APRENDIZAJE PROFUNDO: UNA NUEVA VÍA PARA CONVERTIR EL DATO EN CONOCIMIENTO

JOSE ANTONIO LAGARES

NORBERTO DÍAZ-DÍAZ

CARLOS D. BARRANCO GONZALEZ

Universidad Pablo de Olavide

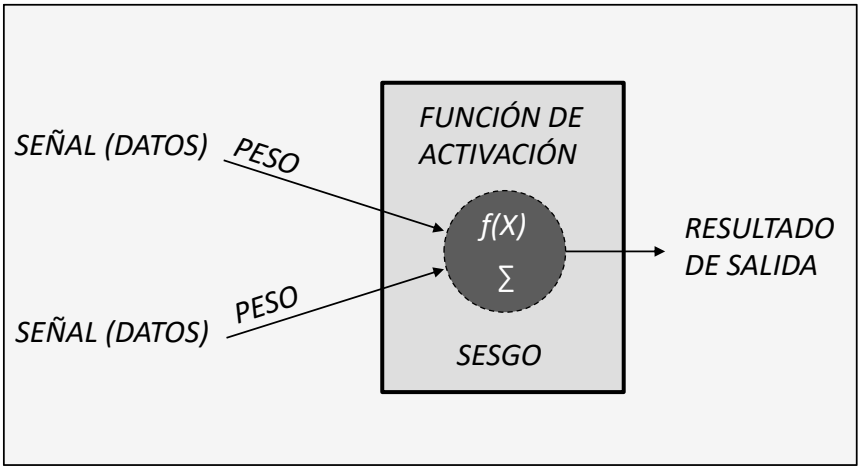
Aproximarse al aprendizaje profundo (*Deep Learning - DL*), como parte de la Informática y un subcampo del aprendizaje automatizado o *Machine Learning*, implica entender algoritmos que tienen que abordan problemas que requieren una capacidad similar al razonamiento humano. Es posible reconocer a las técnicas de aprendizaje profundo como aquellos enfoques que emulan la perspectiva de aprendizaje lógico-conductual que los seres humanos

usan para obtener conocimiento, también llamado enfoque simbólico, o bien a aquellas técnicas que imitan el comportamiento de la naturaleza mediante programación evolutiva, simulación de enjambres o algoritmos genéticos, también conocidas como enfoques sub-simbólicos (Calegari, 2020).

El DL está tomando mayor relevancia en el campo empresarial a través de la potenciación y uso de la Inteligencia Artificial (IA), gracias al empleo de la Computación en la Nube (*Cloud computing - CC*) como herramienta para acceder a ingentes capacidades de cálculo. En diferentes escenarios, las técnicas de *Machine Learning* (ML) clásicas no son suficientemente precisas para abordar problemas que requieren modelos de altísima complejidad. Estas técnicas no encuentran la mejor solución ante un conjunto de datos masivo, como los presentes en problemas dentro del paradigma Big Data. Los enfoques de ML tradicionales tienen una capacidad de aprendizaje finita en relación al número de elementos con que se trabaja (Mitchell, 2015). En cambio, las técnicas de DL ofrecen una posibilidad de aprendizaje casi infinita ya que son capaces de seguir infiriendo conocimiento de forma significativa mientras haya datos disponibles y maximizan su desempeño en función de dicha cantidad (Ng, 2016). Es por ello que estas técnicas son las más usadas cuando el escenario cuenta con fuentes de datos pertenecientes a casos relacionados con enfoques Big Data (C.-W. Tsai, 2015).

En los últimos años se han desarrollado avances de una forma muy significativa dentro del campo del DL, llegando a presentarse enfoques y algoritmos que igualan o superan las capacidades humanas (I. Goodfellow, 2017). A nivel empresarial, no se ha tardado en trasladar estas mejoras a las aplicaciones tradicionales para mejorar su funcionamiento y ampliar sus prestaciones dando lugar a nuevas características presentando nuevas oportunidades y horizontes a productos software ya existentes (Mathias Kraus, 2020).

GRÁFICA 1
ESTRUCTURA DE UNA NEURONA



Fuente: Elaboración propia

En otra instancia, es posible encontrar a empresas punteras como Apple, Google, Microsoft o Amazon apostando cada vez más por implementar *DL* en sus servicios y productos, ofreciendo una gama de asistentes virtuales y/o cognitivos, presentes incluso en dispositivos móviles o domésticos. Cabe señalar que Google incluso ha desarrollado algoritmos de DL, como el llamado Google *DeepMind* que venció en el popular juego *AlphaGo* al campeón mundial Lee Sedolc (1).

A nivel empresarial, es posible encontrar los algoritmos de DL en múltiples áreas para mejorar la automatización de tareas en la minería de texto (*Text Mining*) tales como detectar *spam* en correos electrónicos o su clasificación automática según una categoría, mejorar procesos en la industria 4.0 y revolucionar el sector automovilístico controlando y manejando vehículos de forma autónoma, impactar en la bolsa pronosticando precios de acciones y tendencia, ayudar en el reconocimiento de imágenes detectando entidades, creando avances de automatización en robótica, mejorando procesos en bioinformática para identificar patrones o facilitar la extracción de conocimiento en ensayos, analizar diagnósticos clínicos para la detección temprana de enfermedades, estar presentes en asistentes cognitivos, dar un nuevo sentido al procesamiento del lenguaje natural o ser una nueva pieza clave dentro de la ciberseguridad entre otras muchas aplicaciones.

A pesar de los avances realizados, el DL no está exento de polémica debido al sesgo en que puede incurrir. A lo largo de los últimos años se han detectado casos en los cuales sistemas inteligentes, basados en DL, han planteado cuestiones éticas relacionadas con prejuicios. Es indudable que las empresas optan por este tipo de soluciones buscando exclusivamente una mejora y automatización en los procesos de negocio. No obstante, se han detectado consecuencias imprevistas a la hora de usar estas técnicas en, por ejemplo, procesos de selección y contratación de personal (Dastin, 2017), en los que se sesgó por género o raza a candidatos a puestos de trabajo. También el sesgo está presente en algoritmos de reconocimiento facial (2), vigilancia (3), técnicas de análisis de redes sociales o asociación de imágenes al género (Jieyu Zhao, 2017). Debido a esto, se ha llegado a plantear de con una orientación ética la cuestión de qué datos deben ser usados en este tipo de sistemas.

Por su estructura y la manera de procesar la información, es necesario entender que las técnicas de DL son capaces de llegar a soluciones de forma diferente a los enfoques de programación tradicionales. Se comienza presentando, la diferencia entre ambos enfoques, para a continuación explicar las distintas partes que forman este tipo de algoritmos y se reseñarán los tipos de algoritmos y sus utilidades. Además, se abordarán los distintos modelos más prolíferos de DL para explicar su funcionamiento y resaltarán las bondades en el uso de enfoques *DL*. Por último, se recalcará las distintas herramientas existentes en el mercado para crear estos enfoques y su presencia dentro del CC.

PROGRAMACIÓN REGULAR FRENTE A APRENDIZAJE PROFUNDO

En la programación regular se desarrollan algoritmos para convertir entradas en resultados. Para realizar este proceso, se usan reglas lógicas y elementos matemáticos necesarios. En cambio en el DL se parte de una lista de valores de entrada y resultados esperados a cada uno de los valores de entrada. Estos valores servirán como entrenamiento de la red neuronal ya que, por sí misma, no tiene conocimiento del conjunto de reglas lógicas y matemáticas que debe aplicar para lograr la conversión entre los da-

tos de entrada y los datos de salida. Esta principal diferencia entre la programación regular y DL automático.

Comúnmente, una red neuronal se entrena con una gran capacidad de datos, como los que encontramos en escenarios Big Data. Por cada uno de esos datos de entrada, la red ofrecerá una predicción. Este proceso se repetirá y, guiada por parámetros de configuración, la red neuronal iterará, cientos o incluso miles de veces, para llegar a converger y encontrar una solución. En cada paso, las neuronas presentes en la red usan combinaciones entre los pesos de las conexiones, sesgos y funciones de activación para poder hacer predicciones. Al finalizar cada iteración, la propia red evaluará el resultado de su predicción para ir, poco a poco, ajustando de forma automática sus sesgos y pesos. Según la complejidad de la red y el número de neuronas o capas, así como la capacidad de computación disponible, este proceso puede llevar horas, minutos o incluso días.

PARTES DE UN ALGORITMO DE APRENDIZAJE PROFUNDO

Los algoritmos de DL se caracterizan por extraer conocimiento de los datos. En el centro de estos algoritmos existen unas estructuras llamadas redes neuronales o redes neuronales artificiales (*Artificial Neuronal Network, ANN*) (FV, 2016). Las redes neuronales artificiales emulan la estructuras biológicas que encontramos en el cerebro humano. Los algoritmos usan elementos llamados neuronas, que procesan información para producir resultados.

La estructura de las redes neuronales artificiales sigue un diseño por capas. Cada una de estas capas está, a su vez, formada por un conjunto de neuronas que cuentan con varias vías de entrada (*inputs*) y salidas (*outputs*) hacia otras neuronas. Como mínimo, siempre habrá una capa de entrada y una de salida. Además de las anteriores, pueden existir más capas intermedias denominadas capas ocultas. Las neuronas están conectadas entre sí y transfieren información a través de dichas conexiones. Estos elementos son de vital importancia para la red ya que internamente la red neuronal otorga un peso a cada conexión para medir la importancia de la misma. Cuando una red neuronal es instanciada por primera vez, este peso tiene un valor aleatorio. Por otro lado, cada neurona, a excepción de la capa de entrada, tiene un valor numérico interno que representa el sesgo que la red comete al tratar con los datos. Al recibir los mismos, una neurona aplicará una función matemática para operar con los datos llamada *función de activación*. Posteriormente la neurona aplicará el sesgo interno y valorará la importancia de la conexión. A grandes rasgos, es posible resumir el trabajo de una neurona como la suma de tareas para la recepción de los datos, aplicación de la función de activación, integración de la importancia de la conexión establecida, suma del sesgo y transferencia del resultado numérico obtenido a la siguiente neurona o capa. En escenarios complejos, las redes neuronales pueden llegar a presentar una estructura formada con cientos de neuronas, miles de capas ocultas y millones de conexiones capaces de dar un resultado teniendo en cuenta decenas o cientos de variables.

Una de las características más significativas de este tipo de enfoques es su capacidad de aprendizaje. Para ello, cuentan con una herramienta interna llamada retro-propagación o "*backpropagation*" que permite realizar ajustes a los pesos de las entradas y sesgos de las neuronas, con el objeto de reducir los fallos cometidos. El proceso de retro-propagación es similar al razonamiento humano a la hora de aprender y se realiza dentro del entrenamiento de la red neuronal. Una vez que la red sugiere un valor de salida, contando con una determinada configuración de errores y pesos en sus neuronas, comprueba su grado de acierto o error verificando su resultado con la salida esperada. Cuando la red neuronal encuentra que su predicción ha resultado errónea, extiende la noticia hacia atrás, es decir, desde la capa de salida hasta el resto de las capas para que cambien los sesgos y pesos de las neuronas en una iteración posterior. El proceso de retro propagación es necesario para ajustar los sesgos y los pesos establecidos de forma aleatoria, permitiendo a la red neuronal aprender de sus fallos e ir corrigiendo en cada iteración. No obstante, es posible que encontrar la mejor configuración de pesos y sesgos dentro de las neuronas tome gran cantidad de tiempo pudiendo, inclusive, no converger en una solución óptima. Este coste computacional es el que limitó, en su día, el uso de estas aproximaciones, pudiendo actualmente abordarse gracias a la capacidad de computación actual y a las técnicas de distribución de tareas, enmarcado en el *cloud computing*.

TIPOS DE APRENDIZAJE PROFUNDO Y UTILIDADES

La literatura relacionada con las técnicas de DL ha distinguido diferentes subcampos relacionados directamente a una jerarquía en relación a los tipos de algoritmos existentes. Si bien el primer paso para aplicar DL es el uso de redes neuronales artificiales, existen multitud de topologías para identificar a los tipos de redes.

Aunque existe una gran cantidad de tipos de algoritmos, aquellos más relevantes en la literatura son (CM, 2006): redes con aprendizaje supervisado, no supervisado, semi-supervisado o aprendizaje por refuerzo.

A. Redes con aprendizaje supervisado

   Pertenecen a este tipo aquellas redes que en el momento de su entrenamiento cuentan con los datos etiquetados. Dentro de los datos de entrada que consumen existe un valor a modo de etiqueta para identificarlos y poder asociarlos entre sí de forma categórica. Esta etiqueta

recibe el nombre de *clase* y dentro de conjuntos de datos usados en aprendizaje supervisado pueden existir un numero infinito de las mismas. Cada clase representa un conjunto de elementos con unas características determinadas y propias. Cuando la red neuronal es entrenada, internamente extraerá todo el conocimiento necesario para identificar la etiqueta que consume frente a los datos que evalúa. Así, el algoritmo iterará de forma gradual para entender cómo localizar una clase en concreto y relacionarla a un conjunto de variables. Posteriormente al entrenamiento, será capaz de asociar una futura entrada de información a una clase que ha sido analizada según los valores y características introducidas.

Las dos operaciones fundamentales que se pueden realizar de forma agregada con datos etiquetados son, por un lado, clasificar un dato frente al conjunto en general, y, por otro lado, realizar operaciones de estimación o regresión de cantidades de la información extraída.

B. Redes con aprendizaje no supervisado y semi supervisado

De forma contraria, pueden existir conjuntos de datos los cuales no estén etiquetados con el término clase. En este escenario las redes neuronales aprenden las relaciones internas presentes entre los atributos del conjunto de datos e intentan descubrir afinidades ocultas entre las cualidades de los datos o incluso sugerir agrupaciones desconocidas de los datos que difícilmente pueden ser consideradas por una persona.

No siempre es posible tener un conjunto de datos donde el atributo clase está presente en todos nuestros registros. Existen casos en los que la cantidad de registros categorizados son limitados frente a la cantidad de datos globales. Este tipo de escenarios es abordado por redes neuronales de aprendizaje semi supervisado.

C. Aprendizaje profundo por refuerzo

Las redes creadas bajo este paradigma se centran en el aprendizaje por ensayo y error. Así, estos algoritmos se centran en maximizar la recompensa seleccionando las acciones apropiadas dependiendo de los datos de entrada que tengan para llegar a tal fin. Es un enfoque usado en aquellos escenarios donde existe una interacción con elementos para la extracción de información basados en estados.

Por su forma de abordar problemas, es común comparar dicha técnica con aquellas pertenecientes tanto al aprendizaje supervisado como no supervisado. Es posible recopilar las tareas fundamentales del aprendizaje supervisado en la clasificación o identificación de elementos. Estas técnicas están basadas en el tratamiento de conjuntos de datos que previamente han sido etiquetados. En otras palabras, el objetivo es lograr aprender caminos para categorizar la información. No obstante, ante datos inexplorados, que no han sido categorizados previamente, los modelos creados en base a aprendizaje por refuerzo son capaces de aprender o categorizar en función de su propia experiencia. Por otro lado, la tarea fundamental del aprendizaje no supervisado es encontrar estructuras dentro de una colección de datos quedando descartado el enfoque de maximizar la recompensa u objetivo a conseguir.

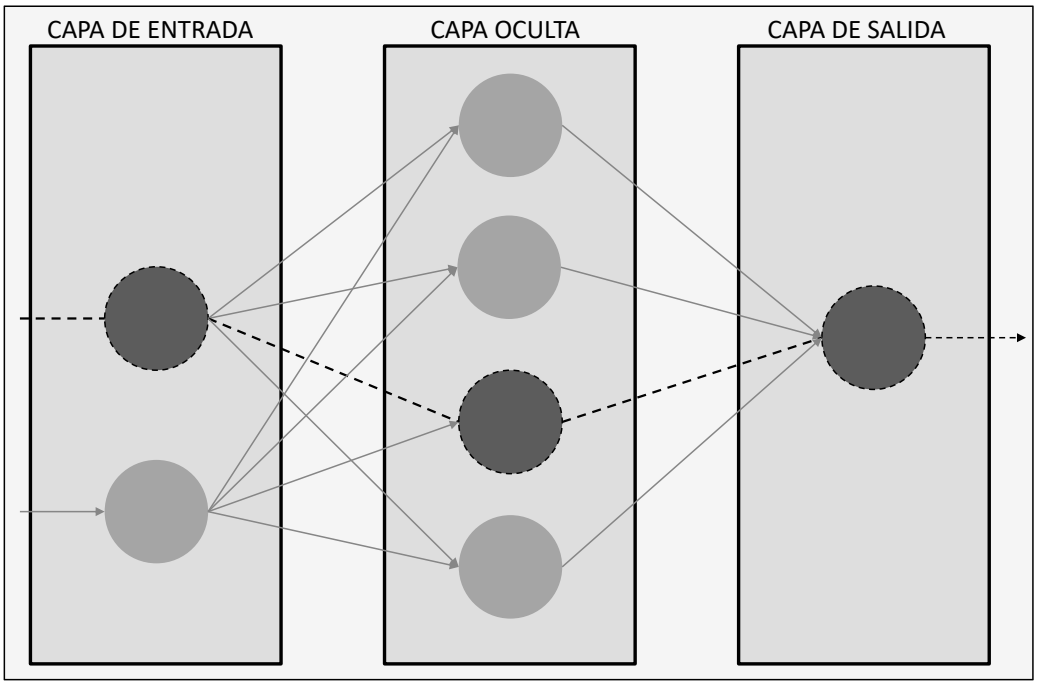
A continuación, serán objeto de análisis las redes neuronales profundas (*Deep Neuronal Network, DNN*), las redes neuronales convolucionales (*Convolutional Neural Network, CNN*), redes neuronales recurrentes (*Recurrent Neuronal Network, RNN*) y un subtipo de las mismas llamado las redes neuronales de gran memoria a corto plazo (*Long Short Term Memory, LSTM*) como las topologías de redes neuronales más prolíferas.

Si bien, dependiendo del tipo de problema a resolver es posible optar por un modelo específico siguiendo una topología concreta, existe a nivel programático la posibilidad de empezar con modelos pre-entrenados. Un modelo pre-entrenado es un modelo listo para ser usado, ya que previamente ha sido entrenado en un tipo de problema o área de conocimiento en concreto. El uso de este tipo de modelos, en vez de optar por una creación de un modelo desde cero, garantiza llegar a tener un algoritmo funcional en el caso de tener una falta de datos de entrenamiento. Además, evitar tener que iniciar la red desde cero evitando la espera y el ajuste de la misma tras su proceso. Existen multitud de modelos pre-entrenados para múltiples tipos de problemas que es posible encontrar dentro de las herramientas comunes de programación de redes neuronales profundas.

Redes neuronales profundas con capas completamente conectadas (DNN, Deep Neural Networks)

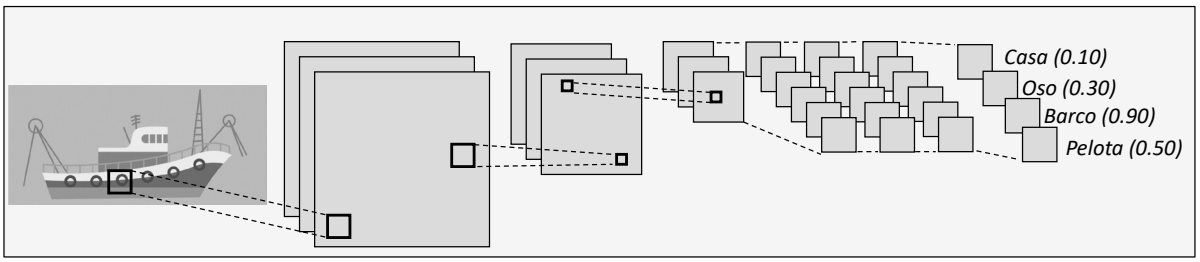
Es una de las topologías más usadas por su potencia y facilidad en su configuración. Este tipo de redes presentan sus neuronas agrupadas en múltiples capas, es decir, poseen un conjunto de neuronas establecidas por niveles. Todas las neuronas de una capa determinada están conectadas con las neuronas de la capa siguiente y a su vez están conectadas con las neuronas de la capa anterior. La información fluirá a través de las neuronas desde la capa de entrada hasta la capa de salida. Este tipo de redes son aptas para abordar un rango amplio de problemas que cuenten con datos estructurados, ya que, debido a su arquitectura, potencian el encuentro de una senda de extracción e identificación de conocimiento entre las neuronas permitiendo identificar patrones en los datos.

GRÁFICA 2
ESTRUCTURA DE UNA RED NEURONAL COMPLETA



Fuente: Elaboración propia

GRÁFICA 3
PROCESO DE CONVOLUCIÓN



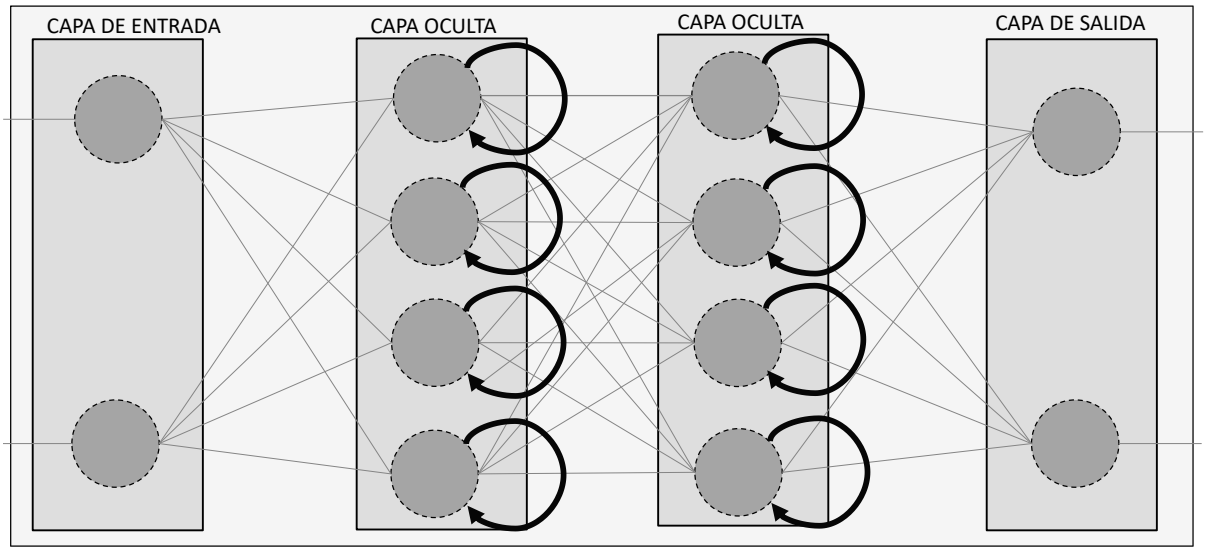
Fuente: Elaboración propia

REDES NEURONALES CONVOLUCIONALES (CNN, CONVOLUTIONAL NEURAL NETWORKS)

Se conoce el término *convolución* como la aplicación de una función matemática sobre otra función. Este concepto es usado dentro de las redes neuronales para identificar patrones en los datos, concretamente, en las redes neuronales convolucionales están usualmente destinadas al tratamiento de imágenes y visión computerizada. Esto es posible gracias a la combinación de múltiples capas junto con este concepto matemático. Si bien se ha presentado la idea de agrupación de neuronas usando capas, en esta topología cada capa tiene el objetivo de identificar formas u objetos dentro de las imágenes que son procesadas usando la convolución realizada por la capa anterior. En comparación con las redes neuronales profundas, las redes convolucionales presentan una menor conexión entre sus neuronas, centrándose en que cada capa aplique su convolución y entregue su procesamiento a la siguiente para así extraer conocimiento.

Para que las imágenes sean tratadas, primero son convertidas en matrices bidimensionales de bytes, que representan sus píxeles. Además, existe una tercera dimensión, el canal usado en la imagen. Es posible encontrar imágenes con 2 canales (blancos y negros) y 3 canales (imágenes a color sobre RGB). Por cada canal existente habrá una matriz numérica que represente la información de la propia imagen. Por tanto, las imágenes serán transformadas en un conjunto de matrices que estarán

GRÁFICA 4
ESTRUCTURA DE RED NEURONAL RECURRENTE



Fuente: Elaboración propia

sobrepuestas unas sobre otras según el número de canales existentes. Posteriormente, para aplicar la convolución, la imagen se divide en áreas y cada una es analizada desde un punto de vista matricial. El objetivo es reducir la dimensión de las matrices aplicándoles operaciones matemáticas que resuma la información contenida en esa área o conjunto de matrices. Este filtro recorre de forma convolucional todas las matrices representadas en la imagen dando como resultado final una nueva matriz que representa la imagen ya tratada. El uso de esta técnica permitirá reducir la cantidad de pixeles mejorando el procesamiento e identificando información relevante dentro de una región de la imagen procesada. Los valores de la matriz filtro junto con el avance de la misma son áreas de ajustes dentro de la red neuronal.

Este tipo de redes son ideales para, partiendo de imágenes como entrada, asignar importancia a objetos para poder identificarlos. La agrupación de capas convolucionales nos ofrece una capacidad para detectar cambios en imágenes, objetos o aplicar filtros para corrección de desenfoque.

REDES NEURONALES RECURRENTES (RNN, RECURRENT NEURAL NETWORK)

Si bien la transferencia de información dentro de una red neuronal se identifica desde la capa de entrada hasta la capa de salida de la misma, las redes neuronales recurrentes presentan un enfoque distinto ya que realizan una propagación de la información inversa. De esta forma, la red dirige a capas anteriores de la misma la información procesada de otra de sus capas. Para tratar la información de forma recurrente, la red permite que el conocimiento generado de las salidas se considere como una nueva entrada en otra capa tratando de extraer patrones de largas secuencias de datos enfatizando en la temporalidad de los mismos. Siguiendo esta misma idea, una capa de neuronas recurrentes se puede concebir de tal manera que cada neurona recibe dos entradas, la entrada correspondiente de la capa anterior y la salida del instante anterior de la misma capa.

Esta topología de red permite detectar eventos o anomalías distribuyendo los datos de entrada a nuevas capas sin que su importancia sea disminuida entre la cantidad de conexiones existentes dentro de la misma. A mayor temporalidad de datos a usar, mayor cantidad de capas deberá tener la red.

Los enfoques presentados en secciones anteriores toman como entrada imágenes o matrices de un tamaño determinado y exponen como salida una matriz, una probabilidad o identificación de una clase determinada. En cambio, el enfoque recurrente permite realizar operaciones sobre un conjunto de datos de entrada teniendo en cuenta la temporalidad, siendo un enfoque ideal para el tratamiento de series temporales o detección de anomalías.

La extracción de conocimiento en estas redes es concebida en la realización de nuevas iteraciones sobre las ya realizadas anteriormente. Una red neuronal recurrente puede contar con miles de parámetros de entrada y/o capas, llegando a consumir una importante cantidad de recursos computacionales. Debido a este factor, la cantidad de memoria reservada para estos modelos puede llegar a ser ingente, siendo las redes de gran memoria de corto plazo una posible alternavita.

REDES DE GRAN MEMORIA DE CORTO PLAZO (LSTM, LONG SHORT-TERM MEMORY)

Este enfoque es considerado como una especialización de las redes neuronales recurrentes. Si bien la gestión de la memoria puede suponer un problema en las redes neuronales recurrentes, se presenta un modelo que mejora la gestión de la misma otorgando la capacidad de recordar la información de entrada a sus capas durante más tiempo y pudiendo borrar aquella información no necesaria o de la que ya ha extraído el conocimiento necesario.

Las redes de gran memoria a corto plazo tratan la información como celdas que cuentan con dos estados: libres o bloqueadas. La información se encontrará bloqueada en neuronas cuando la red considere que dicha información es útil. Por el contrario, si un conjunto de neuronas dispone de estados libres, la información almacenada podrá ser sobrescrita ya que la red neuronal ha considerado que dicha información ya no es útil.

La red establecerá la importancia de la información e irá actuando en consecuencia para ir aprendiendo, funcionando similar a los modelos recurrentes, pero con una gestión de memoria optimizada para ser objeto de uso en grandes volúmenes de datos orientados a series temporales.

ENFOQUES Y CARACTERÍSTICAS EN EL USO DE TÉCNICAS DE APRENDIZAJE PROFUNDO

Por su comportamiento podemos decir que los algoritmos de DL tienen las siguientes características:

- Adaptativos: Los algoritmos perciben el cambio de información y son muy sensibles al mismo a la hora de inferir conocimiento. Una vez que las redes neuronales se crean, deben entrenarse usando un subconjunto de datos. Este entrenamiento no se realiza de forma singular y se completa por fases o iteraciones. En cada iteración, los algoritmos de DL son capaces de evolucionar y así ver más allá de los datos, analizando patrones y extrayendo el conocimiento oculto en los mismos.
- Escalables: Es posible que un conjunto de dichos algoritmos pueda formar una parte software independiente. Llegados a este punto, estos algoritmos pueden ser entendidos como un servicio que ofrece su capacidad predictiva o inductora para ser consumida por otros sistemas informáticos. En sí, este concepto se conoce como *SaaS* (*Sofware as a Service*). Hoy en día, el uso de algoritmos de DL que estén respaldados por una amplia escalabilidad, facilita el éxito empresarial en su integración y su uso en casos Big Data junto sus ya ampliamente conocidas características de capacidad de tratamiento de volumen, velocidad y variedad.
- Iterativos: Los algoritmos intentan inferir o concluir en una solución a un problema dado. Para llegar hasta ese punto es necesario pasar una etapa de entrenamiento e hiperparametrización en los modelos. La robustez que ofrecen estas técnicas también reside en que son enfoques flexibles y soportan trabajar sobre variaciones de un problema determinado una vez que son entrenados. En sí, una red neuronal no requiere que sea preparada para todos los escenarios posibles en los que será usada, aunque la calidad de los resultados ofrecidos por la red frente a casos desconocidos dependerá en un alto porcentaje de la fase de entrenamiento realizada y la calidad de los datos de entrada que se usen.
- Contextuales: Una vez que usemos algoritmos de DL es necesario entender el contexto de los problemas que intentan resolver, los parámetros que usan y los resultados que se ofrecen. El conjunto de soluciones propuestas o inferidas por una red neuronal tiene que ser adaptado a un caso concreto. De forma análoga podemos confirmar que las soluciones a un problema determinado pueden cambiar a lo largo del tiempo dependiendo del escenario en el que nos encontremos. Por ejemplo, si una red neuronal es usada para predecir el tiempo meteorológico o predicciones de bolsa, estará sujeta a cientos o miles de parámetros de entradas que estén en continua evolución. Asimismo, las redes neuronales pueden llegar a inferir conocimiento usando datos estructurados y no estructurados.

HERRAMIENTAS PARA LA CREACIÓN DE ALGORITMOS DE APRENDIZAJE PROFUNDO

En el presente capítulo se introducirán de manera general las herramientas disponibles en el mercado para la creación de redes neuronales bajo dos paradigmas. Por un lado, en la subsección A se abordarán las herramientas *frameworks* para la creación manual de enfoques profundos y en la subsección B se detallarán los enfoques de DL ya desarrollados por los distintos proveedores *cloud* como elementos software listos para ser usados (*SaaS*).

A. Frameworks de desarrollo

A la hora de desarrollar un modelo de DL se tienen en cuenta múltiples detalles tales como la topología de la red a usar, el número de neuronas en las capas de entrada o salida, la cantidad de capas ocultas, la consideración de qué funciones de activación se emplean, etc. Aunque es posible codificar una red neuronal partiendo desde un inicio, determinar los anteriores factores hace que emprender la creación de un modelo se convierta en una tarea ardua. Sin embargo, existen elementos software llamados *frameworks* que disponen de estructuras previas que sirven como plantilla para permitir integrar de forma útil los elementos, ca-

pas y funciones que la forman. Por ello, es posible definir un *framework* en este campo como un conjunto de herramientas que, bajo un mismo espacio de trabajo, simplifican la creación y el desarrollo de modelos de DL. El uso de *frameworks* hace más fácil la tarea de programación, evitando la escritura de código repetitivo y ayudando al usuario final a desarrollar rápidamente. Dentro de sus múltiples funcionalidades, los *frameworks* tienen precargadas estructuras y topologías de redes, funciones de activación y elementos de utilidad de diversa índole que simplificarán la codificación, suponiendo un ahorro de tiempo en el desarrollo (Liermann, 2021).

Si bien los *frameworks* se han presentado como herramientas para ayudar a gestionar la faceta software en la creación de redes, también es posible configurar y aprovechar al máximo el uso de recursos hardware de un equipo o *clúster*. Dentro de la creación de redes neuronales, son las etapas de entrenamiento y evaluación del modelo aquellas que consumen una elevada cantidad de recursos computacionales. Los *frameworks* permiten adaptarse no solo a los recursos existentes en una máquina, sino también encontrar la mejor configuración posible cuando existe la posibilidad de hacer uso de paralelismo o computación distribuida. Dentro de los recursos existentes para el cómputo, existen las unidades de procesamiento central (*CPU*) y las unidades de procesamiento gráfico (*GPU*). Hoy día, ambas unidades permiten la paralelización usando hilos (*threads*). Podemos encontrar *CPUs* con una cantidad cada vez mayor de núcleos e hilos, permitiendo la computación paralela. De forma análoga, las *GPUs* contienen procesadores de centenas o incluso miles de núcleos que, aunque no disponen de la potencia propia de sus homólogos en las *CPUs*, permiten un procesamiento masivamente paralelo, permitiendo obtener buenos modelos en menos tiempo. Basados en este principio, los *frameworks* permiten aprovechar la potencia según en el contexto en el que se ejecuten las redes creadas. Es posible afirmar que los *frameworks* mejoran las etapas pertenecientes al procesado de los datos (*Data Processing*).

A continuación, se abordan los distintos *frameworks* más importantes dentro del gran abanico de posibilidades que es posible encontrar en el mercado.

1. Keras

   Este software es considerado como un envoltorio de distintos *frameworks*. Es usado como una capa de abstracción escrita en Python, la cual permite acceder a funcionalidades y utilidades muy específicas de otros enfoques. El enfoque principal de Keras es poder desarrollar a un alto nivel, independientemente de la herramienta subyacente compatible que esté usando.

   Es posible afirmar que aporta flexibilidad y usabilidad, ya que permite usar complejos *frameworks* de forma casi nativa, siguiendo las mejores prácticas, y crear modelos de DL consistentes de forma rápida usando instrucciones de alto nivel. Además, al trabajar con Python, los modelos resultantes permiten ser comprendidos y depurados por personal no experto ya que no se usan instrucciones a bajo nivel como en otros *frameworks*.

   Keras modulariza cada uno de los elementos existentes en las redes de DL. Desde las capas ocultas, las funciones de activación hasta las neuronas de entrada y salida son representadas como bloques moldeables y configurables a diferentes topologías capaces de aportar el máximo dinamismo posible para crear modelos. En lo que se refiere a las capacidades de paralelización, Keras permite realizar tareas de usando múltiples *CPUs* y *GPUs*.

   Cuenta con licencia *open source* y disfruta del respaldo tanto de empresas importantes en el sector como de la comunidad. No obstante, a pesar de realizar una integración profunda con otros *frameworks*, permitiendo la creación de modelos de forma rápida, abstrayendo las capacidades más importantes y ocultando la complejidad subyacente, la modularidad y simpleza conseguida es fruto de un sacrifico sobre la funcionalidad. En base a lo anterior, no es considerado como un *framework* óptimo para investigar sobre mejoras o nuevas arquitecturas de redes neuronales.

2. Tensorflow

   Tensorflow (4) es uno de los *frameworks open source* más importantes y usados a nivel mundial. Esta herramienta ha sido creada por Google, concretamente por las divisiones de Google Brain y el equipo de investigación de *Machine Learning y Deep Learning* de la propia compañía.

   Tensorflow ha sido diseñado para crear modelos a gran escala, siendo capaz de realizar de forma distribuida entrenamientos y evaluación de redes neuronales. En su interior, representa los cálculos que se realizan y su evolución a través de grafos de flujo de datos. En sí, los nodos de dichos grafos representan las operaciones matemáticas a realizar y las conexiones entre los nodos, también llamadas aristas o tensores, representan matrices multidimensionales con la información resultante.

   Los lenguajes disponibles para realizar operaciones y crear modelos con Tensorflow son Python y C++. Además, Google está trabajando en incluir Java, R y GO como futuras ampliaciones.

Aunque Tensorflow puede ser ejecutado en una única *CPU* o *GPU* (*standalone*), lo habitual es usarlo en arquitecturas distribuidas con múltiples elementos de computación como sistemas distribuidos a gran escala. Para lograr esto, el presente *framework* sigue una arquitectura maestro-esclavo, implementando el patrón de "divide y vencerás" frente a tareas que pueden ser modularizadas. Así, el nodo maestro se encargará de todas las tareas propias de gestión y transferencia de información, mientras que los nodos esclavos realizarán toda la parte intrínseca de computación. Dispone de integración nativa en proveedores como Google Cloud Platform (*GCP*) y Amazon Web Service (*AWS*) permitiendo la posibilidad de usar este *framework* en la nube.

A pesar de que se asocie el DL a la realización de tareas pesadas usando clústeres, es posible ejecutar modelos en dispositivos portables. Tensorflow tiene una versión ligera (*light*) que permite ejecutar modelos profundos en dispositivos Android. Aunque existen restricciones en la cantidad de operaciones soportadas por esta versión, dispone de una integración para móviles y tabletas que permite disponer de un nuevo rango de aplicaciones en dispositivos del día a día. Este tipo de enfoque está presente en otros *frameworks* presentados permitiendo ser instalados y puestos en desarrollos en escenarios empresariales reales y cercanos a las personas.

Debido a su evolución, aceptación por la comunidad, licencia y soporte por parte de Google, Tensorflow se ha convertido en uno de los *frameworks* más populares. No obstante, el uso de este *framework* requiere un *know-how* adecuado sobre modelos de DL, además de un amplio conocimiento en cálculos matemáticos, ya que la propia herramienta actúa a bajo nivel.

3. Torch

Toch (5) es un enfoque escrito en C y C++ que soporta la creación de algoritmos de *Machine Learning* y *Deep Learning*. Fue creado en el año 2002 bajo licencia *open-source* y ha sido ampliamente usado por las grandes compañías y organizaciones del sector. Presenta el concepto de programación orientada a objetos (*Object Oriented Programming, OOP).*

para desarrollar los modelos usando tensores, siendo similar al enfoque usado por Tensorflow. Esta es una peculiar característica que permite realizar operaciones a bajo nivel ayudando a usuarios más avanzados en conceptos informáticos a tratar los modelos de una forma más natural e intuitiva.

Debido a su gran capacidad de edición, Torch es un *framework* ideal para realizar pruebas sobre modelos e investigar su rendimiento (*benchmarking*). Además, al permitir tratar los algoritmos como objetos, otorga la capacidad al usuario final de realizar, modularizar y flexibilizar la creación y configuración de los mismos sin perder velocidad de ejecución. Por contra, es un *framework* que cuyo desarrollo no es activo y tiene poca cuota de mercado.

Durante un tiempo, debido a sus bondades, en el 2018 Facebook presentó su propia variante de este *framework* llamado Pytorch (6). Inspirados en la forma de computar y paralelizar usando *CPUs* y *GPUs*, Pytorch es un enfoque escrito en Python permitiendo interactuar y crear modelos facilitando la tarea de lectura, comprensión y revisión de los mismos, ya que el código usado es de alto nivel. Además, Pytorch usa la técnica de auto diferenciación inversa (*reverse-mode auto-differentiation*) permitiendo cambiar el comportamiento de una red con pocas líneas de código usando grafos dinámicos de computación.

Aunque Pytorch no cuenta con una versión ligera, es posible importar y operar con modelos escritos en otros *frameworks* ya que soporta el protocolo ONNX (*Open Neural Network Exchange*). De forma resumida, Pytorch es un nuevo enfoque de código libre nacido del *framework* Torch que está siendo mantenido por compañías tales como Facebook, Nvidia, Twitter y Uber. Tal ha sido la repercusión del mismo que Uber ha creado Pyro, su propio lenguaje de programación probabilístico usando Pytorch.

4. Microsoft CNTK

Microsoft dispone de un *framework* de uso comercial especializado en *Deep Learning* permitiendo desarrollar modelos para cualquier tipo y tamaño de datos a tratar (7). Es posible usar C++, C# o Python para crear elementos dentro de este conjunto de herramientas cognitivas (*cognitive toolkit*).

Uno de los puntos más competitivos que ofrece es el soporte a las topologías de redes más importantes encontradas en la literatura. Además, esta herramienta soportada la importación de modelos de otros *frameworks* gracias a la aceptación del formato ONNX.

A pesar de no tener una versión orientada a dispositivos móviles, es un *framework* flexible, adaptativo y paralelizable a los recursos disponibles por los clústeres en los que sea ejecutado y respaldado por Microsoft.

5. Caffe

Este enfoque fue creado por el departamento de investigación de la Universidad de Berkeley (8). Dicho *framework* ofrece una capacidad de configurar redes profundas no observado en otros *frameworks*. Cuenta con modelos pre-entrenados para trabajar diferentes tipos de problemas y, si bien es posible definir detalles a

bajo nivel de las capas de dichos modelos, a su vez permite la creación de capas personalizadas para moldear las redes al tipo de problema que se necesite, permitiendo crear configuraciones únicas. Los modelos pre-entrenados que ofrece son totalmente configurables y las nuevas capas que sean añadidas deben hacerse usando C++ o CUDA.

A día de hoy, Caffe no es un *framework* que se siga desarrollando activamente y, por tanto, ampliando así sus funcionalidades. No obstante, existen compañías que debido a sus capacidades siguen usándolo y dando apoyo a la comunidad. Un ejemplo de ello es la compañía Intel, ya que ofrece su propia distribución de Caffe orientada y potenciada al uso de sus chips de gama alta, especialmente dirigidos a servidores de alto rendimiento y supercomputación mejorando la eficiencia del uso de este *framework* en sus equipos.

Gracias a sus múltiples bondades, un equipo de Facebook en el año 2018 lanzó Caffe2 (9). De forma totalmente disruptiva, es un *framework* ligero, modular, prestando una escalabilidad idéntica a su anterior versión y presentando una transición para la compañía mientras desarrollaba su propio *framework*: Pytorch. La principal diferencia con su predecesor recae en la compatibilidad y uso del mismo en dispositivos móviles, junto con su fuerza para aprovechar el máximo hardware disponible independientemente del escenario en el que se ejecute. Además, esta nueva versión ha sido dotada de la posibilidad de escribir sus modelos en código Python, pudiendo realizar una retrocompatibilidad con los modelos desarrollados para la versión anterior e, inclusive, tiene una compatibilidad con los modelos desarrollados en Pytorch.

6. MxNet

Este *framework* ha sido diseñado por la Universidad de Washington y se autoidentifica como un *framework* que permite flexibilidad y eficiencia en la generación de modelos de DL a bajo nivel (10).

Una de las características más significativas es que permite usar programación imperativa y simbólica de forma paralela para facilitar la creación de los modelos. En su interior, MXNet cuenta con tecnología para identificar el tipo de lenguaje usado de forma dinámica y poder, en el mismo momento de ejecución, paralelizar ambos tipos de operaciones. Además, es posible describir todas las operaciones de ejecución que realiza el *framework* en forma de grafo y no de una simple cola. Esto hace posible que el *framework* encuentre la manera más eficiente de procesar la información y entregar modelos robustos sin que consuman demasiados recursos computacionales.

De igual manera que el resto de *frameworks*, soporta paralelización y ejecución en nodos con varias *CPUs* o *GPUs*. Los lenguajes de programación que pueden usarse son Python, R y Julia, Matlab, Wolfram, Perl, Scala y C++.

Si bien no es un *framework* con gran cuota de mercado, permite tener una escalabilidad muy alta y una paralelización casi de forma nativa desde su primer uso. Es por ello que ha sido uno de los primeros *frameworks* en estar presente e integrado en muchos proveedores de *cloud computing*. Cuenta con facilidades para *desplegar* los modelos creados en entornos de producción reales e incluso permite desplegar modelos en móviles, dispositivos IoT o contenedores Docker.

7. Theano

Theano (11) ha sido uno de los *frameworks* pioneros en el campo de la creación de redes neuronales usando computación por GPU. Su desarrollo comienza en 2007 bajo licencia *open source* por la Universidad de Montreal.

El objetivo principal de Theano es la compilación de funciones matemáticas escritas en Python antes de la ejecución y creación de los modelos de DL. Dentro de esta compilación, Theano ayudado de librerías para la transformación de la información, convierte las mismas en estructuras de datos que son usadas por las *CPUs* y *GPUs* de forma más eficiente que el código escrito en Python, aligerando así el procesado de datos (*Data Processing*).

A pesar de ser ampliamente paralelizable y poder ser ejecutado usando varias *CPUs* o *GPUs*, es un proyecto del cual no se prevé que existan nuevas mejoras y funcionalidades de cara a las últimas necesidades que el sector demanda, aunque exista un fuerte apoyo por parte de la comunidad.

B. Software como Servicio (SaaS)

El software como servicio (*SaaS, Software as a Service*) permite a las empresas realizar inversiones en aquellos gastos operativos que lo requieran, evitando realizar inversiones a corto y medio plazo en software y hardware. De forma lógica, las empresas no cubren todas sus necesidades de IT (*Tecnologías de la Información, Information Technology*) usando *SaaS* ya que la delegación total del *know-how* empresarial podría crear una dependencia a largo plazo. En el caso del aprendizaje profundo, cada vez más las empresas proveedoras del sector *cloud* cuentan con servicios dedicados de forma exclusiva tal campo. A excepción de pequeñas configuraciones iniciales, las empresas que opten por este tipo de servicios no tendrán que supervisar infraestructura, red o almacenamiento detrás de la solución escogida, ya que todos

TABLA 1
COMPARATIVA DE FRAMEWORKS

	Plataformas e interfaces programable	Proyecto activo	Procesamiento por GPU	Ejecuciones paralelas
Tensorflow	Python, C/C++, Java, Go, JavaScript, R	Activo	Disponible	Disponible
Keras	Python, R	Activo	Disponible	Disponible
Microsoft Cognitive Toolkit	Python, C++, CLI, .NET	En fase de abandono	Disponible	Disponible
Caffe / Caffe2	Python, MATLAB, C++	Caffe no está activo	Disponible	Disponible
Torch	Lua, C/C++	No está activo	Disponible	Disponible
Pytorch	Python, C++	Activo	Disponible	Disponible
MxNet	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Activo	Disponible	Disponible
Theano	Python	No está activo	Diosponible	Disponible

Fuente: Elaboración propia

estos aspectos recaen íntegramente en el proveedor *cloud* elegido.

Este enfoque es ideal para empresas que no desean realizar una inversión inicial para crear modelos de DL, ya que se centra principalmente en el uso de aplicaciones creadas por proveedores *cloud* abiertas y disponibles al consumo en cualquier momento. Siendo estos últimos los encargados del mantenimiento y almacenamiento de las aplicaciones, delegando toda la responsabilidad sobre los mismos. Además, todas las herramientas *SaaS* disponen de interfaces web para acceder a su configuración, pudiendo configurar sus funcionalidades a través de ellas, aunque también es posible configurarlas usando interfaces de línea de comandos (*CLI, Command Line Interface*).

Es, por tanto, que el uso y creación de modelos a través de *SaaS* permite centrarse en la lógica de negocio, flexibilizando ciertos procesos de negocio de forma contratada a proveedores *cloud*. Este enfoque resulta muy útil siempre que la empresa tenga estructuradas, localizadas y separadas sus capas de negocio (*Business, Service y Data Layers*) pudiendo dividir procesos de las mismas, encajando elementos *SaaS* a la perfección y, por ende, mejorando la versatilidad inicial y reduciendo el tiempo necesario para poner un producto software en producción. No obstante, los modelos de DL propuestos en elementos *SaaS* no son visibles ni conocidos, ya que pertenecen al *know-how* del propio proveedor *cloud*, funcionando exclusivamente como cajas negras. A largo plazo, esta aproximación puede significar una dependencia funcional para las empresas que opten por un uso intensivo de este tipo de soluciones.

En las siguientes subsecciones se abordarán los distintos elementos *SaaS* de los proveedores *cloud* más distinguidos encontrados en el mercado.

1. IBM

IBM trata a sus *SaaS* cognitivos como parte fundamental dentro de su catálogo de productos encabezados por una *suite* de productos bajo el nombre IBM Watson Studio (12). En sí, IBM Watson es un portfolio de aplicaciones y puede ser considerado como una *suite* de herramientas cognitivas y utilidades *cloud* preparadas para tratar problemas orientados al uso de modelos de aprendizaje profundo. Las soluciones de IBM están avaladas por decenas de millones de clientes en todo el mundo y puestas en producción en sectores e industrias de diversa índole, ofreciendo satisfactorios resultados.

Para inferir conocimiento sacando la máxima potencialidad posible de las técnicas de aprendizaje profundo, IBM propone la gobernanza de datos (*Data Governance*) como pilar fundamental usando *Watson Knowledge Catalog*. Un catálogo de datos ayuda a encontrar, preparar, conocer y organizar los datos disponibles. Debido a la importancia del volumen de información dentro de las aplicaciones Big Data, IBM propone soluciones integrales sobre la gestión y catalogación como primera etapa para plantear y desplegar posteriormente el uso de herramientas cognitivas a través de *chatbots* o asistentes virtuales *(Watson Assistant*) o descubrir conocimientos ocultos en los datos (*Watson Discovery*). Si bien estos puntos suponen unos de los grandes avances que propone IBM en su catálogo, es posible usar herramientas *SaaS* para analizar e interpretar el lenguaje natural (*Watson Natural Language Classifier*), analizar textos para extraer entidades, metadatos y opiniones (*Watson Natural Language Understanding*), analizar el tono de las emociones y contenido escrito en ellos (*Watson Tone Analyzer*) o llegar incluso a predecir las necesidades y las características de personalidad a través de los mismos (*Watson Personality Insights*).

TABLA 2
COMPARATIVA DE "SOFTWARE AS A SERVICE" ORIENTADOS A APRENDIZAJE PROFUNDO EN PROVEEDORES CLOUD

	IBM Cloud	Google Cloud Platform	Microsoft Azure	Amazon Web Service
Nombre plataforma integral SaaS	Watson Knowledge Catalog	Vertex AI	Azure Cognitive Services	Amazon SageMaker Studio
Análisis de imágenes	Watson Visual Recognition	Google Vision API	Azure Computer Vision y Custom Vision	Amazon Lookup for Vision y Amazon Panorama
Análisis de video	Watson Visual Recognition	Google Visión API	Azure Computer Vision y Custom Vision	Amazon Lookup for Vision
Transformación de texto a voz	Watson Text to Speech	Google Speech API	Azure Text to Speech	Amazon Polly
Transformación de voz a texto	Watson Speech to Text	Google Speech API	Azure Speech to Text	Amazon Transcribe
Ofrecer preguntas sobre texto	-	-	Azure QnA Maker	-
Identificación de interlocutores en conversación	-	-	Azure Speaker Recognition	-
Detectar contenido ofensivo en textos	-	-	Azure Content Moderator	-
Traducción	Watson Language Translator	Google Translate API	Azure Translator	Amazon Comprehend
Chatbot	Watson Assistant	Google Dialogflow	Azure Bot Service	Amazon Lex
Asistentes virtuales	Watson Assistant	Google Dialogflow	Azure Bot Service	Amazon Lex
Procesamiento del lenguaje natural	Watson Natural Language Classsifier	Google Natural Language API	Azure Language Understanding: LUIS	Amazon Comprehend
Clasificación de textos	Watson Natural Language Classifier	AutoML	Azure Text Analytics	Amazon Textract
Análisis del tono y emociones	Watson Tone Analyzer	Google Natural Language API	Language Understanding: LUIS	Amazon Comprehend
Análisis de la personalidad	Watson Personality Insights	-	Azure Personalizer	-
Detección de anomalías en datos	-	-	Azure Anomaly Detector	Amazon Fraud Detector y Amazon Rekognition
Sistema de recomendaciones personalizadas	-	-	Azure Bing API	Amazon Personalize
Sistema de búsquedas	-	-	Azure Bing API	Amazon Kendra
Análisis de métricas empresariales	-	-	-	Amazon Lookup for Metrics
Previsión de la demanda de ventas o suministros	-	-	-	Amazon Forecast
Soporte para sectores	-	-	-	Industrias y Sanidad
Interoperabilidad con otros frameworks	Tensorflow, Keras y PyTorch	Tensorflow, Keras y PyTorch	Tensorflow, PyTorch y Microsoft Cognitive Toolkit	Tensorflow, MxNet, Keras, Torch, PyTorch y Caffe2
Autocreación de modelos	AutoML y AutoAI	AutoML	Azure Machine Learning	Amazon AutoGluon y Amazon SageMaker Autopilot

Fuente: Elaboración propia

Dejando a un lado los datos estructurados, es posible trabajar con datos no estructurados para crear aplicaciones inteligentes, siendo uno de los proveedores que más abanico de posibilidades ofrece. Para ello IBM Watson Studio cuenta tanto con aproximaciones para convertir el audio y la voz en texto escrito (*Watson Speech to Text*), como herramientas que convierte el texto a audio (*Watson Text to Speech*), traducciones (*Watson Language Translator*). Sin dejar de lado la parte visual, la suite cuenta con *Watson Visual Recognition* para trabajar con contenido de imágenes y videos.

Con referencia a la auto creación de modelos de forma automática, IBM propone *AutoML* (*Automated Machine Learning*) y *AutoAI* (*Automated IA*) como una propuesta robusta a la hora de crear y desplegar modelos en producción en entornos reales *cloud* sin tener habilidad profunda en el dominio de las técnicas acelerando los ciclos de vida. De la misma forma que su competencia, permite la integración e interoperabilidad de modelos usando frameworks como PyTorch o Tensorflow o lenguajes como Python, R o Scala.

2. Google

Google recoge todos sus *SaaS* cognitivos en una misma plataforma llamada Vertex IA (13). Es una herramienta que cuenta con una interfaz

unificada para realizar todas las etapas presentes a la hora de desarrollar modelos de aprendizaje profundo usando los servicios y productos del catálogo de Google.

Vertex IA integra la capacidad de crear modelos de aprendizaje profundo, sin tener altas capacidades de programación, orientados y adaptados a problemas de reconocimiento en imágenes y videos (*Vision API*), problemas orientados a la traducción (*Speech API y Translate API*), procesamiento del lenguaje natural (*Speech API*) y creación de *chatbots* (*Dialogflow*). Todos estos servicios cognitivos ofrecen la posibilidad de trabajar con cualquier conjunto de datos, sean estructurados o no, sin hacer grandes esfuerzos.

Otras de las ventajas de los modelos creados a través de Vertex AI es la interoperabilidad de los mismos. Google plantea una conexión intrínseca de los mismos con *frameworks* de código abierto como Tensorflow o Pytorch. Esta característica también se hace extensible a que sean usados dentro de Big Query (*BQ*). BQ es un almacén de datos columnar para escenarios Big Data usando lenguaje SQL. Con este producto, es posible usar y aplicar modelos creados en Vertex AI como funciones de SQL frente al conjunto de datos almacenados, proporcionando una versatilidad para trabajar con los mismos. De manera inversa, es posible exportar datos almacenados en BQ para ser consumidos por Vertex AI.

Vertex AI ayuda a automatizar, supervisar y controlar de forma integral todos los pasos de *data processing* de forma automatizadas gracias a la creación de flujos de trabajos o *pipelines* administrando e implementando, a escala de las necesidades, la infraestructura necesaria para desarrollar aplicaciones inteligentes.

3. Microsoft Azure

El catálogo de servicios de la nube de Microsoft se ve recogido en *Azure Cognitive Services* (14). Esta *suite* de servicios de aprendizaje profundo pone a disposición de los usuarios y empresas herramientas cognitivas listas para ser usadas y puestas en producción sin tener conocimientos profundos en el área.

El catálogo de servicios de Azure se crea en base a distintas áreas canónicas. En un primer lugar, encontramos las herramientas para identificar y analizar el contenido de imágenes y videos (*Computer Vision y Custom Vision*), además cuenta con un servicio para reconocimiento facial para detectar personas y emoticonos en las imágenes. En segundo lugar, encontramos servicios de reconocimiento de voz que permiten identificar a los interlocutores de una conversación (*Speaker Recognition*), servicios de traducción (*Speech Translation*) o enfoques para convertir voz a texto (*Speech to Text*) o texto a voz (*Text to Speech*). En una tercera posición, Azure ofrece servicios para extraer significados a partir de un texto no estructurado, dando la posibilidad de crear una capa de preguntas y respuestas a partir del mismo (*QnA Maker*), detectar opiniones y entidades (*Text Analytics*) o recopilar información del lenguaje natural o *bots* (*Languaje Understanding*). Por último, y de forma muy singular, Azure ofrece la capacidad de implementar modelos para la toma de decisiones de forma inteligente. En este punto encontramos servicios para detectar anomalías de forma temprana (*Anomaly Detector*) o incluso detectar contenido ofensivo (*Content Moderator*).

Azure ofrece un catálogo extenso orientado a la aplicación de *SaaS* a entornos reales envueltos de macrodatos. Para ello es posible interactuar con los modelos usando Java, Scala, Python y R. En el caso de aplicaciones ampliamente distribuidas, es posible usar contenedores (*Azure Kubernetes Service*) o Azure Databricks como plataforma de análisis basadas en *Apache Spark,* uno de los motores de procesamiento de datos a escala Big Data más importantes a nivel de procesamiento de datos.

4. Amazon Web Service

Amazon Web Service (*AWS*) es uno de los mayores proveedores *cloud* de la industria y ofrece uno de los abanicos más amplio de servicio *SaaS* disponibles hasta la fecha, dando servicio a cientos de miles de clientes. Cuenta con modelos previamente entrenados, orientados al uso de aprendizaje profundo, listos para ser integrados en aplicaciones inteligentes. Todos los servicios disponibles mantienen una integración sencilla para poder ser consumidos para crear recomendaciones personalizadas, modernizar procesos de negocio, mejorar la seguridad o aumentar el compromiso con los clientes finales. Al igual que el resto de proveedores *cloud*, para usar los modelos propuestos por *AWS* no hace falta tener experiencia en el área del aprendizaje profundo.

AWS cuenta con *Amazon SageMaker Studio* (15) como pináculo central para crear modelos a gran escala en un entorno de desarrollo propio de la compañía. No obstante, el catálogo de *AWS* contiene las herramientas cognitivas canónicas de la industria. Es posible encontrar soluciones para recomendaciones personalizadas (*Amazon Personalize*), realizar búsquedas inteligentes dentro de sitios web o aplicaciones (*Amazon Kendra*), procesar y extraer conocimiento y entidades de millones de documentos (*Amazon Textract*), extraer información de texto no estructurado (*Amazon Comprehend*) o realizar traducciones automáticas (*Amazon Translate*). Por otro lado, Amazon también se centra en crear *SaaS* para el análisis de métricas empresariales (*Amazon Lookout For Metrics*) llegando a detectar anomalías, dando previsión de demanda (*Amazon Forecast*) o previniendo el fraude de actividades en línea (*Ama-*

zon Fraud Detector). A nivel de procesamiento de imágenes y videos, Amazon ofrece la detección de defectos en los productos (*Amazon Lookout for Vision*), la capacidad de mejorar y automatizar de tareas de inspección visual (*Amazon Panorama*) y la clasificación de los mismos (*Amazon Rekognition*). Por último, a nivel de texto y lenguaje ofrece la capacidad de usar *bots* y agentes virtuales (*Amazon Lex*) y la capacidad de transformar voz a texto (*Amazon Transcribe*) y viceversa (*Amazon Polly*).

AWS pone a disposición soluciones específicas totalmente adaptadas a sectores industriales y sanitarios y arquitecturas en la nube para ser usadas en casos de uso concretos con el fin de obtener resultados con una mínima inversión de esfuerzo y tiempo. Esta funcionalidad se ve reflejada en *SageMaker Autopilot* como herramienta para crear de forma automática modelos de aprendizaje profundo. Además, a nivel empresarial, *AWS* ofrece detección de comportamiento anormal en maquinaria con análisis predictivo (*Amazon Monitron*) y detección de comportamiento anormales mediante sensores (*Amazon Lookup for equipment*). A nivel del sector sanitario, es posible realizar transcripciones médicas (*Amazon Transcribe Medical*) para transcribir audios de manera precisa y documentar conversaciones de ámbito clínico. También es posible realizar análisis de textos médicos (*Amazon Comprehend Medical*) y realizar búsquedas con KPIs (*Key Performance Indicators*) sobre datos médicos (*Amazon Healthlake*).

CONCLUSIONES

El DL es un subtipo de aprendizaje automatizado en el cual se implementan modelos matemáticos basados en redes neuronales para abordar problemas complejos tales como realizar análisis de audio, detección de rostros en imágenes o seguimiento de objetos en videos entre otros. Las redes usadas siguen una topología de neuronas organizadas por capas y, según su distribución, se adecúan a resolver determinados tipos de problemas . A pesar de que las redes neuronales requieren de grandes conjuntos de datos para su en y una fase de computación intensa para su entrenamiento, dificultando su puesta en marcha inicial, las empresas cuentan con herramientas tales como la computación en la nube y los *frameworks* de desarrollo para facilitar este proceso. La computación en la nube otorga la capacidad de disponer de recursos computacionales suficientes, además de disponer de herramientas *SaaS* listas para ser usadas. Por otro lado, los *frameworks* de desarrollo facilitan el desarrollo nativo e incrementan la capacidad de adopción de las empresas a integrar modelos de aprendizaje profundo dentro de su capa de negocio, otorgando una ventaja competitiva sobre su competencia. Toda esta combinación de técnicas computacionales, tecnologías e infraestructuras, permiten que las organizaciones puedan aprovechar al máximo sus activos de datos.

NOTAS

[1] https://deepmind.com/research/case-studies/alphago-the-story-so-far
[2] https://www.bbc.com/news/technology-33347866
[3] https://algorithmwatch.org/en/google-vision-racism/
[4] https://www.tensorflow.org
[5] http://torch.ch/
[6] https://pytorch.org/
[7] https://docs.microsoft.com/en-us/cognitive-toolkit/
[8] https://caffe.berkeleyvision.org/
[9] https://caffe2.ai/
[10] https://mxnet.apache.org/versions/1.8.0/
[11] https://github.com/Theano/Theano
[12] https://www.ibm.com/es-es/cloud/watson-studio
[13] https://cloud.google.com/vertex-ai
[14] https://docs.microsoft.com/es-es/azure/cognitive-services/what-are-cognitive-services
[15] https://aws.amazon.com/es/sagemaker/studio/

REFERENCIAS

C.-W. Tsai, C.-F. L.-C. (2015). Big data analytics: a survey. *Journal of Big Data*, 1-32.

Calegari, R. C. (2020). On the integration of symbolic and sub-symbolic techniques for XAI: A survey. *Intelligenza Artificiale*. 14. 1-25. 10.3233/IA-190036.

CM, B. (2006). Pattern recognition and machine learning (information science and statistics). *Springer*.

Dastin, J. (2017). Obtenido de "Amazon scraps secret AI recruiting tool that showed bias against women". *Reuters*. https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

FV, V. (2016). *The neural network zoo*. Obtenido de https://www.asimovinstitute.org/neural-network-zoo/

I. Goodfellow, Y. B. (2017). Deep learning. *MIT Press*.

Jieyu Zhao, T. W.-W. (2017). Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints. 2979-2989. 10.18653/v1/D17-1323.

Liermann, V. (2021). Overview Machine Learning and Deep Learning Frameworks. En *The Digital Journey of Banking and Insurance, Volume III: Data Storage, Data Processing and Data Analysis* (págs. 187-224). Springer International Publishing.

Mathias Kraus, S. F. (2020). Deep learning in business analytics and operations research: Models, applications and managerial implications. *European Journal of Operational Research*, 628-641.

Mitchell, M. I. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 255-260.

Ng, A. (2016). Machine learning yearning: Technical strategy for AI Engineers, in the era of deep learning. *Harvard Business Publishing*.