CAPÍTULO III

Un sistema de inteligencia artificial rápido y eficiente energéticamente*

José Duato

Las redes neuronales son una herramienta de inteligencia artificial muy potente, pero adolecen de ciertas limitaciones que dan lugar a una falta de transparencia y a un elevado consumo de energía, especialmente durante el entrenamiento. En este capítulo, tras revisar las principales aportaciones de las matemáticas y la tecnología para optimizar los procedimientos asociados y reducir el consumo de energía, se propone un nuevo sistema de inteligencia artificial para atacar los problemas mencionados. Dicho sistema se basa en un modelo lineal, en el cual cada salida se expresa como una suma ponderada de las entradas, siendo los pesos ajustables. La novedad estriba en que cada entrada, en lugar de tener un solo peso asociado, dispone de varios pesos cuya aportación se suma. Además, dichos pesos no están siempre activos. De hecho, del conjunto de pesos para cada entrada solamente se activa un subconjunto de los mismos para cada muestra. Esta flexibilidad permite aplicar el nuevo sistema a problemas complejos, alcanzando una precisión similar a la de una red neuronal. Sin embargo, el sistema propuesto genera una sencilla explicación para cada muestra y consigue, además, reducir mucho los tiempos de reentrenamiento. Se presenta el diseño del nuevo sistema de inteligencia artificial y se evalúa su rendimiento y precisión.

Palabras clave: inteligencia artificial, consumo de energía, redes neuronales.

* Agradecimientos: A Enrique Quintana-Ortí (UPV), Manel Dolz (UJI) y, sobre todo, José Ignacio Mestre (UJI) por sus inestimables aportaciones en el desarrollo del sistema de inteligencia artificial descrito en este capítulo.

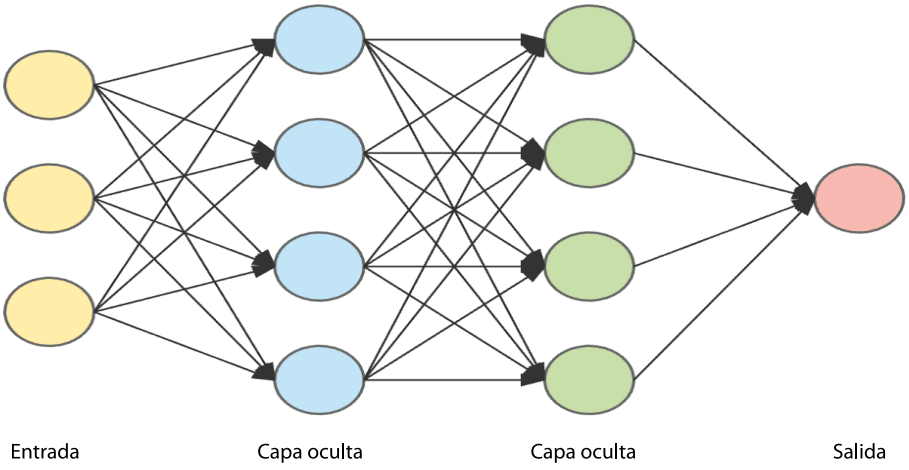
1. INTRODUCCIÓN

Las redes neuronales (Nielsen, 2019) son una herramienta de inteligencia artificial ampliamente utilizada en la actualidad para un gran número de aplicaciones. Es una herramienta muy potente, capaz de capturar la información contenida en grandes volúmenes de datos, establecer relaciones y aplicarlas posteriormente para realizar inferencia, es decir, realizar predicciones para nuevos datos no utilizados durante el entrenamiento.

La capacidad de establecer relaciones requiere que una red neuronal sea capaz de reproducir el comportamiento de cualquier función multivariable. Hay múltiples formas de conseguir esta flexibilidad, pero en el caso de las redes neuronales se ha optado por concatenar múltiples capas de unos elementos denominados neuronas artificiales (o, simplemente, neuronas) cuyo comportamiento se puede definir mediante un sencillo modelo matemático. Dicho modelo matemático consiste en aplicar una función no lineal predefinida, denominada función de activación, a la suma ponderada de las entradas de la neurona más un sesgo. Mientras que la función de activación es fija, los pesos aplicados a las entradas y el sesgo son ajustables. Las neuronas se organizan en varias capas, normalmente con patrones de interconexión regulares entre las mismas, como puede verse en la figura 1. Tradicionalmente se ha distinguido entre: a) la capa de entrada, que no contiene neuronas sino solamente los valores de las componentes de cada muestra, b) las capas ocultas, que contienen neuronas cuya función de activación puede variar de una capa a otra pero se utiliza la misma dentro de cada capa, y c) la capa de salida, con una neurona por cada salida de la red, que puede incorporar alguna función especial (por ejemplo, para convertir los valores de las diferentes salidas de la red neuronal en probabilidades, es decir, valores entre cero y uno y cuya suma total vale uno).

Figura 1.

Red neuronal con dos capas ocultas



Fuente: Elaboración propia.

Para entrenar una red neuronal se define una función de coste, que es una función que penaliza las desviaciones de los valores de salida calculados por la red para una muestra dada respecto de los valores esperados (por ejemplo, el valor medio de los cuadrados de dichas desviaciones). Seguidamente se aplica un algoritmo de optimización que, a partir de un conjunto de muestras de entrenamiento, calcula los valores de los pesos y sesgos de las neuronas que minimizan el valor de la función de coste para las muestras de dicho conjunto. Es necesario validar la precisión que es capaz de alcanzar una red neuronal entrenada, usando para ello un conjunto de muestras de validación, diferentes de las muestras de entrenamiento. Es importante que la precisión alcanzada con las muestras de validación sea similar a la alcanzada con las muestras de entrenamiento. En caso contrario, estaríamos ante un caso de sobreentrenamiento, situación que se da cuando los pesos de la red neuronal se han ajustado excesivamente a las muestras de entrenamiento, lo que obligaría a repetir el entrenamiento e incluso a buscar una configuración de red neuronal más adecuada al problema a resolver.

La gran potencia de las redes neuronales proviene de la combinación de tres factores. En primer lugar, el uso de funciones no lineales en cada una de las neuronas, lo que permite modelar comportamientos altamente no lineales. En segundo lugar, la implementación de un gran número de capas de neuronas (en particular, en las denominadas redes neuronales profundas), permite adaptar el comportamiento de la red neuronal a cualquier grupo de funciones, por complejas que sean. Desde otro punto de vista, se podría considerar que cada capa de neuronas toma pequeñas decisiones, que se basan en las decisiones de la capa anterior. Esta concatenación de pequeñas decisiones construye el equivalente a decisiones mucho más complejas. Y en tercer lugar, la capacidad de entrenamiento a partir de un conjunto de datos de entrenamiento, mediante la minimización de una función de coste. Este proceso se complica porque el número de operaciones aritméticas a realizar para entrenar una red profunda con un conjunto enorme de muestras de entrenamiento es elevadísimo, y puede llegar a tardar semanas e incluso meses en computadores muy potentes.

Debido a la no linealidad de las redes neuronales, el proceso de entrenamiento es iterativo. En cada iteración se reduce el valor de la función de coste. Los métodos más habituales se basan en calcular el vector gradiente (Nielsen, 2019), el cual indica el sentido y magnitud en los que debe modificarse cada peso de la red para progresivamente reducir el valor de la función de coste. A pesar de las numerosas optimizaciones introducidas en el proceso de entrenamiento, este proceso converge muy lentamente. Con frecuencia se requieren varias decenas de épocas, donde en cada época se procesa una vez el conjunto de entrenamiento completo. Uno de los motivos principales de esta lenta convergencia es el denominado problema del gradiente evanescente, que consiste en que algunas componentes del vector gradiente tienen valores cercanos a cero, por lo que los pesos correspondientes apenas se modifican en cada iteración. Este problema empeora con la profundidad de la red neuronal.

El problema de la lentitud, coste y consumo energético asociado que conlleva el entrenamiento se ve notablemente agravado en aquellas aplicaciones en las que se generan nuevos datos de forma frecuente y se requiere reentrenar la red neuronal para incorporar el conocimiento contenido en los nuevos datos. Las redes neuronales sufren el denominado olvido

catastrófico, que consiste en la pérdida no deseada de conocimientos ya adquiridos cuando se entrena solamente con nuevas muestras. Este comportamiento no deseado deriva del método de entrenamiento utilizado, consistente en minimizar una función de coste, ya que las muestras que no se utilicen durante el proceso de entrenamiento no influirán en los valores que se calculen para los pesos. Y este problema se agrava con el paso del tiempo, ya que los modelos utilizados son cada vez de mayor tamaño (Amodei y Hernandez, 2018).

Finalmente, otro problema completamente distinto se deriva del comportamiento de las redes neuronales como una caja negra. No se puede saber el efecto de cada peso de una red. Como consecuencia, una red neuronal no proporciona ninguna explicación de por qué se infiere una predicción determinada para una muestra dada. Este problema, que resulta insignificante en algunas aplicaciones, es de vital importancia en otras, tales como los recomendadores (por ejemplo, para conceder o denegar un préstamo) o los sistemas de ayuda al diagnóstico clínico. En estas aplicaciones, no es posible utilizar un sistema de inteligencia artificial si éste no proporciona una explicación clara del motivo por el que genera una determinada predicción.

La lentitud del entrenamiento no solo supone un tiempo de espera y un coste económico elevados. El entrenamiento con conjuntos de datos de gran tamaño combinado con redes neuronales profundas, una convergencia muy lenta y un gradiente que se desvanece generan un enorme consumo de energía. Por ejemplo, aunque estrictamente no sea una red neuronal, el entrenamiento del GPT-3 utilizado en ChatGPT requirió 1,287 gigavatios hora (The Brussels Times Newsroom, 2024). Esta cantidad es mayor que la energía generada en una planta nuclear durante una hora. Los tiempos de entrenamiento se pueden acortar mediante clústeres muy grandes equipados con potentes aceleradores tipo GPU. En el caso de ChatGPT se redujo el tiempo de entrenamiento a varios días. Si se hubiera ejecutado en una sola GPU, hubiera tardado más de un millón de horas, es decir, más de cien años.

2. REDUCCIÓN DEL CONSUMO DE ENERGÍA

Antes de plantear nuevas propuestas para reducir el consumo energético de las redes neuronales, conviene revisar los principales avances que se han producido a lo largo de varias décadas de investigación y desarrollo. Estos avances se han producido principalmente en tres frentes. Por una parte, tenemos los avances en los algoritmos matemáticos empleados, siendo los más relevantes aquellos que consiguen reducir el número de operaciones aritméticas a realizar y/o contribuyen a reducir el movimiento de datos, ya que además de reducir el consumo también mejoran las prestaciones. Por otra parte, tenemos los avances combinados en tecnología de fabricación de chips y en arquitectura de computadores, que han dado como resultado los sofisticados aceleradores de cálculo para inteligencia artificial que existen en la actualidad.

2.1. Avances en los algoritmos

Dado que las neuronas artificiales incorporan una función de activación no lineal y que el objetivo del proceso de entrenamiento es ajustar los pesos de las neuronas para que el valor de la función de coste alcance su valor mínimo para un determinado conjunto de muestras de entrenamiento, los investigadores se decantaron por sencillos métodos iterativos, tales como el método del descenso del gradiente. Éste es un algoritmo de optimización iterativo de primer orden para minimizar cualquier función multivariable diferenciable.

La idea de este algoritmo es avanzar, en cada iteración, en dirección opuesta al gradiente de la función en el punto actual, ya que ésta es la dirección en la que se reduce más rápidamente el valor de la función de coste. Se han realizado muchísimos avances en la mejora de los algoritmos de entrenamiento, pero sin duda las dos contribuciones más relevantes han sido el algoritmo de retropropagación (*backpropagation*) para el cálculo del gradiente y la versión estocástica del método de descenso del gradiente.

El algoritmo de retropropagación parte de la observación de que las expresiones matemáticas para las componentes del gradiente para una determinada capa de la red neuronal son idénticas a las expresiones para la capa posterior de la red, pero añadiendo un par de factores en cada nueva componente. Este algoritmo ha permitido que al calcular el gradiente, en lugar de calcular cada componente desde cero, se puedan reutilizar los cálculos de las componentes de una capa para calcular las componentes de la capa anterior (que es la siguiente que se calcula, ya que se procesan las capas en orden inverso). En redes neuronales profundas, este algoritmo puede llegar a reducir el número de operaciones aritméticas a menos de la centésima parte respecto al algoritmo original.

En teoría, en cada iteración se deberían incluir todas las muestras de entrenamiento en el cálculo del gradiente de la función de coste en ese punto. Sin embargo, se ha podido comprobar que basta con una pequeña parte (denominada lote o *batch*) de dicho conjunto de muestras, seleccionado aleatoriamente, para conseguir buena precisión en el cálculo del gradiente. Esta variante del algoritmo de descenso del gradiente, denominada estocástica, consigue también reducir el número total de operaciones aritméticas a menos de la centésima parte respecto a la versión no estocástica. Afortunadamente, esta reducción puede combinarse con la conseguida por el algoritmo de retropropagación, dando lugar a una reducción combinada que puede superar los cuatro órdenes de magnitud (es decir, reducir el número de operaciones en un factor de 10.000).

Hay muchas otras optimizaciones que consiguen resultados menos vistosos, pero no por ello menos importantes, ya que todos contribuyen a mejorar la eficiencia energética. Por citar algunas de estas optimizaciones:

- Proceso por lotes. En lugar de procesar secuencialmente las muestras de un lote, se pueden procesar todas a la vez. Al procesar lotes de muestras, los productos matriz por vector que deben ejecutarse durante el entrenamiento de una muestra se con-

vierten en productos matriz por matriz. Y este cambio es aprovechado por los aceleradores actuales para conseguir unas prestaciones mucho más elevadas y un consumo de energía mucho menor.

- Funciones de activación más sencillas, como por ejemplo la ReLU (Rectified Linear Unit) (ver figura 5). Estas funciones y sus derivadas son más rápidas de calcular y consumen menos energía que otras funciones tradicionales tales como la sigmoide.
- Transferencia de conocimiento. En algunas aplicaciones es posible utilizar modelos entrenados para una aplicación en otra aplicación diferente. De este modo, un modelo puede basarse en conocimientos previos para dominar nuevas tareas, y se puede seguir entrenando el modelo a pesar de tener datos limitados.
- Métodos de poda. Permiten eliminar muchas neuronas sin perder apenas precisión, especialmente cuando la red neuronal estaba sobredimensionada, lo que normalmente redunda en una reducción del número de operaciones a realizar. Pero si la poda no es estructurada, puede resultar difícil aprovecharla debido a la arquitectura interna de los aceleradores actuales.
- Métodos de inicialización de los pesos. Una inicialización de los pesos que reduzca la probabilidad de obtener valores muy pequeños en el cálculo de las componentes del gradiente normalmente conseguirá reducir el número de iteraciones necesarias durante el entrenamiento.
- Hiperparámetros variables dinámicamente. Los hiperparámetros son parámetros globales del sistema o del algoritmo de entrenamiento. Su valor puede ser fijado por el usuario. Algún hiperparámetro, como la velocidad de aprendizaje (que es el factor que se aplica al gradiente para determinar cuánto se progresa en cada iteración del entrenamiento) es crucial, pudiendo afectar mucho al número de iteraciones necesarias durante el entrenamiento, y pudiendo hacer incluso que el entrenamiento no converja ni termine nunca. Una optimización consiste en variar dinámicamente el valor de este hiperparámetro para conseguir los mejores resultados.

2.2. Avances en la tecnología

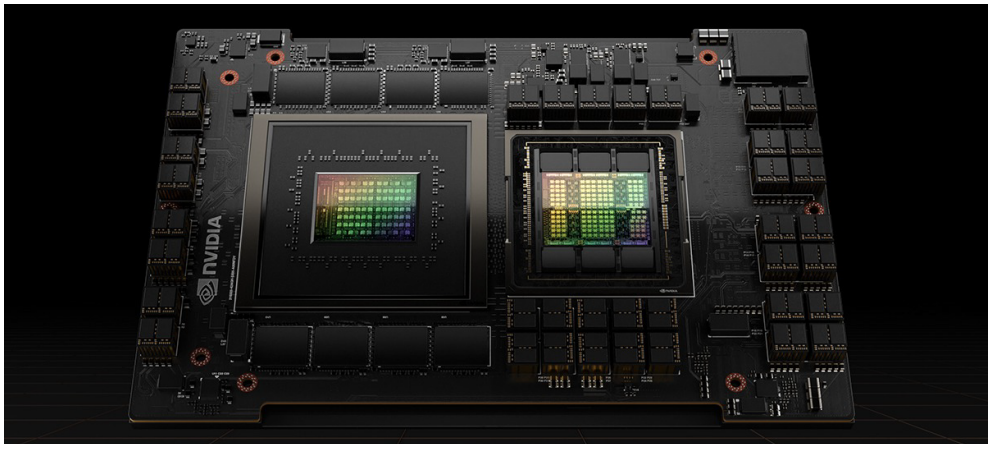
Aunque los avances en los algoritmos han sido trascendentales para conseguir unos requisitos de potencia de cálculo y unos niveles de eficiencia energética que hacen viables los sistemas de inteligencia artificial actuales, su contribución queda ensombrecida por la descomunal magnitud de los avances en la tecnología. Para tener una idea de conjunto de la aportación de la tecnología y su evolución a lo largo de las últimas ocho décadas, basta con comparar el supercomputador actual con mayor eficiencia energética con el primer ordenador electrónico.

El ordenador más eficiente es el que encabeza la lista del Green500 en junio de 2024 (Green500, 2024), el cual es capaz de ejecutar 72,7 GigaFLOPS por vatio. Está basado en el superchip de NVIDIA denominado GH200 Grace Hopper Superchip (NVIDIA GH200

Grace Hopper Superchip, 2024), cuya imagen sin disipador de calor puede verse en la figura 2. Si comparamos este ordenador con el primer ordenador electrónico, el ENIAC, consigue una mejora de la eficiencia energética en un factor de 36 billones ($3{,}6\times10^{13}$). Esta impresionante mejora, que es más de mil millones de veces mayor que la alcanzada por las principales mejoras en los algoritmos, se ha conseguido combinando tecnología (escala de integración de los chips, familia lógica CMOS) y arquitectura (por ejemplo, los aceleradores tipo GPU).

Figura 2.

NVIDIA GH200 Grace Hopper Superchip



Fuente: Extraída de https://www.nvidia.com/es-es/data-center/grace-hopper-superchip/

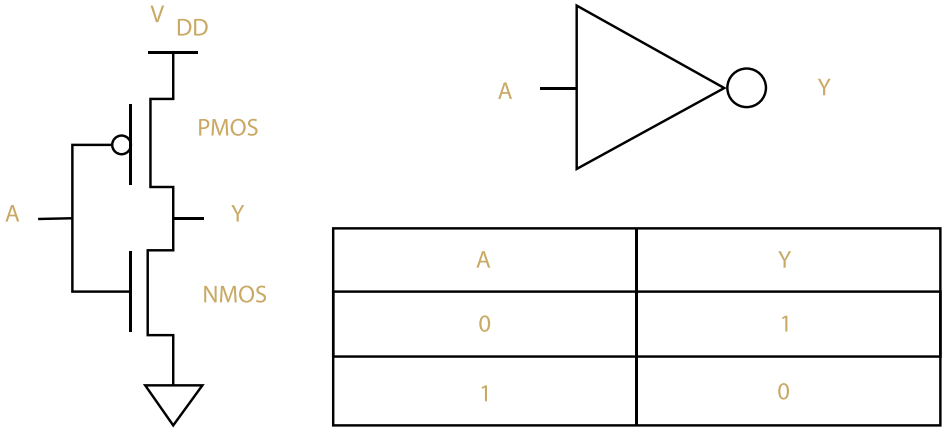
Por otra parte, la elevada escala de integración de los transistores ha permitido pasar de un tamaño de transistor de 5 mm a 3 nm, lo que ha supuesto un factor de reducción en el tamaño superior a un millón de veces. Como los transistores se fabrican sobre chips de dos dimensiones, la integración ha permitido empaquetar hasta 200.000 millones de transistores, como es el caso del GH200 Grace Hopper Superchip (y mucho más, hasta 4 billones de transistores, si se utiliza una oblea entera de silicio, como es el caso de los chips WSE-3 de Cerebras). La menor distancia entre transistores ha permitido además reducir el consumo de energía para mover datos y subir la frecuencia de reloj de MHz a GHz.

Un componente importante de la tecnología es la familia lógica utilizada en los chips. En la década de los 80 existían muchas familias lógicas con diferentes características, desde la rápida ECL, pasando por la popular TTL, las PMOS y NMOS utilizadas en los primeros microprocesadores, hasta la lenta CMOS de muy bajo consumo. De ellas, únicamente queda la CMOS, que es la que consigue los consumos más bajos de energía, y que ha mejorado muchísimo sus prestaciones a lo largo de cuatro décadas. Como puede verse en la figura 3, gracias al uso de transistores MOS complementarios, cuando un transistor conduce, el otro no lo hace, consiguiendo así unos consumos bajísimos. No obstante, a pesar de su bajo con-

sumo, cuando se integran 200.000 millones de transistores en un chip, y a pesar de implementar múltiples optimizaciones para minimizar el consumo de energía, dicho chip puede llegar a requerir 500 W de potencia.

Figura 3.

Puerta lógica CMOS: inversor



A	Y
0	1
1	0

Fuente: Extraída de https://www.geeksforgeeks.org/cmos-logic-gate/

Es importante mencionar que cuando hablamos de tecnología, muchas veces nos olvidamos de la aportación de la arquitectura de los computadores, que es la ciencia que se ocupa de la organización de los componentes y sus interconexiones, tanto dentro un chip como de los chips entre sí, para conseguir objetivos diversos tales como elevadas prestaciones, bajo consumo, buena fiabilidad, etc. Para hacernos una idea de la importancia de la arquitectura del chip, basta con comparar dos chips actuales (que usan tecnologías de fabricación similares) con arquitecturas muy diferentes. Si nos fijamos en un procesador (CPU) de uso general, encontramos que uno de los más rápidos (AMD Ryzen 9 7950X3D 16-Core) es capaz de ejecutar 7,29 GigaFLOPS por núcleo (lo que conseguiría un pico de 116,64 GigaFLOPS con los 16 núcleos), mientras que un acelerador tipo GPU diseñado específicamente para inteligencia artificial alcanza 7,1 TeraFLOPS, es decir, más de 60 veces más rápido. Pero no solamente es mucho más rápido sino que también es mucho más eficiente energéticamente, en una proporción bastante similar.

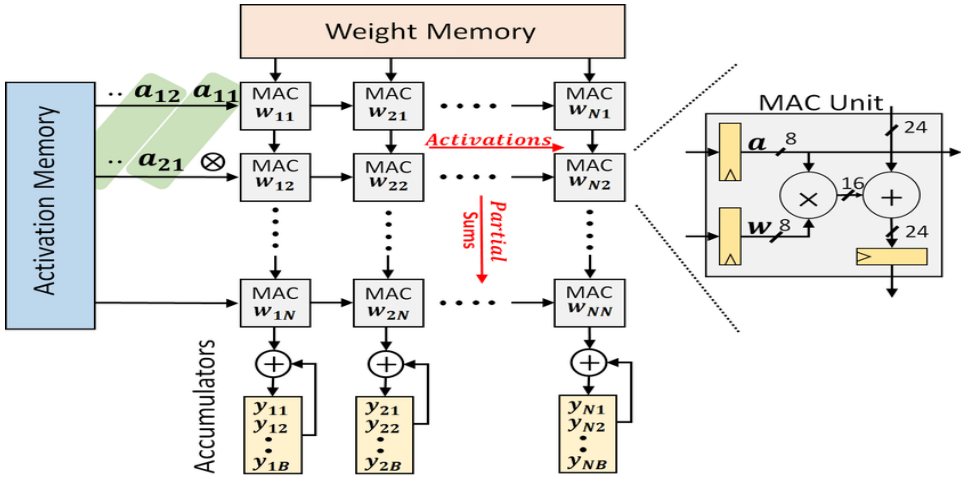
El punto de partida para que una GPU consiga mucha más potencia de cálculo que una CPU es que dedica un porcentaje mucho más grande de los transistores a las unidades aritméticas, con lo que en una GPU hay un número mucho más grande de estas unidades. Pero con ello no se conseguiría ni mayor eficiencia energética ni tampoco mayor potencia de cálculo, ya que la memoria no tendría suficiente ancho de banda para poder alimentar con datos dichas unidades aritméticas. Así pues, se requiere una arquitectura diferente.

Los aceleradores actuales (sean GPUs u otras variantes) son energéticamente mucho más eficientes que una CPU gracias a que reutilizan muchas veces cada dato leído de memoria. Su arquitectura deriva del concepto de procesador sistólico, en el cual:

- Las unidades aritméticas de procesamiento se organizan de forma estructurada, con frecuencia formando una estructura bidimensional en la que cada unidad se conecta únicamente a sus vecinas (en la figura 4, las vecinas de la derecha y de abajo).
- Los datos avanzan de forma sincronizada, de modo que cada dato se mueve de una unidad a la siguiente a cada ciclo de reloj. Varias filas y varias columnas de datos se mueven a la vez.
- Cada vez que un dato que avanza en vertical coincide con otro que avanza en horizontal, se opera con dichos datos en la unidad aritmética correspondiente (en la figura 4, una multiplicación seguida de una suma).

Figura 4.

Procesador sistólico para redes neuronales



Fuente: Extraída de https://arxiv.org/html/2402.18595v2

Existen muchas variantes de estos procesadores sistólicos. En inteligencia artificial se utilizan fundamentalmente para realizar productos de matrices (por ejemplo, la matriz de pesos de las neuronas de una capa por la matriz de activaciones, que son las salidas de la capa anterior para las diferentes muestras de un lote), motivo por el cual cada unidad aritmética ejecuta una multiplicación seguida de una suma. Por otra parte, la figura muestra una variante en la cual se precarga la matriz de pesos en registros internos de las unidades aritméticas, y posteriormente se introduce ciclo a ciclo la matriz de activaciones por la izquierda y el resultado del producto va saliendo, también ciclo a ciclo, por la parte inferior.

Las elevadas prestaciones y eficiencia energética se consiguen porque: 1) cada dato leído de memoria se procesa muchas veces sin tener que traerlo de nuevo de memoria; y 2) cada dato solamente tiene que viajar de una unidad a la vecina para poder realizar las siguientes operaciones con dicho dato.

Además del uso de procesadores sistólicos, se utilizan otras técnicas adicionales para aumentar la velocidad de procesamiento y reducir el consumo de energía aún más. Por ejemplo, se utilizan operaciones aritméticas de muy baja precisión, llegando a representar un número en coma flotante con 16 bits (FP16) e incluso con solo 8 bits (FP8). Los aceleradores actuales incorporan soporte hardware para estos formatos.

Otro componente en el que se han desarrollado importantes mejoras en la tecnología y la arquitectura es la memoria. Se han desarrollado técnicas de fabricación de chips con integración 3D. Consiste en apilar chips dentro de un mismo encapsulado, aumentando así el número de transistores que se pueden usar. Se aplica a las memorias, ya que éstas disipan poco calor, y permite poner los chips más cerca unos de otros y reducir así mucho el movimiento de datos. En la actualidad, se están integrando en un mismo encapsulado un acelerador tipo GPU junto con una pila (3D) de chips de memoria, consiguiendo reducir aún más la distancia recorrida por los datos y aumentar notablemente el ancho de banda de memoria.

2.3. Margen de mejora de la tecnología

Tras repasar las impresionantes mejoras desarrolladas en la tecnología y en la arquitectura de los chips cabe plantearse si se puede hacer algo más. La respuesta es afirmativa, pero el margen de maniobra es ya muy escaso. La escala de integración de los chips está tocando el techo. Además, una mayor integración ya no reduce el consumo de energía sino que lo aumenta (por ejemplo, debido a las corrientes de fugas como consecuencia de que las capas aislantes de los transistores son muy finas). Por otra parte, los aceleradores están muy optimizados para minimizar el movimiento de datos y reutilizar datos al máximo. Finalmente, para fabricar transistores aún más pequeños, solamente queda el carbono por encima del silicio en la tabla periódica. Investigadores del MIT (Massachusetts Institute of Technology) han demostrado que los transistores de efecto de campo (FET) fabricados con nanotubos de carbono son más eficientes energéticamente que los transistores FET de silicio y que pueden ser fabricados en grandes cantidades en las plantas de fabricación de chips de silicio, usando obleas de tamaño estándar (Becky Ham, 2020). Pero de momento, aún no se están comercializando.

3. OPORTUNIDAD DE AHORRO DE ENERGÍA

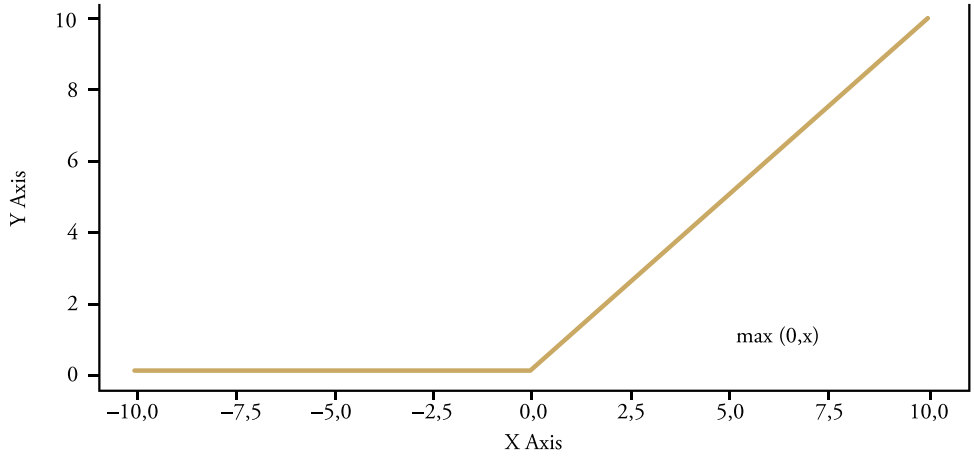
Una línea de mejora poco explorada consiste en centrarse en aplicaciones que requieren un reentrenamiento frecuente y diseñar métodos de reentrenamiento incremental que no sufran olvidos catastróficos. En aplicaciones que requieran un reentrenamiento frecuente,

este enfoque tendría un potencial de ahorro superior a 1.000 veces. Lamentablemente, el reentrenamiento incremental no funciona en redes neuronales debido al olvido catastrófico. Esto se debe a que las muestras que no se procesan durante la optimización de la función de coste no contribuyen al entrenamiento de los pesos de la red neuronal.

Para buscar una solución a este problema, consideremos una red neuronal cuyas neuronas implementan la función de activación ReLU, representada en la figura 5. Para valores de entrada negativos o nulos la función ReLU tiene una salida nula y para valores positivos se comporta como la función identidad. Cuando la salida de la función ReLU es nula, la neurona correspondiente no aporta nada a ninguna salida de la red neuronal y diremos que dicha neurona está inactiva. En cambio, cuando la ReLU se comporta como la función identidad hace que la neurona correspondiente tenga un comportamiento lineal. Recordemos que, en tal caso, la neurona simplemente calcula la suma ponderada de las entradas más el sesgo. En ese caso, diremos que la neurona está activa.

Figura 5.

Función de activación Rectied Linear Unit (ReLU)



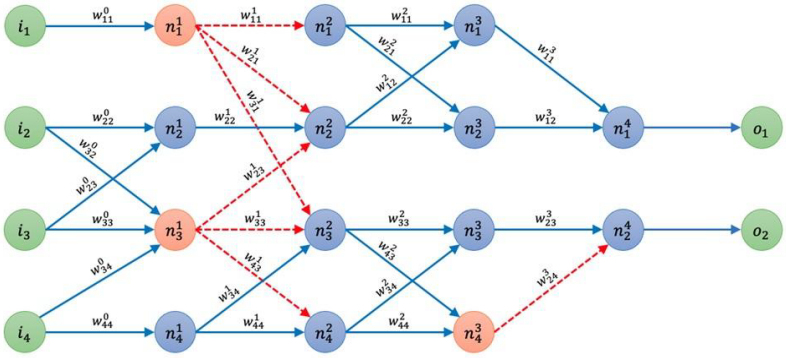
Fuente: Elaboración propia.

Así pues, cuando aplicamos una muestra a las entradas de una red neuronal, ciertas neuronas se activarán y otras quedarán inactivas. La figura 6 muestra esta situación para una pequeña red de ejemplo, en la que las neuronas activas se representan en color azul y las inactivas en rojo. Si queremos calcular el valor de una salida de la red neuronal (por ejemplo, o_1) para la muestra aplicada, hay que tener en cuenta que es la salida de la neurona n_1^4, que está activa, por lo que $o_1 = w_{11}^3 a_1^3 + w_{12}^3 a_2^3 + b_1^4$, donde a_1^3 y a_2^3 son las salidas de las neuronas n_1^3 y n_2^3, respectivamente, y b_1^4 es el sesgo de la neurona n_1^4. Si ahora sustituimos de forma recursiva el valor de cada salida de una neurona activa por su expresión matemática, obtenemos la expresión mostrada en la figura 6. El valor de la salida o_2 se calcularía de forma análoga.

Como puede observarse en la figura 6, cuando hacemos inferencia para una muestra dada, es decir, cuando aplicamos dicha muestra a la red, cada salida puede expresarse como una combinación lineal de las entradas más varios términos independientes. Por ejemplo, el rectángulo azul muestra los términos que relacionan la entrada i_2 con la salida o_1 y el rectángulo verde muestra los términos independientes aportados por la neurona n_1^2. Como detalle adicional, podemos observar que los coeficientes de cada término de la expresión para una salida de la red se pueden calcular como producto de los pesos a lo largo de la ruta desde la entrada hasta la salida correspondiente, o como producto de los pesos a lo largo de la ruta desde una neurona hasta la salida correspondiente (multiplicado por el sesgo de dicha neurona) en el caso de los términos independientes. En resumen, el valor de una salida de la red se puede calcular como la suma de las aportaciones de las diferentes rutas activas para la muestra dada, donde una ruta activa es aquella que tiene todas sus neuronas activas desde el punto de inicio hasta la salida correspondiente. El punto de inicio de una ruta puede ser una entrada o una neurona de una capa oculta o de salida.

Figura 6.

Cálculo del valor de una salida de una red neuronal con ReLU para una muestra dada



$$
\begin{aligned}
o_1 \;=\;& b_1^4 \\
&+\; w_{11}^3 b_1^3 + \boxed{w_{11}^3 w_{11}^2 b_1^2} + w_{11}^3 w_{12}^2 b_2^2 + w_{11}^3 w_{12}^2 w_{22}^1 b_2^1 + \boxed{w_{11}^3 w_{12}^2 w_{22}^1 w_{22}^0 i_2} + w_{11}^3 w_{12}^2 w_{22}^1 w_{23}^0 i_3 \\
&+\; w_{12}^3 b_2^3 + \boxed{w_{12}^3 w_{21}^2 b_1^2} + w_{12}^3 w_{22}^2 b_2^2 + w_{12}^3 w_{22}^2 w_{22}^1 b_2^1 + \boxed{w_{12}^3 w_{22}^2 w_{22}^1 w_{22}^0 i_2} + w_{12}^3 w_{22}^2 w_{22}^1 w_{23}^0 i_3
\end{aligned}
$$

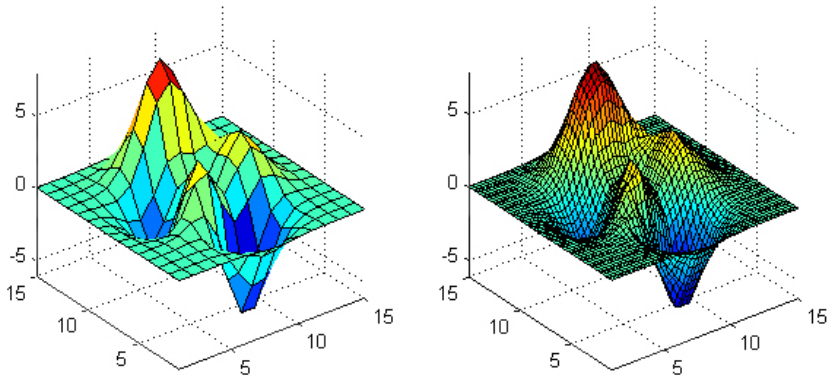
Fuente: Elaboración propia.

Es más, ante muy pequeñas variaciones en los valores de entrada respecto a los correspondientes a la muestra aplicada, el estado de activación de las neuronas de la red no cambiará, por lo que las expresiones de las salidas tampoco cambiarán y la red neuronal se comportará como un modelo lineal. En cambio, cuando las variaciones en los valores de entrada sean de mayor magnitud y cambie el estado de activación de una o varias neuronas, el resultado será otro modelo lineal diferente, análogo al mostrado pero con otros valores para los coeficientes del modelo lineal. Este comportamiento es bien conocido y se dice que una red neuronal cuyas neuronas implementan la función de activación ReLU genera salidas lineales por tra-

mos. De este modo, cualquier salida de la red puede generar funciones tan complejas como sea necesario, para ajustar así la relación entre los valores de entrada y de salida de las muestras. Si se quiere obtener mayor precisión en el ajuste, basta con hacer los tramos lineales más cortos, lo que se consigue utilizando una red neuronal más profunda (ver figura 7).

Figura 7.

Precisión de una función de salida aproximada mediante tramos lineales



Fuente: Elaboración propia.

4. NUEVO SISTEMA DE INTELIGENCIA ARTIFICIAL

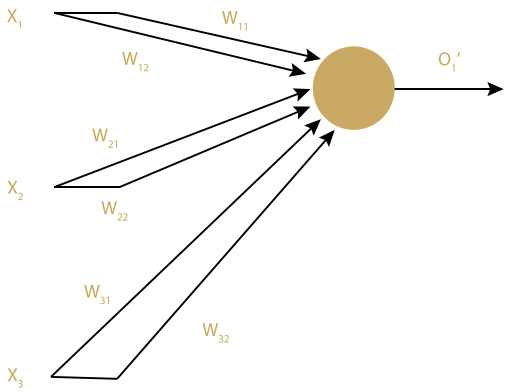
Hemos visto que una red neuronal con la función de activación ReLU selecciona un subconjunto de las neuronas de la red (las que hemos denominado neuronas activas) para cada muestra. En general, dicha selección variará de una muestra a otra. En base a estas observaciones se puede desarrollar un nuevo sistema de inteligencia artificial basado en un modelo lineal, pero que se comporta como un sistema no lineal y emula con precisión las redes neuronales basadas en la función de activación ReLU. Para conseguir dicho comportamiento no lineal, el nuevo sistema deberá seleccionar para cada muestra un subconjunto de los parámetros del modelo lineal (Duato *et al.,* 2023). Si las selecciones de parámetros son diferentes para diferentes muestras, dicho sistema también generará salidas lineales por tramos.

Al tratarse de un modelo lineal, dicho sistema puede representarse mediante una red neuronal de una sola capa, con una sola neurona por cada salida del sistema, y cuyas neuronas no tienen función de activación para que tengan un comportamiento lineal. Sin embargo, este diseño tan sencillo no permitiría suficiente flexibilidad como para poder aproximar el comportamiento de una red neuronal profunda de forma precisa. Para dotar a este sistema de mayor flexibilidad a la hora de elegir subconjuntos del mismo para cada muestra, basta con que cada neurona tenga varios pesos asociados a cada entrada, en lugar de un

solo peso, como puede verse en la figura 8. Por simplicidad, en dicha figura se han representado únicamente dos pesos asociados a cada entrada de la neurona. Cada uno de esos pesos se puede seleccionar o no, de forma individual, para cada muestra. Esto da lugar a cuatro combinaciones por entrada. Por ejemplo, para la entrada X_1, podemos no seleccionar ningún peso (en cuyo caso esa entrada no influirá en la salida), seleccionar w_{11}, seleccionar w_{12} o seleccionar ambos pesos (en cuyo caso su aportación se suma). Nótese que la selección de pesos para cada entrada es independiente de la selección para el resto de entradas. Así pues, en la figura 8, a pesar de su simplicidad, se pueden generar 4×4×4=64 combinaciones diferentes de selección de pesos. El número de combinaciones también puede calcularse como 2^p, siendo p el número de pesos que pueden ser seleccionados individualmente (6 en este ejemplo).

Figura 8.

Ejemplo de neurona con 3 entradas y 2 pesos por entrada



Fuente: Elaboración propia.

Por ejemplo, si se usan 10 pesos por entrada, se puede elegir entre 1024 combinaciones diferentes para distintas muestras, y dichas combinaciones son independientes de las que se pueden elegir para el resto de entradas. De este modo se puede conseguir una gran flexibilidad y obtener un comportamiento análogo al de una red neuronal con la función de activación ReLU.

Los principales beneficios del enfoque propuesto son:

- Excelente interpretabilidad, ya que directamente se genera un modelo lineal para cada muestra.
- Se pueden fusionar fácilmente varias copias de un modelo lineal determinado, sin más que calcular la media ponderada parámetro a parámetro. Como factor de ponderación de los parámetros de cada modelo se puede usar el número de muestras utilizadas para entrenar dicho modelo.

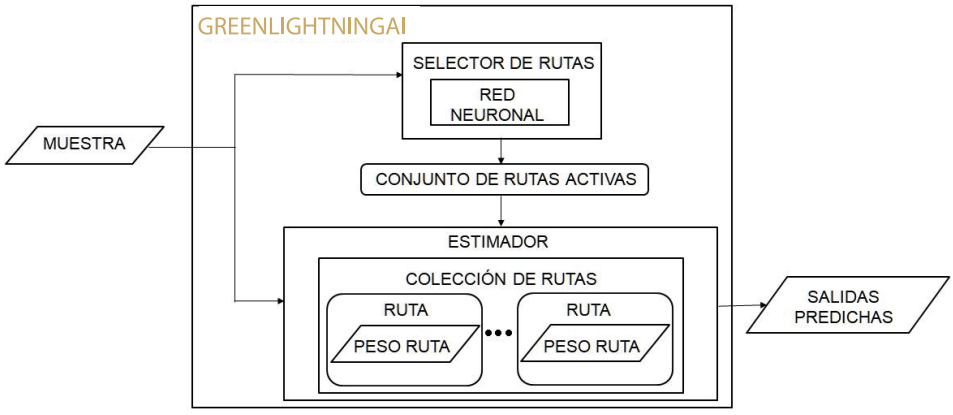
- La fusión de modelos permite realizar reentrenamiento incremental, entrenamiento federado, reentrenamiento incremental federado, etc.

Sin embargo, queda un importante problema por resolver, que es la selección de un subconjunto del modelo lineal para cada muestra. Dicha selección debe cumplir una importante propiedad: Dado un par de muestras, cuanto más similares sean las muestras más similares deben ser los correspondientes subconjuntos de parámetros seleccionados.

Este problema se puede resolver de diversas formas. En una primera versión se ha resuelto utilizando una pequeña red neuronal, con función de activación ReLU, para realizar la selección (Duato *et al.,* 2023)(ver figura 9). Seguidamente se describe la integración de dicha red neuronal con el modelo lineal. En cualquier red neuronal se pueden identificar una serie de rutas posibles desde cualquier entrada a cualquier salida. Es lo que denominamos colección de rutas, e incluye todas las rutas posibles definidas por los patrones de conexión entre las capas de la red. El modelo lineal se dimensiona a partir de esta colección de rutas, de modo que se define un peso entrenable (y solamente uno) por cada ruta posible en la red neuronal que se va a usar para seleccionar subconjuntos del modelo lineal.

Figura 9.

Diagrama de bloques del nuevo sistema de inteligencia artificial



Fuente: Elaboración propia.

Cuando una red neuronal está basada en la función de activación ReLU, para una muestra dada se activan algunas de esas rutas y otras no, como se ha visto en el ejemplo de la figura 6. El conjunto de rutas activas para una muestra dada es lo que hace falta calcular en el sistema propuesto, enviando dicha información al modelo lineal, denominado estimador, el cual seleccionará exclusivamente el subconjunto de pesos del modelo que corresponden a las rutas activas. Seguidamente, dicho subconjunto se puede utilizar para

la operación deseada con dicha muestra, ya sea realizar inferencia o procesar dicha muestra como parte de un proceso iterativo de entrenamiento. El diagrama de bloques del nuevo sistema de inteligencia artificial se muestra en la figura 9.

La red neuronal utilizada para calcular el conjunto de rutas activas para cada muestra, denominada selector de rutas, se puede inicializar entrenándola previamente con cualquier método de entrenamiento. De hecho, es posible entrenar el selector de rutas con una pequeña fracción de las muestras de entrenamiento y con pocas iteraciones. El motivo es que no necesita generar una salida precisa. Basta con que discrimine adecuadamente entre diferentes muestras. Es más, en general no hace falta actualizar el selector de rutas cuando se reentrena el sistema propuesto, por el mismo motivo. Para reentrenar el sistema, basta con reentrenar los pesos del estimador, para lo cual puede usarse cualquier método de entrenamiento habitualmente utilizado para entrenar redes neuronales (ya que en realidad es equivalente a una red neuronal de una capa).

5. VERIFICACIÓN EXPERIMENTAL

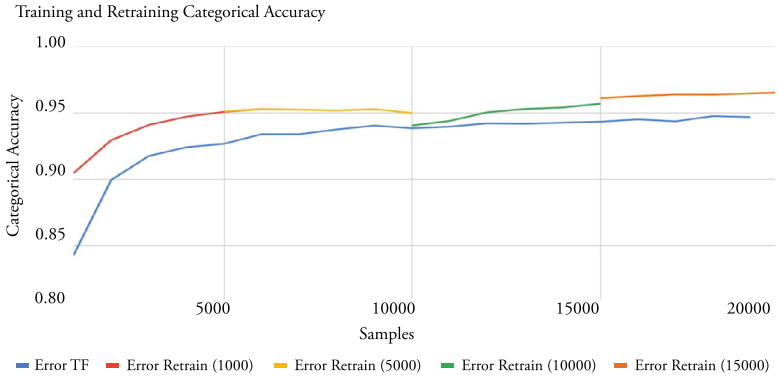
Con objeto de verificar que el selector de rutas no requiere ser reentrenado cuando se reentrenan los pesos del modelo lineal (es decir, del estimador), se ha llevado a cabo un sencillo experimento, comparando una red neuronal de dos capas con 300 y 10 neuronas, respectivamente, con el nuevo sistema de inteligencia artificial configurado de modo que tenga aproximadamente el mismo número de pesos entrenables que la red neuronal. Como conjunto de datos de entrenamiento y validación se ha utilizado el popular MNIST (Lecun *et al.,* 2010), que es un conjunto de imágenes de dígitos manuscritos.

Tanto la red neuronal como el selector de rutas y el estimador del nuevo sistema se han entrenado utilizando el método estocástico de descenso del gradiente. La figura 10 muestra la precisión obtenida entrenando con diferentes números de muestras. La curva azul corresponde a la red neuronal convencional. La curva roja corresponde al nuevo sistema, en el que el selector de rutas ha sido entrenado inicialmente con 1.000 muestras y no se ha vuelto a reentrenar, entrenando únicamente el estimador para los diferentes puntos de la curva. Las curvas amarilla, verde y naranja también muestran resultados en los que se entrena solamente el estimador con tamaños crecientes del conjunto de muestras, habiendo inicializado previamente el selector de rutas con 5.000, 10.000 y 15.000 muestras, respectivamente.

Como puede verse en la figura 10, el nuevo sistema consigue una precisión comparable a la de una red neuronal convencional, a pesar de que el selector de rutas se haya inicializado con un número menor de muestras que las utilizadas para entrenar el estimador. Es más, la reinicialización del selector de rutas con un mayor número de muestras (5.000, 10.000 y 15.000 muestras) no parece aportar ningún beneficio, lo que prueba que basta con entrenar inicialmente dicho selector de rutas con un porcentaje reducido de las muestras (salvo en aquellas aplicaciones en las que las muestras puedan variar mucho a lo largo del tiempo).

Figura 10.

Evaluación de la necesidad de reentrenar el selector de rutas



Fuente: Elaboración propia.

6. EVALUACIÓN DE PRESTACIONES

Con objeto de evaluar los beneficios aportados por el nuevo sistema de inteligencia artificial, se han realizado numerosas comparaciones con redes neuronales convencionales. Las figuras 11 y 12 muestran el tiempo de ejecución relativo y la precisión obtenida en validación cuando comparamos una red neuronal profunda de 152 capas (ResNet152) con un diseño basado en el nuevo sistema de inteligencia artificial con un número similar de parámetros entrenables. El conjunto de datos es ImageNet (Stanford Vision Lab, 2021), un conjunto de más de 1.200.000 imágenes de 1.000 clases de objetos. Nótese que el consumo de energía es aproximadamente proporcional al tiempo de ejecución.

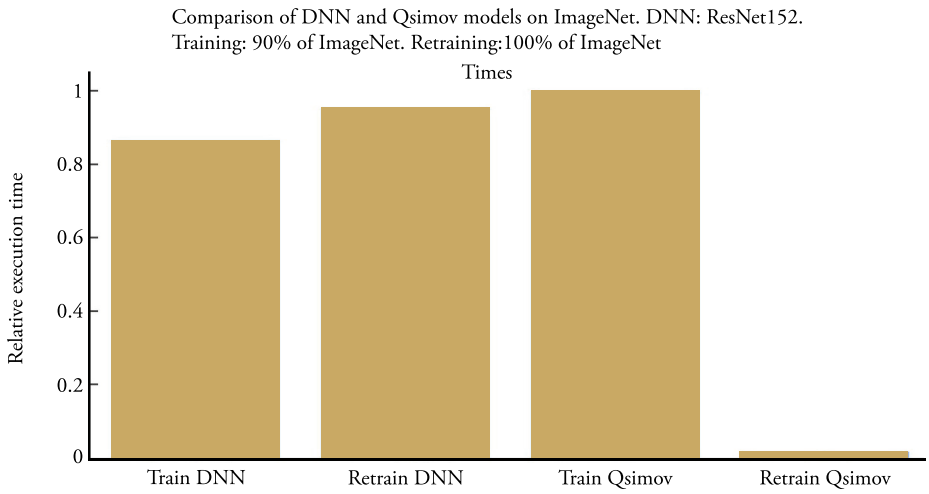
Para evaluar los beneficios del reentrenamiento incremental, que solamente es posible con el nuevo sistema, se ha realizado primero un entrenamiento con el 90 % de ImageNet, seguido de un reentrenamiento con el resto de imágenes. Dicho reentrenamiento solamente ha tenido que procesar el 10 % restante de ImageNet en el caso del nuevo sistema, manteniendo inalterado el selector de rutas y actualizando únicamente los pesos del estimador. Sin embargo, en el caso de la red ResNet152 ha sido necesario procesar el 100 % de ImageNet debido al olvido catastrófico.

Las cuatro barras de cada figura corresponden al entrenamiento de la red neuronal (DNN), el reentrenamiento de dicha red, el entrenamiento del nuevo sistema (Qsimov) y el reentrenamiento incremental del mismo. Como puede observarse en la figura 11, la capacidad de reentrenamiento incremental supone una enorme reducción en el tiempo de procesamiento y en la correspondiente energía consumida. La reducción en el tiempo de ejecución no solo se debe a que el nuevo sistema solamente necesita procesar las muestras nuevas. El diseño del estimador como una red de una sola capa sin función de activación hace que la convergencia del proceso iterativo de entrenamiento sea mucho más rápida que en una red

neuronal profunda. Como puede observarse en la figura 12, a pesar de procesar solamente las muestras nuevas durante el reentrenamiento y actualizar únicamente los pesos del estimador, la precisión conseguida por el nuevo sistema de inteligencia artificial es prácticamente idéntica a la obtenida por la red neuronal.

Figura 11.

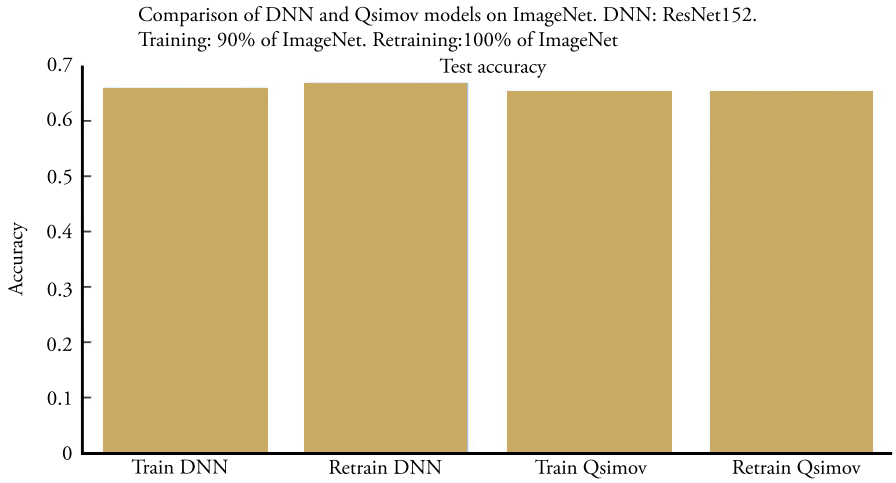
Evaluación comparativa de los tiempos de entrenamiento y reentrenamiento



Fuente: Elaboración propia.

Figura 12.

Evaluación comparativa de las precisiones alcanzadas



Fuente: Elaboración propia.

7. CONCLUSIONES

En este artículo se ha presentado una revisión sucinta de los grandes avances en los métodos matemáticos y en los algoritmos asociados para reducir el número de operaciones aritméticas necesarias para entrenar redes neuronales, con la consiguiente reducción en los tiempos de entrenamiento y en el coste y consumo de energía asociados. También se han revisado las aportaciones realizadas por los avances en la tecnología y en la arquitectura de los computadores, mostrando que su impacto en la reducción del consumo de energía es muchos órdenes de magnitud superior a la reducción lograda por los métodos matemáticos.

Dado que los márgenes de mejora son cada vez más pequeños, se ha propuesto y evaluado un nuevo sistema de inteligencia artificial orientado específicamente a reducir el coste y consumo de energía en aquellas aplicaciones que requieran un reentrenamiento frecuente. Para ello, el nuevo sistema se ha diseñado a partir de un modelo lineal, de modo que permite combinar rápidamente diferentes modelos con la misma arquitectura. Los resultados de evaluación muestran la enorme reducción en los tiempos de ejecución que se consigue al reentrenar con el nuevo sistema, sin que ello afecte a la precisión del modelo.

Referencias

Amodei, D., y Hernandez, D. (2018). Ai and compute. OpenAI. https://openai.com/blog/ai-and-compute

Becky Ham (MIT News correspondent). (2020). MIT News. https://news.mit.edu/2020/carbon-nanotube-transistors-factory-0601

Duato, J., Mestre, J. I., Dolz, M. F., y Quintana-Ortí, E. S. (2023). GreenLightningAI: An Efficient AI System with Decoupled Structural and Quantitative Knowledge. arXiv. https://arxiv.org/abs/2312.09971

Green500, June 2024 List. (2024). Top500. https://top500.org/lists/green500/2024/06/

LeCun, Y., Cortes, C., y Burges, C. J. C. (2010). THE MNIST DATABASE of handwritten digits., https://www.kaggle.com/datasets/hojjatk/mnist-dataset

Nielsen, M. (2019). Neural Networks and Deep Learning. http://neuralnetworksanddeeplearning.com/

Nvidia GH200 Grace Hopper Superchip. (2024). Nvidia. https://www.nvidia.com/es-es/data-center/grace-hopper-superchip/

Stanford Vision Lab, Stanford University, Princeton University. (2021). ImageNet. https://www.image-net.org/

The Brussels Times Newsroom. (2024). ChatGPT consumes 25 times more energy than Google. The Brussels Times. https://www.brusselstimes.com/1042696/chatgpt-consumes-25-times-more-energy-than-google