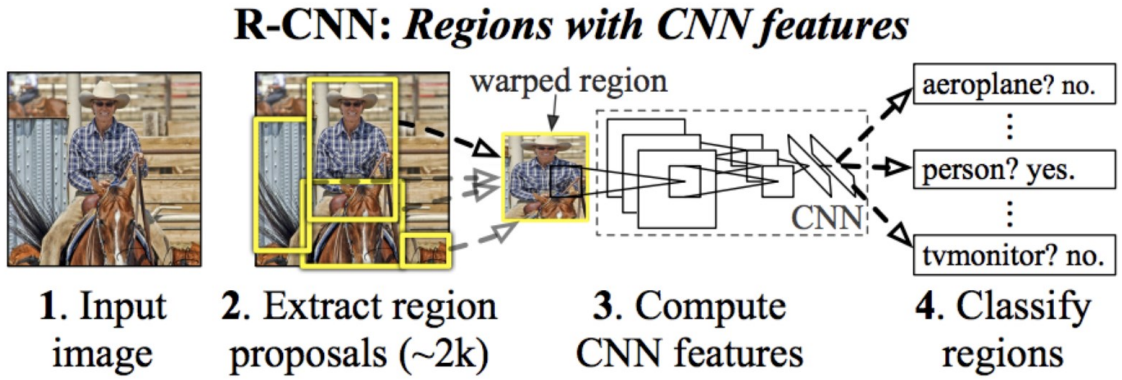

Fig. 32. Método de trabajo de las redes R-CNN (recogido de [117]). Cada región de interés se alimenta a través de una capa de agrupación de otras varias y, a continuación, a una red totalmente conectada que produce un vector de características para cada ROI. Este es un ejemplo claro de la conjunción de técnicas tradicionales de visión por computador y redes neuronales, aunque ya se puede sospechar (y cierto es) que "toda" la labor puede ser llevada a cabo por las últimas (concretamente por una Faster R-CNN, que constituye una de las bases de esta rama de estudio dentro de la IA y las RNA).

En la implementación práctica realizada con MATLAB se ha decidido realizar cierta etapa de preprocesamiento de imagen para extraer las características más determinantes que "explican" los defectos que pudieran tener los objetos a clasificar, cuidando de no modificar "perceptiblemente" en tal procesamiento la bondad de las imágenes sin defectos. Para esto se aplican técnicas tradicionales (de manipulación del píxel mediante filtros, típicamente) de visión artificial tanto a la imágenes de entrenamiento (y validación) como a las de test; esto no es realmente obligatorio porque podría aplicarse a uno de ambos conjuntos, pero el resultado podría ser impredecible si no se analiza el caso de uso con sumo detenimiento y particularizado a los objetos concretos, por lo que se decide ahorrar tiempo (al tratarse este trabajo de una introducción a este campo de estudio) y modificar concurrentemente todos los subconjuntos de datos.

Se decide realizar varias operaciones a todas las imágenes y por igual (implementado en un programa o *script* ".mlx" independiente): se hace un ajuste de los valores de intensidad

Lebesgue en la teoría de la medida [124]) de un espacio dimensional finito a otro con cualquier grado de precisión deseado, siempre que se disponga de un número suficiente de unidades ocultas (neuronas).

de cada canal de color mediante el mapeo de los valores de intensidad [límites de contraste] de la imagen de partida a unos nuevos valores en la imagen de salida (función *imadjust*; se busca aumentar por lo general el contraste y, si acaso, mantener un canal de color modestamente inferior a los otros dos para ayudar a distinguir mejor los defectos a través de una frecuencia concreta [la del rojo, p. ej.]), se computa un filtro no lineal tridimensional (en todo el canal RGB, como el resto de técnicas) de mediana para eliminar ruido impulsional (valores atípicos, que pueden ser conceptualmente menos interesantes que el ruido blanco de cara a un aprendizaje más completo de la RNA) preservando todo lo posible los bordes y pudiendo conseguir así destacar más uniformemente ciertos defectos como deformaciones o magulladuras (función *medfilt3*, especificando un filtro relativamente pequeño de 5x5 para no influir demasiado en la presencia de los defectos más pequeños; se replican los valores en los bordes de la imagen de forma reflejada), se hace un filtrado "de moda" con el que se sustituye cada pixel por la moda de los valores de su vecindad (vecindad 7x7 [p. ej.], valor relativamente grande elegido empíricamente) (función *modefilt*; puede ser interesante para restar peso a la computación del entrenamiento y además tiene un propósito muy similar, sino totalmente semejable, al de las capas max-pooling de las RNC [preservar solamente "las características más características" de cada ventana de análisis para reforzar el proceso categórico de datos]) (de nuevo se replican los valores en los bordes de la imagen de forma reflejada) y, por último, se realza la "vasculatura" (las estructuras alargadas o tubulares) de la imagen mediante un filtro de Frangi (*Hessian-based multiscale Frangi vesselness filter*, implementado con la función *fibermetric*), a fin de destacar defectos alargados como ralladuras o roturas pequeñas pero alargadas.

Como se puede ver en la siguiente imagen, el mapa de características de la imagen con defectos y sin ellos se ve notablemente simplificado, con lo que la red (como puede parecer a priori, pues para un humano así ocurre) puede tender a aprender mejor si se utilizan los métodos de procesamiento adecuados para resaltar y extraer, específicamente, "lo determinante" para la clasificación.

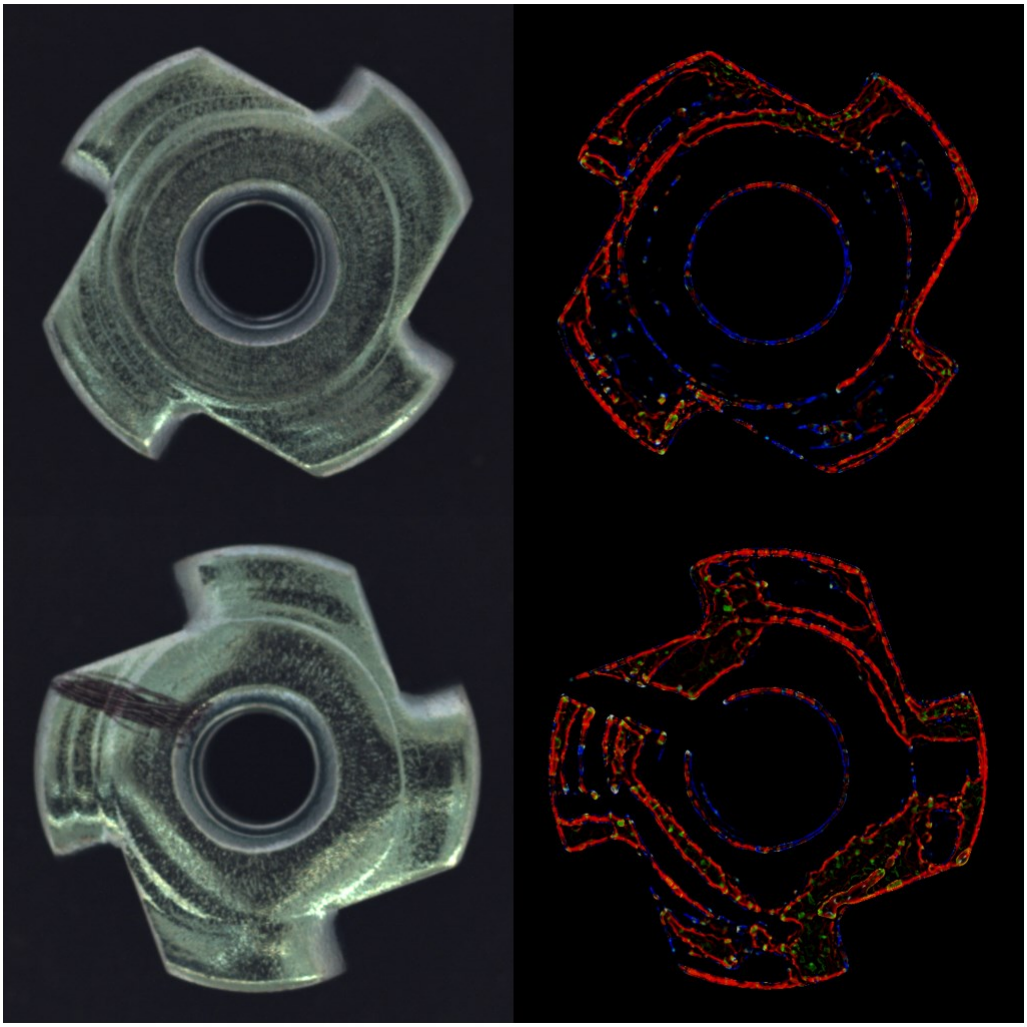

Fig. 33. De izquierda a derecha y de arriba abajo se muestra una imagen sin defectos antes y después del preprocesamiento y otra imagen defectuosa arbitraria antes y después del preprocesamiento, respectivamente. El propósito con las técnicas de visión artificial es "remarcar" en la imagen la presencia de defectos en caso de que existan y, de lo contrario, resaltar la cualidad de homogeneidad (poca variabilidad en el conjunto útil de píxeles de que se compone el objeto dentro de la escena) que caracteriza a los objetos sin defectos o normales. De hecho, la red incluso clasifica "medianamente" bien imágenes de entrada que estén sin preprocesar (esto es, en efecto, un sistema inteligente).

Se ve que las imágenes sin defectos estarían formadas, por lo general, de una especie de toroide plano relativamente homogéneo rodeado de cuatro regiones simétricas con apenas diferencias entre sí, por el contrario, puede verse cómo la imagen defectuosa (que es clasificada por la propia red como tal) presenta una clara desligadura entre las zonas mencionadas que "rompe" con el arquetipo de "objeto sin defectos" que se supone aprende a reconocer la red neuronal a lo largo de su entrenamiento.

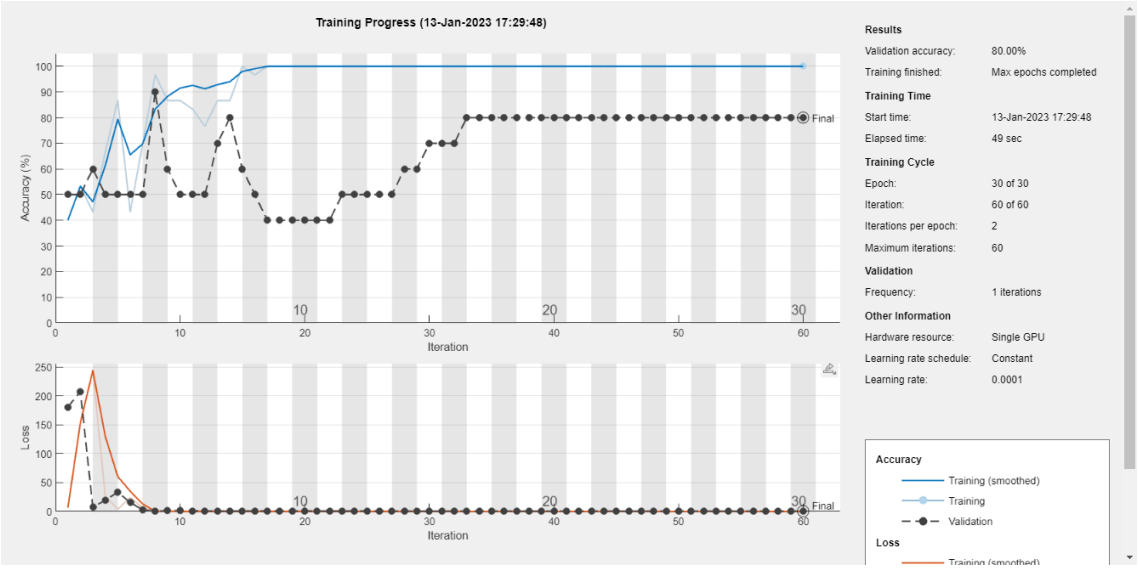

Fig. 34. Gráfica de progreso del entrenamiento tras la etapa de preprocesamiento de imágenes. En comparación con la gráfica del entrenamiento I (donde se muestra un 60 % de precisión de validación), el aumento de precisión es visible (ahora se muestra un 70 % de error de validación, aun con una arquitectura tan sencilla y simplemente ilustrativa) y, desde luego, no casual, sino influenciado por las mejoras que el preprocesamiento añade al aprendizaje de la red.

Tabla 7. Resultados de ciertas pruebas tras la etapa de preprocesamiento de imágenes.

Lote del test / Resultado	Normales	Defectuosas	Precisión (%)		
			prueba	train	valid.
20 normales	12	8	60	100	80
45 defectuosas	16	29	64,44		
20 normales y 45 defectuosas	28	37	63,08		

Aunque (casi irrelevantemente) el primer test resultase mejor para el entrenamiento I que para este último tras preprocesamiento, todos los demás resultados mejoran, de suerte, lo suficiente para poder sentenciar que una técnica de “normalización” de imágenes previa a la entrada de la red, o la potenciación de ciertas características e incluso el enfoque de regiones de interés en la imagen que se supone contienen los defectos buscados, ayuda a que la RNC “sepa generalizar” mejor el discernimiento de categorías y, de cierta forma, aprender mejor ante la detección de los patrones y las características “que realmente

importan". También se puede concluir, refiriéndose a todos los experimentos realizados, resaltando que las imágenes y los objetos que se quieren clasificar, aunque separados en dos categorías semánticamente opuestas, son suficientemente similares entre sí (p. ej., los defectos no resaltan exageradamente ni son muy grandes) para hacer de este problema de clasificación una tarea especialmente complicada de solventar, en comparación con otro tipo de clasificaciones con mayor contraste entre las categorías a predecir. Sin más complicación, un enunciado sumamente más sencillo, y que sin modificación alguna en las redes neuronales artificiales más "básicas" que se presentan en los primeros experimentaciones devolvería rendimientos y precisiones mucho más "perfectos", se correspondería a la tarea de clasificar los objetos pertenecientes a los cuatro datasets con que se trabaja (es decir, clasificar los tipos de objetos particulares, que son "mucho" más diferentes entre sí que los pertenecientes a los subconjuntos abstractos "defectuoso" y "no defectuoso" englobados en cada conjunto "objeto" [la variabilidad es también mucho menor en este supuesto más sencillo]).

De todas maneras y concluyentemente, habiendo visto todo lo que se ha tratado y como se comenta en [111, pág. 726], puede ser "más práctico" para una primera mejor "aproximación", ya que se está tratando de abordar la comprensión y explotación de las posibilidades que brindan las RNC a la clasificación de imágenes y particularmente la detección de defectos, configurar y entrenar una red neuronal con un número aceptablemente cuantioso de muestras levemente distorsionadas con ruido antes que tratar de eliminar manualmente el ruido de otras imágenes en la entrada de la red a la hora de "ponerla en funcionamiento". Esto es así porque, simple y llanamente, las redes neuronales artificiales tienen la capacidad de aprender a hacerlo, siempre que se cuente con un conjunto de imágenes suficientemente variado y minucioso para el entrenamiento (por parte de todas las categorías). Alternativamente, puede ser posible (cabría comprobarlo) que estructuras de redes notablemente más sencillas (por ejemplo, con una sola capa oculta o pocas más), dado el relativamente bajo número de imágenes de muestra disponibles para el entrenamiento, proporcionasen comportamientos y predicciones generalmente más ajustadas, aunque quedasen algo alejadas del aprendizaje profundo.

Los resultados, a la vista de la anterior tabla y en comparación con la primera (del entrenamiento I), son absolutamente prometedores, pues se puede inducir que "existe" cierta manera de la que la visión artificial podría ayudar aún más (los métodos implementados son complacientes mas, cómo no, mejorables, escalables y ampliables) a mejorar el rendimiento y la precisión (además de la generalización ante entradas) de la red neuronal convolucional. Los resultados finales, lejos de ser completamente definitorios según un criterio común de cara a implementar la red en una aplicación real, dejan poco que desear en lo relativo a la compresión de la inmensa cantidad de posibilidades que ofrecen los sistemas inteligentes específicos de esta rama del conocimiento tan sorprendente que es el aprendizaje automático.

Conclusiones

Si las redes neuronales se enfocaron, hasta hace relativamente poco, en la forma en que los humanos piensan, con el aprendizaje profundo (deep learning) el alcance se amplía para convertir unas herramientas restringidas al campo de la investigación específica en potentes e incondicionales métodos de negocio, por ejemplo. La mejora creciente en potencia de cómputo para simular las capas de las RNA, subida en la cantidad de datos generados por Internet para (aún más) óptimas selecciones de características, introducción de unidades de procesamiento gráfico de vanguardia para priorizar de manera eficiente las tareas aritméticas (más operaciones con mayor precisión, subprocesos pseudoconcurrentes, etc.)… influyen positivamente en el avance del aprendizaje profundo que esencialmente se relaciona con la creación de más capas adicionales en de las RNA regulares, pues un mayor número de capas, si equivale a un mayor número de conexiones y de mejor o igual calidad, dirige proporcionalmente a los sistemas a ser más inteligentes (aumenta el número de características, factores o patrones adicionales que se pueden analizar para buscar respuestas deseables). Sin embargo, el hecho de mejorar los resultados simplemente escalando el número de datos no es algo que se cumpla de manera ilimitada con los métodos de aprendizaje tradicional (tienen una capacidad finita de aprendizaje independientemente de cuántos datos adquieran), al contrario de lo que pasa con las posibilidades que ofrece el aprendizaje profundo. Por añadidura, el hecho de que las redes profundas proporcionen resultados fehacientes de manera no necesariamente supervisada (sin el suministro a priori de datos etiquetados manualmente) ensalza la posibilidad de aproximarse asintóticamente a la modelización ideal la inteligencia humana y la explotación máxime de los beneficios de otros conceptos tan en auge como el Big Data o la Industria 4.0 (donde se gestionan y producen cantidades ingentes de datos, que no serían manejables con IA menos profunda) [96, págs. 53, 60]. Un claro ejemplo se ve en algunos resultados extraídos de la experimentación realizada; aunque se estuviera aumentando eventualmente la profundidad y la complejidad de las redes neuronales (de forma “cruda”, es decir, agregando capas o neuronas directamente a la estructura “básica”), los resultados parece que no eran más generales, ajustados o veraces puesto que existía un problema de más peso, que era una pobre elección en las imágenes de entrenamiento dada la complejidad innata del problema al que se estaba

enfrentando. Como se comenta en múltiples ocasiones en este documento, es vital la correcta selección de muestras, partición de conjuntos, el número de imágenes de muestra disponibles, etc., obteniendo aun así (final y ocasionalmente) unos resultados prometedores, pues se puede inducir que "existe" cierta manera de la que la visión artificial podría ayudar aún más a mejorar el rendimiento y la precisión de la red neuronal convolucional, entre otros. Cabe decir que decidir entrenar la red con la "mitad inferior" de las imágenes defectuosas que proporcionan los datasets y utilizar la "mitad superior" para establecer los lotes que se usan en las pruebas de precisión de la red es una decisión algo austera o estricta en el sentido de que se priva a la red de aprender con la suficiente certeza la variabilidad de los objetos defectuosos; con las simple decisión alternativa de escoger un menor tamaño de lote para los datos de "train" y mayor para los de entrenamiento (concretamente, para los del número de objetos con defectos) el resultado puede mejorar sensiblemente. Como se comenta al final de "Experimentación particular con las RNC y validación descriptiva", podría recurrirse a técnicas como el sobremuestreo de las imágenes defectuosas para tratar de solventar este problema sobrellevando el hecho de que no se dispone de la capacidad de hacer más fotografías similares para aumentar el lote de muestras.

A lo largo de la programación de las pruebas se recurren a numerosas funciones de MATLAB que surgieron de interés en el proceso de investigación de las funcionalidades típicas de recursos como "Deep Learning Toolbox", pero se pudieron haber contemplado opciones como *Deep Network Designer*, *nprtool* u otras alternativas de la misma forma de la que se particularizó el estudio y la implementación de las que se emplean. Otras opciones que se pudieron contemplar para la resolución de este tipo de problemas en detección de fallos y anomalías abarcarían algoritmos tanto de aprendizaje supervisado (árboles de decisión, clasificadores de soporte vectorial, otras técnicas que se engloben dentro del "Soft Computing" [redes bayesianas, algoritmos evolutivos, lógica *fuzzy*]…) como no supervisado (*one-class classifiers*, *deep autoencoders*…), implementables en otros lenguajes como Python, R o C++ con ayuda de librerías y bibliotecas como scikit-learn, Tensorflow, PyTorch y Keras. Más aún, con relación al trabajo práctico llevado a cabo, ya que los resultados varían en función del problema particular tratado podría ser

útil recurrir a varias métricas de comportamiento distintas (no solamente matriz de precisión, rendimientos y resultados puntuales…) para ofrecer una mejor y mayor perspectiva para evaluar el comportamiento de los modelos de clasificación, como matrices de confusión [ya usadas], exhaustividad (*recall*) (informa sobre la proporción de datos pertenecientes [desde la entrada] a cierta clase que se clasifican correctamente [implementado mediante una función propia]), precisión estadística (informa sobre la proporción de datos que el modelo clasifica como pertenecientes [desde la salida] a cierta clase que se clasifican correctamente) o "medida F" ("F1-score") (expresa de cierta manera el rendimiento del modelo, que es igual al cociente entre el doble del valor de la precisión en un producto con el recall y la suma de esos dos parámetros). Incluso para obtener mejores resultados en la evaluación se puede probar a ajustar los datos de entrada a diferentes modelos de aprendizaje automático, sean algoritmos de ML híbridos o de conjunto (Adaboost, árboles de decisión…) u otros paradigmas de DL.

Aunque numerosos campos específicos y extrapolables a problemas de índole industrial se pueden explorar y explotar con el estudio de las redes neuronales de aprendizaje profundo y otras técnicas, como el análisis de vibraciones en motores (análogo al reconocimiento de voz) o el diagnóstico visual en el control de procesos (análogo al reconocimiento facial o de expresiones), el DL en el ámbito industrial está ciertamente abierto y es novedoso hoy día. Los trabajos y estudios realizados en muchos ámbitos de la IA (especialmente las redes convolucionales, feedforward, deep autoencoders [capaces de transformar a su salida la representación de las señales de entrada sin modificar la información útil contenida en estas] y recurrentes) pueden trasladarse al sector industrial si se cuenta con la debida imaginación, lo que constituye un importante campo de estudio en vista del futuro[11].

[11] Recogido del artículo de Anthony Wing Kosner "Tech 2015: Deep Learning and Machine Intelligence Will Eat the World" (2014) [Online], revista Forbes. Disponible en: https://www.forbes.com/sites/anthonykosner/2014/12/29/tech-2015-deep-learning-and-machine-intelligence-will-eat-the-world/?sh=44a759a25d94.

Referencias, bibliografía y lecturas recomendadas

[1] J. C. Ponce Gallegos et al., Inteligencia Artificial, 1a ed., LATIn, 2014.

[2] Bertrand Russell, "Theory of Knowledge" (para La Enciclopedia Británica), 1926.

[3] Mendel, J.M., & Mclaren, R.W. (1994). Reinforcement-learning control and pattern recognition systems. Mathematics in science and engineering, 66, 287-318.

[4] Simon Haykin, Neural Networks: A Comprehensive Foundation, Pearson Education (US) (7 agosto 1997), ISBN-13 978-0138958633.

[5] Descartes, René. 1637. Discourse on method. Trans. John Cottingham, Robert Stoothoff and Dugald Murdoch. In *The philosophical writings of Descartes*, Vol. I, 109-151. New York: Cambridge University Press.

[6] Hilera, J. & Martínez Hernando, V. (1995). Redes neuronales artificiales: fundamentos, modelos y aplicaciones.

[7] Kulstad, Mark y Laurence Carlin, "Leibniz's Philosophy of Mind", The Stanford Encyclopedia of Philosophy (Winter 2020 Edition), Edward N. Zalta (editor).

[8] Isaac Asimov, Runaround, *Stounding Science Fiction*, 1941.

[9] Hubert L. Dreyfus, What Computers Can't Do: The Limits of Artificial Intelligence, 1978, HarperCollins, ISBN-13 978-0060906139.

[10] Roger Penrose, The Emperor's New Mind «Concerning Computers, Minds, and The Laws of Physics», 1ª ed. Oxford: Oxford University Press, 1989.

[11] Cole, David, "The Chinese Room Argument", The Stanford Encyclopedia of Philosophy (Winter 2020 Edition), Edward N. Zalta (editor).

[12] Searle, John. 1980a. "Minds, Brains, and Programs." Behavioral and Brain Sciences.

[13] Bruno Siciliano, Oussama Khatib (editores), Springer Handbook of Robotics (Introducción pág. 1) 2008, Springer, ISBN 978-3-540-23957-4.

[14] M. A. Delgado, Yo y la energía (Nikola Tesla), 10a ed., 2011, Editorial Turner, ISBN 978-84-7506-293-8.

[15] Andrew P. Sage, James D. Palmer, Software Systems Engineering, 1990, Wiley-Interscience, ISBN 10047161758X.

[16] Foundations and Grand Challenges of Artificial Intelligence. Reddy, R. Winter, 1988, AI Magazine, p. 9.

[17] Andries P. Engelbrecht, Computational Intelligence «An Introduction», 2a ed., Wiley, 2007, ISBN 978-0-470-03561-0.

[18] Turing, Alan, "Computing Machinery and Intelligence (1950)", in B J Copeland (ed.), The Essential Turing (Oxford, 2004; online ed., Oxford Academic, 12 Nov. 2020).

[19] Noam Chomsky, Language and Mind, 2006, 3a ed., Cambridge University Press, ISBN 9780521674935.

[20] W.W. McCulloch and W. Pitts, A Logical Calculus of the Ideas Inminent in Nervous Activity, Bulletin of Mathematical Biophysics, 5:115-133, 1943.

[21] W. Pitts and W.W. McCulloch, How We Know Universals, Bulletin of Mathematical Biophysics, 9:127-147, 1947.

[22] Ashby, William Ross, "An introduction to cybernetics", Chapman and Hall, 1965.

[23] Hans Drischel, Einführung in die Biokybernetik (Introducción a la biocibernética), Akademie-Verlag, 1972.

[24] Norbert Wiener, Cybernetics: Or Control and Communication in the Animal and the Machine. Paris, 2a ed., 1961, (1a ed. en 1948), (Hermann & Cie) & Camb. Mass. (MIT Press) ISBN 978-0-262-73009-9.

[25] Tatiana Baidyk y Ernst Kussul, Redes neuronales «visión computacional y micromecánica», 2009, Editorial Itaca, ISBN-13 9786072000216.

[26] Feigenbaum, E.A. (1963) Computers and Thought. In: E. A. Feigenbaum and J. Feldman Eds., A Collection of Articles, McGraw-Hill Book Co., New York, 477-523.

[27] McCullough, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), 115–133.

[28] Basak, Jayanta & Murthy, Chaitra & Chaudhury, Santanu & Majumder, Dwijesh. (1993). A Connectionist Model for Category Perception: Theory and Implementation. Neural Networks, *IEEE Transactions on Neural Networks*. 4. 257-269. 10.1109/72.207613.

[29] Francis Chollet, *Deep learning with python*, 2018, Manning Publications Co., ISBN 9781617294433.

[30] Morris RG. D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949. Brain Res Bull. 1999 Nov-Dec;50(5-6):437. doi: 10.1016/s0361-9230(99)00182-3. PMID: 10643472.

[31] Haykin, S. (1999) Neural Networks: A Comprehensive Foundation. 2nd Edition, Prentice- Hall, Englewood Cliffs, NJ.

[32] A. M. Uttley, “A theory on the mechanism of learning based on the computation of conditional probabilities”, Proceedings of the First International Congress on Cybernetics, Naumur, 1956, pp. 830-856.

[33] F. Rosenblatt, Principles of Neurodynamics, Science Editions, New York, 1962.

[34] Ian Goodfellow, Yoshua Bengio y Aaron Courville, Deep Learning (sitio web: http://www.deeplearningbook.org), MIT Press, 2016.

[35] Minsky, M.L. y Papert, S.A. (1969) Perceptrons. MIT Press, Cambridge.

[36] Bonifacio Martín del Brío y Alfredo Sanz Molina, Redes neuronales y sistemas borrosos, 2006, Ra-Ma S.A., ISBN-13 9788478977437.

[37] J. Aroztegui Vélez, E. García Buendía y J. M. Benito Escario, Introducción a la programación en Inteligencia Artificial, 2009, UCM, Antonio Benítez (editor).

[38] Stan Birchfield, Image Processing and Analysis, 2016, Cengage Learning, ISBN-13 9781285179520.

[39] M. Dorigo, *Optimization, Learning and Natural Algorithms*, 1992, tesis PhD, Politécnico de Milán, Italia.

[40] Hilbert, Martin & López, Priscila. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. Science (New York, N.Y.). 332. 60-5. 10.1126/science.1200970.

[41] Shun-ichi Amari, Forty Years of Perceptrons, 2001, RIKEN Brain Science Institute.

[42] S. N. Sivanandam, S. N. Deepa, Principles of Soft Computing, 2a ed., 2007, Wiley, ISBN-13 9788126510757.

[43] David E. Rumelhart; James L. McClelland, "A General Framework for Parallel Distributed Processing," in Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations , MIT Press, 1987, pp.45-76.

[44] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323, 533-536.

[45] Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation.

[46] Hee-Seung Na and Youngjin Park, "Symmetric neural networks and its examples," [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, Baltimore, MD, USA, 1992, pp. 413-418 vol.1, doi: 10.1109/IJCNN.1992.287176.

[47] Christos Volos y Viet-Thanh Pham, Mem-elements for Neuromorphic Circuits with Artificial Intelligence Applications, 2021, Academic Press, ISBN-13 9780128211847.

[48] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities (1982), Proceedings of the National Academy of Sciences of the United States of America, 79 8, 2554-8.

[49] Yorek, Nurettin & Ugulu, Ilker & Aydin, Halil. (2017). kohonen-self-organizing-maps-shyam-guthikonda.

[50] Kohonen, T. (1982) Analysis of a simple self-organizing process. Biological Cybernetics, 44, 135-140.

[51] Broomhead, D.S. and Lowe, D. (1988) Multi-Variable Functional Interpolation and Adaptive Networks. Complex Systems, 2, 327-355.

[52] Igor Aizenberg, Complex-Valued Neural Networks with Multi-Valued Neurons: 353 (Studies in Computational Intelligence), 2011, Springer.

[53] Aizenberg I.N., Aizenberg N.N. and Vandewalle J., "Multi-Valued and Universal Binary Neurons: Theory, Learning, Applications", 2000, Kluwer Academic Publishers, Boston/Dordrecht/London.

[54] Barto, A. G.; Sutton, R.S.; and Anderson, C.W., "Neuronlike adaptive elements that can solve difficult learning control-problems", 1983, *IEEE Transactions on Systems, Man and Cybernetics*, 362.

[55] Hubel D. H., Wieles T. N., "Receptive fields of single neurons in the cat's striate cortex", J Physiol, 1959, 148(3):574-91.

[56] Kunihiko Fukushima, Neocognitron, 2007, Scholarpedia, doi: 10.4249/ scholarpedia.1717.

[57] J. Weng, N. Ahuja and T. S. Huang, "Cresceptron: a self-organizing neural network which grows adaptively", [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, 1992, pp. 576-581 vol.1, doi: 10.1109/IJCNN.1992.287150.

[58] Weng, J; Ahuja, N; Huang, TS (1993). "Learning recognition and segmentation of 3-D objects from 2-D images". Proc. 4th International Conf. Computer Vision: 121–128. doi:10.1109/ICCV.1993.378228. ISBN 0-8186-3870-2. S2CID 8619176.

[59] Schmidhuber, Jürgen (2015). "Deep Learning". Scholarpedia. 10 (11): 1527–54. CiteSeerX 10.1.1.76.1541. doi:10.1162/neco.2006.18.7.1527. PMID 16764513. S2C ID 2309950.

[60] LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition". Proceedings of the IEEE 86 (11): 2278–2324.

[61] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput. 2006 Jul;18(7):1527-54. doi: 10.1162/neco.2006.18.7.1527. PMID: 16764513.

[62] Ciresan, Dan; Ueli Meier; Jonathan Masci; Luca M. Gambardella; Jurgen Schmidhuber (2011). "Flexible, High Performance Convolutional Neural Networks for Image Classification". Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two 2: 1237–1242.

[63] J.L. Ojeda, J.M. Icardo, Neuroanatomía Humana, «Aspectos funcionales y clínicos», 2004, Elsevier Masson, ISBN-13 9788445814086.

[64] Ryan Splittgerber, Neuroanatomía Clínica, 8a ed., 2019, Lippincott Williams & Wilkins, Wolters Kluwer, ISBN-13 9788417602109.

[65] Santiago Ramón y Cajal, "Histología del sistema nervioso humano y de las vértebras", 1911, Maloine, Paris.

[66] N. M. Amosov, T. N. Baidyk, A. D. Goltsev, A. M. Kasatkin, L. M. Kastkina, E. M. Kussul, D. A. Rachkovski, *Neurocomputadoras y robots inteligentes*, Naukova Dumka, Kiev, 1991.

[67] Jacek M. Zurada, "Introduction to Artificial Neural Systems", 1994, Jaico Publishing House, ISBN-13 9788172242664.

[68] Elaine Rich y Kevin Knight, Artificial Intelligence, 2a ed., 1991, McGraw-Hill Education, ISBN-13 9780070522633.

[69] Teuvo Kohonen, An introduction to neural computing, Neural Networks, Volume 1, Issue 1, 1988, Pages 3-16, ISSN 0893-6080, https://doi.org/10.1016/0893-6080(88)90020-2.

[70] R. Hecht-Nielsen, "Neurocomputing: picking the human brain," in IEEE Spectrum, vol. 25, no. 3, pp. 36-41, March 1988, doi: 10.1109/6.4520.

[71] O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.

[72] Albawi, Saad & Abed Mohammed, Tareq & ALZAWI, Saad. (2017). Understanding of a Convolutional Neural Network. 10.1109/ICEngTechnol.2017.8308186.

[73] LeCun, Y., Bengio, Y. and Hinton, G. (2015) Deep Learning. Nature, 521, 436-444. http://dx.doi.org/10.1038/nature14539.

[74] Sainath, T.N., Kingsbury, B., Mohamed, A., Dahl, G.E., Saon, G., Soltau, H., Beran, T., Aravkin, A.Y., & Ramabhadran, B. (2013). Improvements to Deep Convolutional Neural Networks for LVCSR. 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, 315-320.

[75] Hecht-Nielsen, R. (1989). Theory of the backpropagation neural network. International 1989 Joint Conference on Neural Networks, 593-605 vol.1.

[76] O. Abdel-hamid, L. Deng, and D. Yu, "Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition," no. August, pp. 3366–3370, 2013.

[77] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, C. V Jan, J. Krause, and S. Ma, "ImageNet Large Scale Visual Recognition Challenge".

[78] R. E. Turner, "Lecture 14 : Convolutional neural networks for computer vision", 2014, Departamento de Ingeniería (CBLL), Universidad de Cambridge.

[79] J. Wu, "Introduction to Convolutional Neural Networks," pp. 1–28, 2016.

[80] Wei Xiong , Bo Du, Lefei Zhang, Ruimin Hu, Dacheng Tao "Regularizing Deep Convolutional Neural Networks with a Structured Decorrelation Constraint " IEEE 16th International Conference on Data Mining (ICDM) , pp. 3366–3370, 2016.

[81] I. Kokkinos, E. C. Paris, and G. Group, "Introduction to Deep Learning Convolutional Networks, Dropout, Maxout 1," pp. 1–70.

[82] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097- 1105).

[83] Mark Skilton y Felix Hovsepian, "The 4th Industrial Revolution «Responding to the Impact of Artificial Intelligence on Business»", 2018, Palgrave Macmillan, ISBN-13 ISBN 978-3-319-62479-2, https://doi.org/10.1007/978-3-319-62479-2.

[84] Su, Y., Jiang, X. Prediction of tide level based on variable weight combination of LightGBM and CNN-BiGRU model. Sci Rep 13, 9 (2023). https://doi.org/10.1038/s41598-022-26213-y.

[85] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradientbased learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[86] Hinton, G.E. (2002). Training products of experts by minimizing contrastive divergence. Neural Computation, 14 (8): 1711–1800.

[87] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger, "A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection", IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[88] Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", Preprint, submitted March 2, 2015. https://arxiv.org/abs/1502.03167.

[89] Dai, Z., & Heckel, R. (2019). Channel Normalization in Convolutional Neural Network avoids Vanishing Gradients. ArXiv, abs/1907.09539.

[90] Léon Bottou, "Online Algorithms and Stochastic Approximations", *Online Learning and Neural Networks*, 1998, Cambridge University Press, ISBN 978-0-521-65263-6.

[91] Freund, Y. y Schapire, R.E., "A desicion-theoretic generalization of on-line learning and an application to boosting," in Computational learning theory, 1995, 23-37.

[92] Chawla, Nitesh & Bowyer, Kevin & Hall, Lawrence & Kegelmeyer, W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. (JAIR). 16. 321-357. 10.1613/jair.953.

[93] Blackard, J. A. y D. J. Dean. "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic

variables". Computers and Electronics in Agriculture Vol. 24, Issue 3, 1999, pp. 131–151.

[94] Blackard, Jock A. 1998. "Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types." Ph.D. dissertation. Department of Forest Sciences. Colorado State University. Fort Collins, Colorado. 165 páginas.

[95] Rowley, H., Baluja, S., & Kanade, T. (1996). Neural network-based face detection. Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 203-208.

[96] Neil Wilkins (editor), Inteligencia artificial «Lo que usted necesita saber sobre el aprendizaje automático, robótica, aprendizaje profundo, Internet de las cosas, redes neuronales, y nuestro futuro», 2019.

[97] Alom, Md. Zahangir & Taha, Tarek & Yakopcic, Christopher & Westberg, Stefan & Hasan, Mahmudul & Esesn, Brian & Awwal, Abdul & Asari, Vijayan. (2018). The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches.

[98] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 6 (June 2017), 84–90. https://doi.org/10.1145/3065386.

[99] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[100] Duran-Lopez, Lourdes & Dominguez-Morales, Juan Pedro & Conde-Martin, Antonio & Vicente Díaz, Saturnino & Linares-Barranco, Alejandro. (2020). PROMETEO: A CNN-based computer-aided diagnosis system for WSI prostate cancer detection. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3008868.

[101] Hoeser, Thorsten & Kuenzer, Claudia. (2020). Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends. Remote Sensing. 12. 10.3390/rs12101667.

[102] Sun, Lian & Yan, Hexiang & Xin Kunlun, Kun-lun & Tao, Tao. (2019). Contamination source identification in water distribution networks using convolutional neural network. Environmental Science and Pollution Research. 26. 10.1007/s11356-019-06755-x.

[103] Philip D. Wasserman, *Neural Computing: Theory and Practice*, 1989, Van Nostrand Reinhold, Nueva York, ISBN-13 9780442207434.

[104] Yann LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard y L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", *Neural Computation*, vol. 1, no. 4, pp. 541-551, Winter 1989.

[105] M.S. Hoque, M.C. Fairhurst, A moving window classifier for off-line character recognition, in: Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, 2000, pp. 595–600.

[106] Ernst Kussul, Tatiana Baidyk, *LIRA neural classifier for handwritten digit recognition and visual controlled microassembly*, Neurocomputing, Volume 69, Issues 16–18,2006, Pages 2227-2235, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2005.07.009.

[107] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539.

[108] G. P. Martinsanz, M. S. Peñas, Inteligencia Artificial e Ingeniería del Conocimiento, 2005, Ra-Ma, ISBN-13 8478976760.

[109] Sin-Yeung Cho and T. W. S. Chow, "Shape and surface measurement technology by an improved shape-from-shading neural algorithm," in IEEE Transactions on Industrial Electronics, vol. 47, no. 1, pp. 225-230, Feb. 2000, doi: 10.1109/41.824148.

[110] Neubauer, C. (1991). Fast detection and classification of defects on treated metal surfaces using a backpropagation neural network. [Proceedings] 1991 IEEE International Joint Conference on Neural Networks, 1148-1153 vol. 2.

[111] Rafael C. González, Richard E. Woods, Digital Image Processing, 2ª ed., 2001, Prentice Hall, Pearson Education International, ISBN 0201180758.

[112] Shannon, C.E. (1948). A mathematical theory of communication. Bell Syst. Tech. J., 27, 623-656.

[113] Moribata, Y., Kurata, Y., Nishio, M. et al. Automatic segmentation of bladder cancer on MRI using a convolutional neural network and reproducibility of radiomics features: a two-center study. Sci Rep 13, 628 (2023). https://doi.org/10.1038/s41598-023-27883-y.

[114] Antonio Gulli, Amita Kapoor y Sujit Pal, *Deep Learning with TensorFlow 2 and Keras: Regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API*, 2a ed., 2019, Packt Publishing, ISBN-13 9781838823412.

[115] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv, abs/1505.04597.

[116] Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T. et al. Selective Search for Object Recognition. Int J Comput Vis 104, 154–171 (2013). https://doi.org/10.1007/s11263-013-0620-5.

[117] R. Girshick, J. Donahue, T. Darrell and J. Malik 'Rich feature hierarchies for accurate object detection and semantic segmentation'. CVPR 2014.

[118] S. Ren, K. He, R. Girshick and J. Sun "Faster R-CNN: Towards Real-Time Object Detecetion with Region Proposal Networks". NeurIPS 2015.

[119] Z. Zhao, P. Zheng, S. Xu and X. Wu 'Object Detection with Deep Learning: A Review'. IEEE Transactions on Neural Networks and Learning Systems PP(99): 1-21, January 2019.

[120] Stephon Alexander, El jazz de la física «El vínculo secreto entre la música y la estructura del universo», 3ª ed., 2017, Tusquets Editores, S. A., ISBN-13 9788490663684.

[121] S. Karner. Laws of Thought. Encyclopedia of Philosophy, 4:414–417, 196.

[122] Alice Zheng y Amanda Casari, *Feature Engineering for Machine Learning «Principles and Techniques for Data Scientists»*, O'Reilly, 2018, ISBN-13 9781491953242.

[123] Hornik, K., Stinchcombe, M., White, H.: Multilayer Feedforward Networks are Universal Approximators. Neural Networks 2, pp. 359–366, 1989.

[124] Carl B. Boyer, Historia de la matemática, vol. 18 de Ciencia y Tecnología, Alianza Editorial, 1999, ISBN-13 9788420681863.

[125] Aurélien Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2019, O'Reilly Media, ISBN-13 9781492032649.