Del paralelismo masivo a la eficiencia neuronal: El cambio de paradigma en el hardware de IA

Eiroa, Facundo Nicolas

Ferreyro, Nazareno Gastón

Universidad del CEMA

Lenguajes Formales y Teoría de la Computación

Prof. Mario Samuel Moreno

Julio 2026

Referencias

Introducción 4

Objetivos 5

Hipótesis 5

Marco teórico 6

La CPU (Central Processing Unit) y el cuello de botella secuencial 6

La GPU (Graphics Processing Unit) y el paralelismo masivo 6

Performance de distintos modelos de IA Local en GPUs discretas 7

Los CUDA (Compute Unified Device Architecture) cores y su utilidad en los LLMs 8

Ventajas y desventajas frente a la CPU 9

Ventajas y desventajas frente a las GPU genéricas 11

La NPU (Neural Processing Unit) y un procesamiento eficiente 12

TPU (Tensor Processing Unit) 14

Ejemplos prácticos 15

Generación de frames 15

Fondo borroso 16

Conclusión 17

Anexo 18

CPU 18

TOPS 18

HBM 19

Desenfoque gaussiano.....20

Bibliografía .....**¡Error! Marcador no definido.**

Introducción

La inteligencia artificial (IA) ha experimentado recientemente fuertes avances, en gran medida impulsados por el desarrollo de cada vez más potentes *Large Language Models* (LLM's) y redes neuronales profundas. Tradicionalmente, la inmensa carga computacional requerida para el entrenamiento y la inferencia de estos modelos está siendo delegada a infraestructuras de servidores masivos, que poseen distintos componentes de hardware como los 'Aceleradores' de IA.

Sin embargo, a medida que la IA se integra de manera ubicua en la vida cotidiana de las personas, la dependencia exclusiva de servidores en la nube para absolutamente todas las tareas presenta desafíos insostenibles a largo plazo en términos de latencia, privacidad de datos, costos de servidores y consumo de ancho de banda.

Esto plantea un problema que puede ser solucionado, derivando ciertas tareas al hardware propio del cliente. El hardware de consumo debe poder recibir gran parte de estas actividades de inferencia así logrando solventar gran parte de estos inconvenientes. Para esto, se necesita una gran evolución para soportar estas exigencias, superando las limitaciones del procesamiento secuencial tradicional mayormente incompatible con las nuevas tecnologías, y abrazando el procesamiento en paralelo masivo.

Objetivos

El objetivo general que planteamos en esta investigación es analizar la evolución y adaptación de la arquitectura de hardware en niveles de servidores y grandes centros de datos, así como también a nivel de clientes y dispositivos personales. Los objetivos específicos que queremos abarcar son:

- Evaluar las capacidades y limitaciones de los componentes de hardware tradicionales (CPU y GPU) en tareas de inferencia local
- Explicar el funcionamiento de los *Tensor Processing Units* (TPU)
- Examinar el rol urgente que debería cumplir las Unidades de Procesamiento Neuronal (NPU) en el cliente.
- Explorar aplicaciones prácticas de IA de ejecución local que trascienden los casos de uso convencionales (como la generación de texto o código)

Hipótesis

Para viabilizar la integración masiva de IA en el uso cotidiano, es imperativa la adopción de hardware local especializado en el cliente. La estandarización de módulos de procesamiento locales permitirá delegar las tareas de inferencia eficientemente, reduciendo la dependencia de unidades gráficas (GPU) discretas de alto coste y mitigando los cuellos de botella de la infraestructura en la nube.

Marco teórico

La CPU (Central Processing Unit) y el cuello de botella secuencial

La arquitectura tradicional de la CPU esta optimizada para la baja latencia y la ejecución de instrucciones complejas de propósito general. Los procesadores multinúcleo (es decir, los que contienen varios procesadores independientes en un mismo encapsulado) poseen bastante potencia, pero no son lo suficientemente paralelos (no tienen una cantidad masiva de aquellos núcleos). Cuando tienen que enfrentarse a tareas de inferencia de inteligencia artificial, que requieren el cálculo simultáneo de millones de operaciones de álgebra lineal (multiplicación de matrices y tensores), se encuentran con un cuello de botella masivo. Esto se debe a que carece de la arquitectura paralela extrema que poseen otros componentes que procesan estas tareas, y la descarta como una solución eficiente.

La GPU (Graphics Processing Unit) y el paralelismo masivo

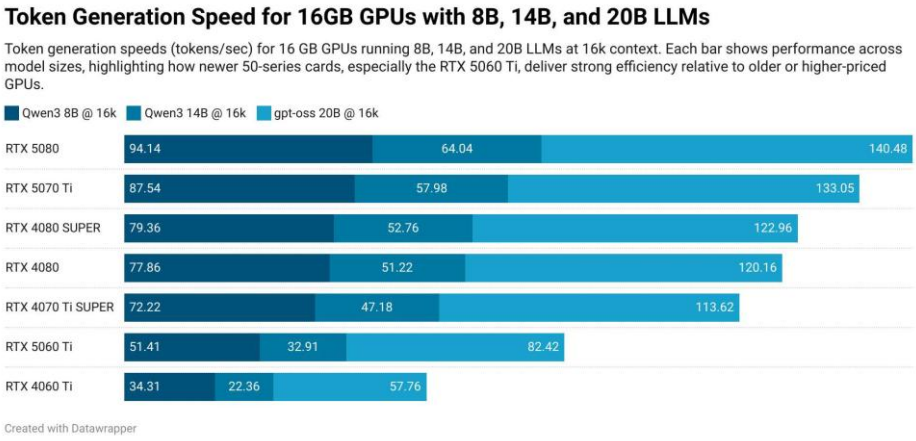
La GPU representa la antítesis arquitectónica de la CPU en términos de cálculo intensivo. Fue diseñada originalmente para el renderizado de gráficos 3D. (Como curiosidad, en el siglo pasado se les llamaban “Aceleradoras 3D” (Sametband, 1999), y solo estaban pensadas para jugar videojuegos generalmente). Para poder calcular millones de píxeles en simultáneo, la GPU emplea una arquitectura masivamente paralela basada en modelos como SIMT (Single Instruction, Multiple Threads). Esto significa que en lugar de destinar transistores a la memoria caché y al control de flujo, lo que es llamado Datapath en un CPU, (Patterson)), los dedicamos en potenciar solamente las ALU (Unidades Aritmético Lógicas), albergando cientos o miles de núcleos más simples diseñados para ejecutar la misma instrucción sobre múltiples conjuntos de datos de forma concurrente.

Esta topología resulta ideal para el aprendizaje profundo, donde la carga de trabajo subyacente consiste en multiplicaciones de masivas de matrices y objetos multidimensionales. En los últimos años, las arquitecturas de GPU han dado un paso más allá al integrar hardware especifico en el silicio de propósito específico como los núcleos Tensor de Nvidia, (NVIDIA, s.f.), que están diseñados exclusivamente para acelerar operaciones matemáticas de menor precisión que las redes neuronales utilizan para la inferencia sin perder capacidad predictiva. (Kirk, 2016)

No obstante, depender de GPUs discretas para la IA local presenta desventajas estructurales para la computación hogareña. Demandan un alto consumo energético (TDP muy alto), requieren sistemas de refrigeración voluminosos, y suponen un gran costo económico. Estas barreras físicas y financieras buscan una solución más factible para los clientes.

Performance de distintos modelos de IA Local en GPUs discretas

Figura 1



Nota: Adaptado de Hardware Corner

Para agregar, en esta imagen se observa la performance de varios modelos de IA open-source local (Qwen 8 y 14 B, y gpt-oss 20B), mostrando la cantidad de tokens por segundo emitidas por cada uno en distintas GPU de gama media, orientadas para clientes entusiastas (serie RTX) de NVIDIA. Esto prueba que las GPU realmente son capaces de ejecutar modelos extensos de lenguaje (LLMs).

Los CUDA (Compute Unified Device Architecture) cores y su utilidad en los LLMs

Un CUDA core es la unidad de ejecución elemental dentro de las unidades de procesamiento gráfico (GPU) de NVIDIA, encargada de realizar operaciones aritméticas escalares sobre datos en punto flotante y en enteros. A diferencia de los núcleos de una unidad central de procesamiento (CPU), un CUDA core es deliberadamente simple: prescinde de lógica compleja de control y de grandes memorias caché privadas, de modo que un solo procesador gráfico puede integrar miles de ellos operando de manera simultánea. Este diseño persigue maximizar el rendimiento agregado (throughput) en cargas de trabajo intensamente paralelas (Owens et al., 2008).

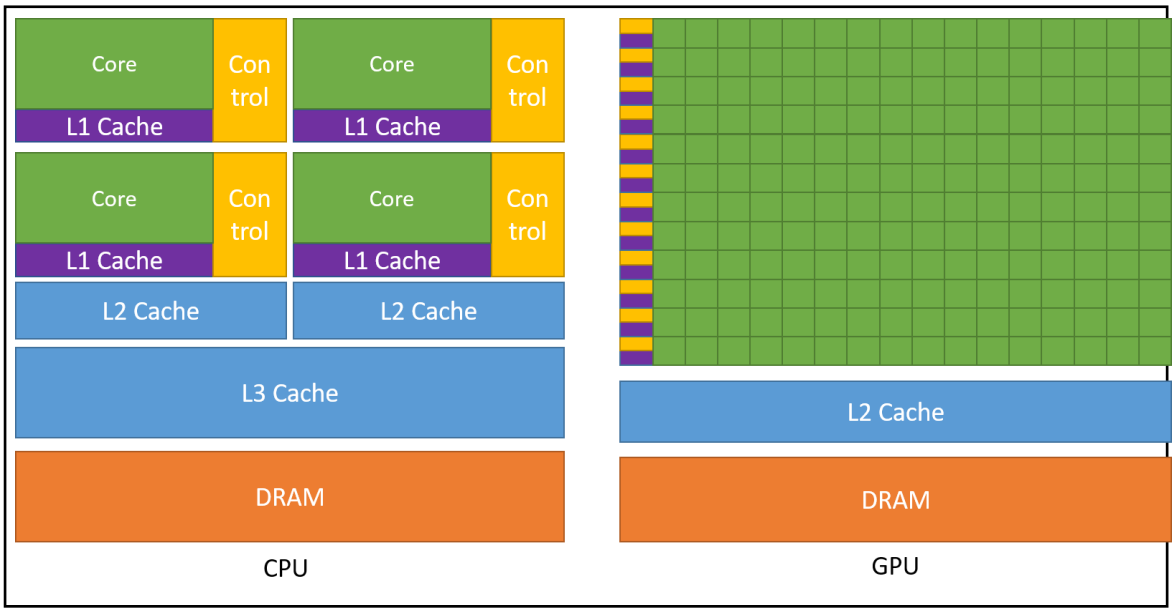
La ejecución sobre estos núcleos se rige por el modelo SIMT (Single Instruction, Multiple Threads), una variante del paralelismo de datos en la que un mismo flujo de instrucciones se aplica de forma concurrente sobre numerosos hilos, cada uno operando sobre datos distintos (Nickolls et al., 2008). Los hilos se agrupan en conjuntos de 32 denominados *warps(*grupo de 32 hilos que la GPU planifica y ejecuta en conjunto, como una sola unidad*)*, que constituyen la unidad de planificación efectiva: el planificador emite una instrucción y todos los hilos del *warp* la ejecutan en paralelo sobre los CUDA cores disponibles. Este esquema permite ocultar la latencia de acceso a memoria mediante la conmutación rápida entre numerosos *warps* en vuelo, en lugar de recurrir a grandes jerarquías de caché (Nickolls et al., 2008).

Los CUDA cores no actúan de forma aislada, sino que se organizan jerárquicamente dentro del Streaming Multiprocessor (SM), el bloque de cómputo fundamental de la arquitectura. En la arquitectura Turing, cada SM contiene 64 núcleos FP32 (operaciones con números decimales (coma flotante) de precisión simple) y 64 núcleos INT32 (operaciones con números enteros de 32 bits.), además de 8 Tensor Cores y 1 RT Core(*Ray Tracing Core*) las cuales son unidades que son especializadas en ray tracing (trazado de rayos), internamente, el SM se subdivide en cuatro bloques de procesamiento, cada uno con 16 núcleos FP32, 16 núcleos INT32, dos Tensor Cores, un planificador de *warps* y una unidad de despacho (NVIDIA, 2018). Cada SM dispone además de 256 KB de archivo de registros y de 96 KB de memoria configurable entre caché L1 y memoria compartida (NVIDIA, 2018). Una particularidad de Turing es la incorporación de una ruta de datos entera independiente, capaz de ejecutar instrucciones de enteros de forma concurrente con las de punto flotante; según NVIDIA, esto permite procesar alrededor de 36 operaciones enteras por cada 100 operaciones de punto flotante sin bloquear la tubería de ejecución (NVIDIA, 2018).

Ventajas y desventajas frente a la CPU

La diferencia esencial entre una GPU basada en CUDA cores y una CPU radica en el objetivo de diseño: la CPU optimiza la latencia de hilos individuales mediante núcleos complejos, predicción de saltos y amplias cachés, mientras que la GPU optimiza el rendimiento agregado mediante la replicación masiva de núcleos sencillos (Owens et al., 2008). De allí derivan sus ventajas comparativas.

Figura 2



Nota: Adaptado de NVIDIA, Turing Architecture In-Depth

En primer lugar, el paralelismo masivo: una GPU ejecuta miles de hilos de forma concurrente, lo que la hace muy superior a la CPU en problemas de paralelismo de datos, como el álgebra lineal densa, el procesamiento de imágenes o el entrenamiento de redes neuronales (Nickolls et al., 2008). En segundo lugar, la eficiencia energética por operación: al destinar la mayor parte del área del chip a unidades aritméticas en lugar de a lógica de control, la GPU obtiene un mayor rendimiento por vatio en cargas adecuadas; en Turing, por ejemplo, la memoria GDDR6 opera a 14 Gbps con una mejora del 20 % en eficiencia energética respecto de la generación anterior (NVIDIA, 2018). En tercer lugar, el modelo SIMT permite escalar el rendimiento simplemente añadiendo más SM, sin reescribir el código (Nickolls et al., 2008).

Estas ventajas tienen como contrapartida limitaciones claras. La latencia de un hilo individual en la GPU es alta: el rendimiento se consigue por volumen, no por la rapidez de una tarea aislada, por lo que aplicaciones de control de flujo serial o con baja paralelizabilidad rinden mejor en la CPU (Owens et al., 2008). El modelo SIMT penaliza, además, la ramificación intensiva: cuando los hilos de un mismo *warp* siguen caminos distintos en una bifurcación condicional, se produce la divergencia de *warps*, que obliga a ejecutar serialmente los caminos divergentes y degrada el rendimiento (Nickolls et al., 2008). Por último, los datos deben transferirse entre la memoria del *host* (CPU) y la del dispositivo (GPU) a través del bus de comunicacion PCI Express (Peripheral Component Interconnect Express), un costo de transferencia que puede anular las ganancias del cómputo paralelo si el volumen de datos movido es elevado en relación con el cálculo realizado (Owens et al., 2008).

Ventajas y desventajas frente a las GPU genéricas

Más allá de la comparación con la CPU, conviene distinguir entre una GPU genérica —entendida como hardware de cómputo paralelo sin un ecosistema de software maduro— y una GPU de NVIDIA integrada al ecosistema CUDA. La principal ventaja de esta última no reside únicamente en sus CUDA cores, sino en la combinación de unidades de cómputo especializadas y de librerías de alto nivel.

En el plano del hardware, las arquitecturas recientes incorporan unidades de función fija junto a los CUDA cores. Los Tensor Cores —ocho por SM en Turing— aceleran las multiplicaciones de matrices propias del aprendizaje profundo, mientras que el RT Core —uno por SM— acelera el recorrido de jerarquías de volúmenes envolventes para el trazado de rayos en tiempo real (NVIDIA, 2018). En el plano del software, el ecosistema CUDA ofrece librerías optimizadas como cuBLAS, para subrutinas de álgebra lineal, y cuDNN, con

primitivas de redes neuronales profundas (convolución, atención, normalización y *pooling*); estas librerías son aprovechadas por los principales marcos de trabajo, como PyTorch o TensorFlow (NVIDIA, s.f.). Este conjunto reduce el esfuerzo de desarrollo y permite explotar el hardware especializado sin programarlo de manera directa, ventaja de la que carece una GPU genérica sin ese ecosistema.

La contrapartida es la dependencia de un ecosistema propietario. CUDA, sus librerías y las unidades especializadas son tecnologías controladas por NVIDIA y solo funcionan sobre su hardware, lo que genera un efecto de *lock-in*: el código escrito específicamente para CUDA no es portable a GPU de otros fabricantes y requiere ser reescrito o adaptado a estándares abiertos —como OpenCL o SYCL— para ejecutarse en plataformas alternativas (Owens et al., 2008). Esta dependencia condiciona las decisiones de adquisición de hardware y limita la portabilidad de las soluciones desarrolladas sobre la plataforma.

La NPU (Neural Processing Unit) y un procesamiento eficiente

La Unidad de Procesamiento Neural surge como una respuesta arquitectónica directa a los compromisos mencionados anteriormente (problemas energéticos y económicos). Se puede definir como un microprocesador informático especializado diseñado para imitar las funciones de procesamiento de un cerebro humano, estando optimizado para las distintas tareas de IA neuronal y Machine Learning que podríamos necesitar (IBM, s.f.).

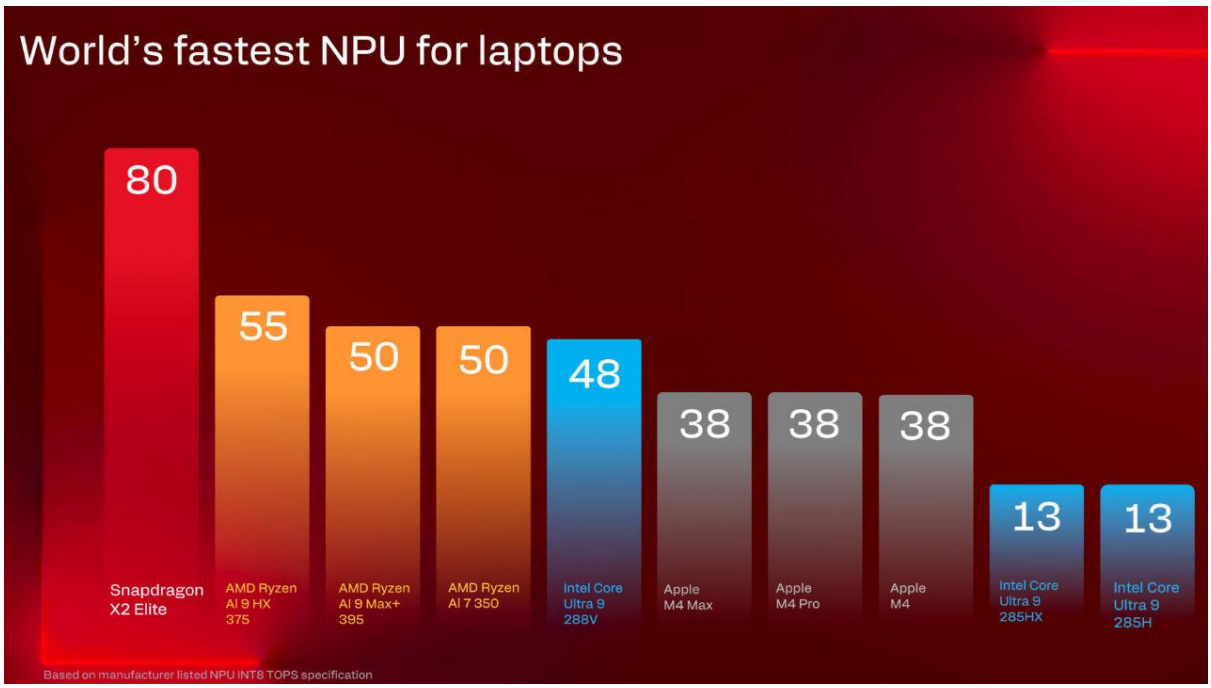
Estos dispositivos son también conocidos como “Aceleradores de IA”, y se cataloga como una arquitectura de dominio específico. Se especializan en aquel cálculo de estructuras de álgebra multidimensionales mencionadas. Estos dispositivos plantean como prioridad la eficiencia algorítmica, logrando hacer tareas de inferencia de IA en tiempo real operando a un

TDP realmente más bajo que el de una GPU. Esto permitiría a los usuarios que no poseen GPUs discretas permitirles correr cargas de IA de forma local.

La tendencia actual de los fabricantes es integrar estos bloques neuronales directamente dentro del encapsulado del microprocesador (CPU), formando lo que es llamado System On a Chip o SOC. Esta arquitectura heterogénea permite al sistema operativo delegar las tareas de inferencia pesadas y continuas a la NPU, liberando a la CPU para la lógica del sistema y evitando el encendido de la GPU dedicada (si es que la tiene), prolongando significativamente la autonomía en dispositivos portátiles y disminuyendo considerablemente la latencia. (HP, 2026).

Performance de las NPU más populares

Figura 3



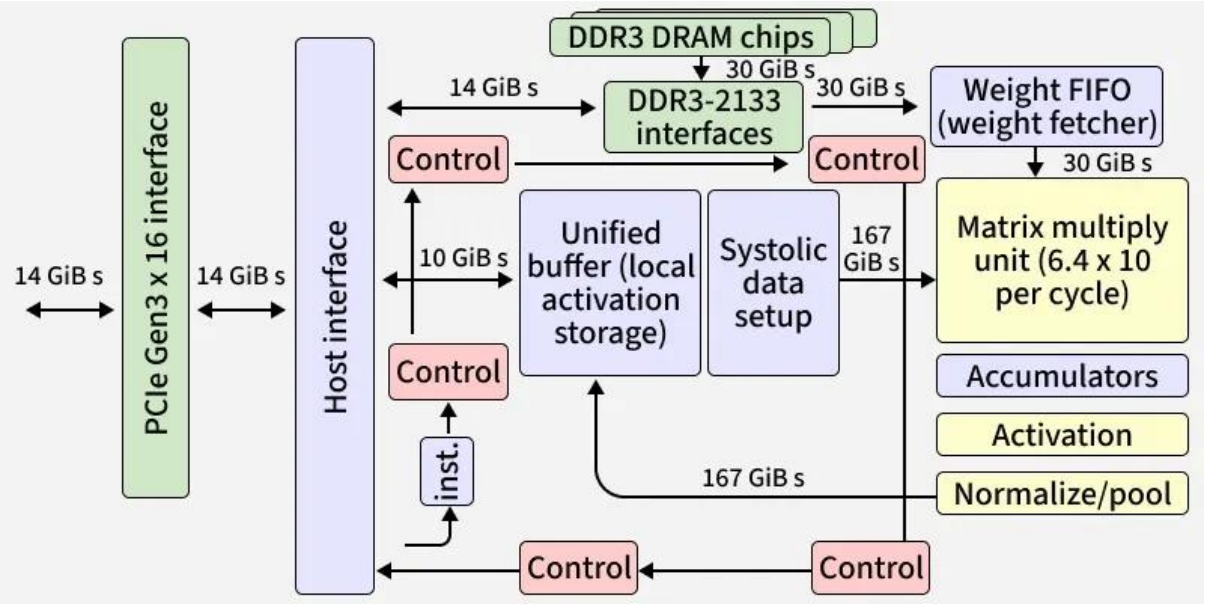
Nota: Adaptado de Qualcomm

En la imagen superior se puede observar el rendimiento de diversas NPU de computadoras portátiles. La unidad de medida son TOPS. (Véase anexo para ampliar). Para fines del 2025, se puede considerar la NPU incorporada del microprocesador Snapdragon X2 Elite la más potente.

TPU (Tensor Processing Unit)

Las TPUs son aceleradores con diseño personalizado creados específicamente para cargas de trabajo de IA, como agentes, generación de código, modelos de lenguaje grandes (LLMs), generaciones multimodales, entre otras muchas cosas. (Este concepto fue creado por Google, y es como ellos manejan la IA en sus centros de datos). (Google Cloud, s.f.)

Figura 4



Nota: TPU v1 Architecture. Adaptado de Geeks for Geeks

La Matrix Multiply Unit (MXU) es el *core* de este sistema. Allí se encuentran un conjunto de 256x256 Unidades Aritmético Lógicas (ALUs). Los TPU usan otra unidad de coma flotante, llamada bfloat16, que recorta la mantisa del número flotante. Así se usa la mitad de la memoria y se duplica el throughput.

Con respecto al tipo de memoria, se usa del tipo HBM (ver anexo), luego un buffer de gran tamaño (CMEM) y otra memoria llamada SparseCore. (Geeks For Geeks, 2025)

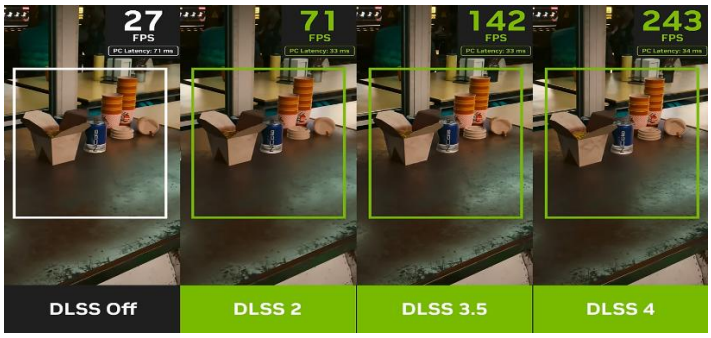
Ejemplos prácticos

Generación de frames

La generación de frames produce cuadros intermedios entre dos frames renderizados, incrementando la tasa de cuadros percibida.

NVIDIA introdujo DLSS 3 Frame Generation con la arquitectura Ada Lovelace (RTX 40), empleando un autoencoder convolucional alimentado por los vectores de movimiento del motor de juego y por un acelerador de flujo óptico (Optical Flow Accelerator) implementado en hardware dedicado; el sistema inserta un frame sintético, duplicando la tasa de cuadros (NVIDIA, 2022). Con la arquitectura Blackwell (RTX 50), DLSS 4 reemplazó el acelerador de flujo óptico por un modelo de IA que genera hasta tres frames adicionales por cuadro renderizado, alcanzando un multiplicador de hasta 4x; el nuevo modelo resulta un 40 % más rápido y consume un 30 % menos de memoria de video que su predecesor (NVIDIA, 2025a; TechPowerUp, 2025d). DLSS 4.5 extendió esta capacidad a un modo 6x con ajuste dinámico según la frecuencia del monitor (NVIDIA, 2026a; Tom's Hardware, 2026; TechSpot, 2026)

Figura 5



Nota: DLSS 4 / DLSS 3 / DLSS 2 / DLSS Off Comparison | Cyberpunk 2077. Adaptado de NVIDIA

AMD, en contraste, basa su generación de frames en flujo óptico sin redes neuronales entrenadas. FSR 3 Frame Generation utiliza los vectores de movimiento del motor de juego para interpolar un cuadro intermedio (AMD, s.f.a), mientras que AMD Fluid Motion Frames 2 (AFMF 2) opera a nivel de controlador, sin acceso a datos internos del juego, lo que amplía su compatibilidad a cualquier título, pero degrada la calidad e introduce artefactos, dado que la interfaz se interpola junto con la escena (AMD, s.f.b). Ambas soluciones se limitan a un multiplicador de 2x. Con RDNA 4, la suite FSR "Redstone" incorpora aceleración por IA sobre las nuevas unidades de cómputo en punto flotante de 8 bits (FP8), aunque su funcionamiento interno no ha sido documentado en detalle (TweakTown, 2025a).

Fondo borroso

Un ejemplo clásico para demostrar el uso de una NPU para los usuarios es el desenfoque de fondos durante videollamadas (también llamado efecto Bokeh).

Lo que ocurre aquí es que, para aplicar un fondo borroso en vivo, se debe tomar cada fotograma que es capaz de capturar la cámara como un caso especial. Un algoritmo debe detectar que píxeles corresponden a un humano (foreground), y cuales corresponden a el fondo (background). A eso, luego se le aplica un algoritmo matemático (como un desenfoque gaussiano, ver anexo).

El problema, proviene de la exigencia que le causa al CPU hacer esta tarea. Algo que parece simple, resulta muy costoso para el hardware. Se le podría delegar a la GPU, pero también generaría problemas, en especial en portátiles con iGPU[1], que sufrirían de stuttering[2] y drenado de batería. (Ignatov, y otros)

[1] iGPU significa GPU integrada. Se encuentra en el mismo encapsulado que el procesador.

[2] Se le llama stuttering a la interrupción irregular de frames que afecta severamente la fluidez.

La NPU ejecuta un modelo de visión artificial ligero para poder identificar siluetas humanas. Al estar diseñada para hacer operaciones de algebra de forma masiva, puede realizar esta tarea consumiendo muy poca energía, comparado a las otras alternativas anteriormente mencionadas.

Además, la unidad de procesamiento neuronal trabaja en conjunto con el ISP (procesador de imagen de la cámara), por lo que el desenfoque se realiza antes de que el flujo de video llegue al espacio de usuario. Como resultado de esto, el sistema operativo ya presenta la cámara virtual ya procesada. Por lo tanto, el software (como por ejemplo Zoom o Teams), no tratan de aplicar sus algoritmos ineficientes sobre la cámara, y tampoco pueden saber que esa imagen fue generada por Inteligencia Artificial. Por ejemplo, en Windows se utiliza la API “Windows Studio Effects”. (Microsoft, s.f.)

Conclusión

A partir de lo investigado, podemos concluir que, aunque el hardware de procesamiento se encuentra en una fase de rápida evolución para delegar las tareas de inferencia al cliente, aún enfrenta barreras críticas en términos de consumo energético y costo económico.

Por un lado, las arquitecturas tradicionales presentan extremos opuestos: la CPU sufre de cuellos de botella secuenciales insalvables para las operaciones de álgebra multidimensional, mientras que la GPU, a pesar de su paralelismo masivo impulsado por arquitecturas como CUDA y núcleos Tensor, exige un TDP y una inversión que limitan su viabilidad como estándar masivo para la computación hogareña.

Por otro lado, las Unidades de Procesamiento Neuronal (NPU) emergen como la respuesta arquitectónica idónea para mitigar este impacto energético. Como se demostró en

tareas de visión artificial en tiempo real, operan con una eficiencia algorítmica superior. Sin embargo, a este hardware especializado todavía le falta madurez. Las NPU actuales dominan la inferencia ligera, pero aún carecen de la flexibilidad, el ancho de banda masivo y el ecosistema de software maduro que poseen las GPU para ejecutar modelos complejos a gran escala.

En definitiva, para lograr la descentralización de la IA y abandonar la dependencia exclusiva de los servidores en la nube, el verdadero desafío de la industria no radica únicamente en aumentar la potencia bruta, sino en abaratar costos y superar el actual muro energético, madurando el ecosistema de los aceleradores neuronales locales.

Anexo

CPU

La unidad central de procesamiento es el componente principal funcional de una computadora. La CPU es un conjunto de circuitos electrónicos que ejecutan el sistema operativo y el diverso software de una computadora y gestionan una variedad de operaciones informáticas. El CPU es el cerebro de una computadora, y muchas veces, microprocesador, procesador, y CPU son sinónimos.

El microprocesador contiene distintos componentes, como la Unidad de Control, la Unidad Aritmético Lógica y la Unidad de Memoria. (IBM, s.f.)

TOPS

TOPS significa Tera Operations Per Second, es la unidad de medida con la que se cuantifica un rendimiento de un sistema de computación cuando está dirigido a IA. Esta unidad es importante para indicar la velocidad de ejecución de tareas de inferencia de IA, y

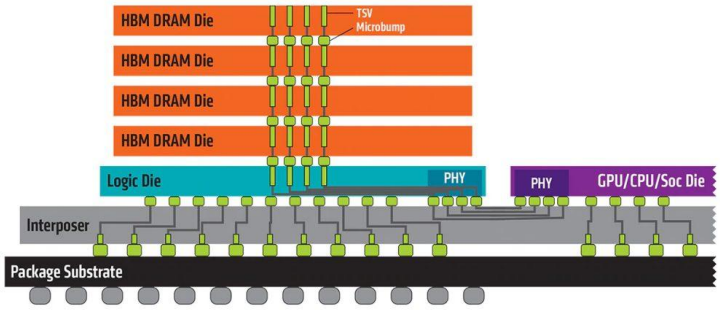
también son importantes a la hora de entrenar LLMs. 1 TOPS = 10^{12} operaciones por segundo. (Fernández, s.f.)

HBM

HBM significa High Bandwidth Memory, es decir, memoria de alto ancho de banda. La diferencia con las memorias RAM tradicionales es que se apilan los bancos de memoria uno encima de otro, hasta tener ocho unidades en una pila. Los bancos de memoria están conectados entre sí con pequeños hilos de oro para garantizar la máxima velocidad.

Luego, los *die* de memoria son conectados mediante el interposer, a diferencia de las memorias DRAM tradicionales que se conectan mediante una interfaz DIMM/SODIMM a la placa base. Claramente son mucho más veloces que las memorias tradicionales de PC (DDRx). (Guía Hardware, 2024)

Figura 5



Arquitectura de una memoria HBM. Fuente: Guía Hardware

Desenfoque gaussiano

Este método consiste en aplicar una función matemática en una imagen para desenfocarla. Es muy usado en tareas de diseño gráfico y fotografía. Para lograr esto, se logra operando una convolución a cada píxel que desea ser desenfocado más una matriz del mismo tamaño con muestras de una distribución normal (Por ese motivo el nombre “gaussiano”).

(Adobe, s.f.)

Referencias

AMD. (n.d.). AMD FidelityFX Super Resolution 3: AMD Fluid Motion Frames.

Adobe. (n.d.). *Desmitificación del desenfoque gaussiano*. Retrieved from adobe.com/mx/creativecloud/photography/discover/gaussian-blur.html **(anexo)**

AMD. (s.f.). AMD FidelityFX Super Resolution 3: AMD Fluid Motion Frames. https://www.amd.com/en/products/software/adrenalin/afmf.html

Fernández, Y. (n.d.). *Qué son los TOPS, qué miden y para qué se usa esta unidad de medida en la inteligencia artificial*. Retrieved from Xataka: xataka.com/basics/que-tops-que-miden-se-usa-esta-unidad-medida **(anexo)**

Geeks For Geeks. (2025). *What is a TPU?* Retrieved from www.geeksforgeeks.org/operating-systems/what-is-tpu-tensor-processing-unit/

Google Cloud. (n.d.). *TPU*. Retrieved from https://cloud.google.com/tpu

Guía Hardware. (2024). *HBM: qué es este tipo de memoria y para qué se usa*. Retrieved from guiahardware.es/hbm/ **(anexo)**

HP. (2026, Junio 10). *¿Por qué todos hablan de las NPU?: qué hacen y por qué son importantes*. Retrieved from hp.com/co-es/shop/tech-takes/que-es-npu-unidad-procesamiento-neuronal-explicado?msockid=3b0b521b6d6d66b0187e45736c42675b

IBM. (n.d.). *¿Qué es NPU (unidad de procesamiento neural)?* Retrieved from ibm.com/mx-es/think/topics/neural-processing-unit

IBM. (n.d.). *¿Qué es una unidad central de procesamiento (CPU)?* Retrieved from https://www.ibm.com/mx-es/think/topics/central-processing-unit **(anexo)**

Ignatov, A., Timofte, R., Chou, W., Wang, K., Wu, M., Hartley, T., & Van Gool, L. (n.d.). AI Benchmark: Running Deep Neural Networks on Android Smartphones. *arXiv preprint arXiv:1810.01109*. doi:https://doi.org/10.48550/arXiv.1810.01109

Kirk, D. B. (2016). *Programming Massively Parallel Processors: A Hands-on Approach (3ra ed.).* Morgan Kaufmann. Retrieved from Kirk, D. B., & Hwu, W. W. (2016). Programming Massively Parallel Processors: A Hands-on Approach (3ra ed.). Morgan Kaufmann.

Microsoft. (n.d.). *Información general sobre Windows Studio Effects*. Retrieved from learn.microsoft.com/es-es/windows/apps/develop/windows-integration/studio-effects

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with CUDA. *Queue, 6*(2), 40–53. https://doi.org/10.1145/1365490.1365500

NotebookCheck. (2025). AMD FSR Redstone promises faster RDNA 4 ray tracing with ML Ray Regeneration and more.

https://www.notebookcheck.net/AMD-FSR-Redstone-promises-faster-RDNA-4-ray-tracing-with-ML-Ray-Regeneration-and-more.1021290.0.html

NVIDIA. (n.d.). *Núcleos Tensor de NVIDIA*. Retrieved from nvidia.com/es-es/data-center/tensor-cores/

NVIDIA. (2022). Introducing NVIDIA DLSS 3. NVIDIA GeForce News. https://www.nvidia.com/en-us/geforce/news/dlss3-ai-powered-neural-graphics-innovations/

NVIDIA. (2025). NVIDIA DLSS 4 introduces Multi Frame Generation & enhancements for all DLSS technologies. NVIDIA GeForce News. https://www.nvidia.com/en-us/geforce/news/dlss4-multi-frame-generation-ai-innovations/

NVIDIA. (2026). NVIDIA DLSS 4.5 delivers major upgrade with 2nd gen transformer model for super resolution & 6X dynamic multi frame generation. NVIDIA GeForce News. https://www.nvidia.com/en-us/geforce/news/dlss-4-5-dynamic-multi-frame-gen-6x-2nd-gen-transformer-super-res/

NVIDIA. (2018, 14 de septiembre). *NVIDIA Turing architecture in-depth*. NVIDIA Developer Blog. https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/

NVIDIA. (s.f.). *CUDA deep neural network (cuDNN)*. NVIDIA Developer. Recuperado el 13 de junio de 2026, de https://developer.nvidia.com/cudnn

NVIDIA GeForce. (2025, Diciembre 30). DLSS 4 / DLSS 3 / DLSS 2 / DLSS Off comparison | Cyberpunk 2077 [Video]. YouTube. https://www.youtube.com/watch?v=yWYbqOFyB5Q

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU computing. *Proceedings of the IEEE, 96*(5), 879–899. https://doi.org/10.1109/JPROC.2008.917757

Patterson, D. A. (n.d.). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface.* Morgan Kaufmann.

PenBrief. (2025). *The Definitive Snapdragon X2 Elite 80 ...* Retrieved from penbrief.com/snapdragon-x2-elite-80-tops-review/

Sametband, R. (1999, mayo 24). *La Nación*. Retrieved from Aceleradoras 3D, cada día ofrecen más poder de cálculo: lanacion.com.ar/tecnologia/aceleradoras-3d-cada-dia-ofrecen-mas-poder-de-calculo-nid178170/

TechPowerUp. (2025). NVIDIA introduces DLSS 4 with Multi-Frame Generation for up to 8X framerate uplifts. https://www.techpowerup.com/330620/

TechSpot. (2026). Nvidia DLSS 4.5 arrives with a new transformer model, improved visuals, and 6x multi-frame gen. https://www.techspot.com/news/110819-nvidia-unveils-dlss-45-new-transformer-model-improved.html

Tom's Hardware. (2026). Nvidia introduces DLSS 4.5 and Multi Frame Generation 6X at CES 2026. https://www.tomshardware.com/pc-components/cpus/nvidia-introduces-dlss-4-5-and-multi-frame-generation-6x-at-ces-2026

WItt, A. (2025). *The Definitive GPU Ranking for LLMs: Token Generation & Prompt Processing Performance*. Retrieved from hardware-corner.net/gpu-ranking-local-llm/