

Las habilidades se comprimen. Los *hechos*, no.

Cinco semanas en un número, porque la cadencia semanal se detuvo tras el número 04 y esta ventana empieza exactamente donde aquella cerró. Los papers convergieron en una asimetría desde dos direcciones que no se citan entre sí: la on-policy distillation cambia cómo usa un modelo lo que sabe, no lo que sabe, y la cuantización agresiva conserva la fluidez mientras los hechos se van primero — en ambos casos, la métrica de control estándar mide la parte que sobrevive. Mientras tanto llama.cpp etiquetó tres releases, enseñó a los MoE grandes a cachear sus expertos en la GPU, hizo que el speculative decoding compense en Apple silicon y cambió su puerto por defecto. Y un tipo nuevo de modelo de una startup, Jev, de TypeSafe, que responde preguntas tipadas con probabilidades en lugar de generar texto, tuvo imitadores abiertos en tres días y soporte en llama.cpp en tres semanas.

01

La on-policy distillation enseña habilidades, no hechos. En un estudio controlado con cuatro modelos de tres familias, la OPD con KL inversa transfiere la habilidad composicional a problemas no vistos pero "transfiere un conocimiento factual mínimo"; la KL directa lo restaura 2610.09639. Un estudio con crosscoders encuentra que OPD deja más del 98% de las features que el student usa con frecuencia dentro de un 20% de sus tasas de activación 2609.35210. Cuatro grupos, cuatro métodos, una respuesta: OPD reorganiza lo que el student ya tiene.

02

Los modelos pequeños cuantizados agresivamente conservan las palabras y pierden los significados — y la perplejidad no lo ve. En 27 artefactos GGUF, todos los modelos de menos de 2B en Q2 pierden un 20–67% de su capacidad de definir palabras raras que todavía usan, mientras dos artefactos con penalizaciones de perplejidad casi idénticas pierden el 3,6% y el 43,6% de sus definiciones. Q4_K_M sigue limpio a partir de 1B 2610.04403.

03

Una clase de modelo nueva se formó en tres semanas: los decision models estilo JEV. TypeSafe lanzó Jev el 15 de septiembre, un modelo alojado que recibe un estado y preguntas tipadas y devuelve un valor con una probabilidad calibrada para cada una, sin generar texto post. Modelos abiertos con el mismo formato /v1/systemone llegaron en días — OpenJev reporta un 84,0% frente al 85,4% de Jev alojado model card — llama.cpp adoptó el formato el 2 de octubre #29818, y OpenAI publicó su propia Decisions API el 6 changelog.

04

llama.cpp se aceleró donde más dolía. El flag opcional --moe-cache-mib mantiene en VRAM los expertos usados recientemente de un MoE grande cuando el modelo vive en la RAM del sistema, reportado en 1,57–2,20× de decode en una sola RTX 4090 o 5090, con el procesamiento del prompt cayendo a 0,73× en la propia tabla de dos GPU del autor #29887 #30112. Y en los Mac M1–M4, donde DFlash2 era más lento que el decode normal, un kernel de matmul para pocas filas lo lleva de 30,2 a 110,0 t/s en código frente a 32,1 en secuencial #29869.

05

Antes de actualizar llama.cpp, lee tres líneas. El puerto por defecto del servidor es ahora 9931 #30159, mmap / --mlock han desaparecido #28334, y las sesiones guardadas antiguas ya no cargan #28498. Si expones rpc-server, el primer tag con el arreglo de un bug de ejecución remota de código sin autenticación es la v0.5.0 #24292.

La publicación de la ventana todavía no funciona upstream. DeepSeek-V4.1-Flash, con licencia MIT, 763B almacenados con 8B activos en prefill y 16B en decode, reduce su KV cache global a 890 bytes por token [2609.19969](#); su PR en llama.cpp lleva abierta desde el día del lanzamiento [#28696](#). De catorce filas de la §4, ocho funcionan hoy en llama.cpp estándar.

Gemini 4 Argon está anunciado, no disponible. Google lo anunció el 30 de septiembre con un precio introductorio de 2 \$ de entrada y 10 \$ de salida por millón de tokens y un límite de 1M tokens de salida, pero de forma escalonada: primero defensores de ciberseguridad verificados, y a ellos sin salvaguardas de ciberseguridad, luego la API de pago. Al compilar no tiene ID de modelo público ni entrada en el changelog de la API de Gemini [post](#), [30 sep](#).

CONTENIDO

- §1 llama.cpp — tres tags, un puerto nuevo y MoE que ya no tiene que caber
- §2 Los papers — las habilidades se comprimen; los hechos, no
- §3 Desde los laboratorios — una clase de modelo nueva, dos precios, menos mandos
- §4 Pesos que se publicaron
- §5 Qué haría yo

Esta ventana va de la compilación del número 04, el mediodía del domingo 6 de septiembre, a la de este, el sábado 10 de octubre — 34 días, el tamaño del mes del número 01 y no de una semana, así que las secciones son más largas y la selección más estricta. Se añadieron cuatro términos al

glosario

. **Corregido el 10 de octubre:** la primera versión cubría Jev, de TypeSafe, y los modelos estilo JEV solo como "decision models" genéricos, sin nombrar Jev ni su lanzamiento; la §1, la §3, la §4 y la §5 ya lo hacen, y Gemini 4 Argon se añadió a los puntos de arriba. Una segunda pasada, motivada por una nueva comprobación de cobertura contra Hacker News, Hugging Face y GitHub, añadió Strata (§1), y SWE-2 de Cognition, Dots, los resultados de matemáticas y el despliegue de GPT-6 de OpenAI, y ZCode de Z.ai (§3).

Tres tags, un puerto nuevo y MoE que ya no tiene que caber

718 pull requests fusionadas entre el mediodía del 6 de septiembre y el 10 de octubre, de la build `b10822` a la `b11540` — unas 150 por semana, más rápido que el ritmo del verano. En la ventana se etiquetaron tres versiones: `v0.4.1` el 14 de septiembre notas, `v0.5.0` el 23 notas y `v0.6.0` el 5 de octubre notas, así que ninguna es "la release" — y varios de los cambios de abajo se fusionaron después de la `v0.6.0` y solo existen en builds nightly. Agrupado por lo que cambia para quien lo ejecuta.

Un MoE más grande que tu VRAM tiene caché

El cambio con más probabilidades de alterar lo que ve la gente es una caché LRU en la GPU para los pesos de mixture-of-experts que viven en la RAM del sistema #29887. Hasta ahora, un MoE demasiado grande para la tarjeta ejecutaba sus expertos descargados desde memoria del host en cada token. El nuevo flag `--moe-cache-mib N` mantiene en la GPU los expertos usados recientemente y sube solo los fallos; solo entra en juego con batches pequeños, así que es una mejora de decode, y el autor sugiere dimensionarla en torno al 10% del tamaño total de los expertos. El soporte multi-GPU llegó un día después #30112.

Las cifras reportadas son sobre Qwen3.8-Flash-Next en Q4_0 (93,7 GiB, de los que 65,4 GiB son expertos) en un EPYC de 16 núcleos sobre PCIe 4.0, decode frente a la vía `--fit` existente:

CONFIGURACIÓN	DECODE T/S	ACELERACIÓN	ACIERTOS
RTX 4090, <code>--fit</code> + caché de 6,5 GB	25,0 → 39,4	1,57×	72%
RTX 4090, <code>-cmoe</code> + caché de 11 GB	25,0 → 40,7	1,62×	77%
RTX 5090, <code>--fit</code> + caché	30,8 → 54,5	1,77×	77%
RTX 5090, <code>-cmoe</code> + caché de 19 GB	30,8 → 67,8	2,20×	89%
2× RTX 4090	34,8 → 60,4 / 64,4	1,74–1,85×	–

Una máquina, un modelo, un autor: las magnitudes son afirmaciones. El detalle que debería figurar en todo resumen de esto y normalmente no figurará: en la tabla multi-GPU, **el procesamiento del prompt cae a**

0,91× y 0,73× de la base. Es opcional, va por GPU y todavía no está en ninguna release etiquetada. Si ejecutas un MoE grande en una sola tarjeta de consumo para chat, merece la pena probarlo en una nightly; si procesas prompts largos repartidos en dos tarjetas, mide antes de quedártelo.

El mismo problema se atacó aguas abajo, y atrajo más atención que nada del repositorio principal: **Strata**, un instalador y motor de un clic con licencia MIT para Windows y Linux que "usa partes de llama.cpp / ggml", ejecuta Qwen3.8-Flash-Next en un PC gaming con una tarjeta de 12 GB y 32–64 GB de RAM, tras una API compatible con OpenAI y Anthropic en localhost. Su autor reporta 94 t/s de decode y 2.650 t/s de procesamiento del prompt a 32K en una RTX 5070 con el cuantizado Q2_0, y 53 t/s en IQ3_S repo. Superó las 20.000 estrellas en GitHub en dos semanas. Lee las auditorías de cuantización de la §2 junto a las cifras en Q2: encontraron los modelos de más de unos 3B mayormente robustos en Q2 donde los pequeños no lo eran, aunque el mayor que probaron era de 14B, y que "la validación debe hacerse por artefacto".

El speculative decoding madura: correcto por encima de cero, y rápido en un Mac

Dos cambios convierten el speculative decoding de truco de centro de datos en algo que el dueño de un portátil debería activar. Primero, en las GPU de Apple sin la tensor API — de M1 a M4 — las multiplicaciones de matrices de 2 a 16 filas que necesita la verificación se ejecutaban en kernels mat-vec cuyo coste crecía con cada fila. El autor lo dice sin rodeos: en un M3 Ultra, el decode con DFlash2 de Qwen3.8-27B era por eso *más lento* que el decode normal en master. Un kernel MMA para pocas filas lo arregla #29869, extendido al resto de tipos de cuantización dos días después #30065. Reportado en un M3 Ultra con target Q4_0 y drafter Q8_0: código a temperatura 0 de 30,2 a 110,0 t/s, prosa de 16,8 a 62,6, frente a 32,1 del decode secuencial, que no cambia. Si este verano probaste DFlash en un Mac de la serie M y concluiste que no ayudaba, tenías razón, y deberías volver a probar.

Segundo, el muestreo especulativo ya es correcto con temperatura. Los drafters basados en modelo solo ofrecían su token más probable, y el target lo aceptaba solo si coincidía exactamente, así que la aceptación se desplomaba al subir la temperatura. La construcción estándar de muestreo por rechazo está ahora disponible tras `--spec-draft-sampling probabilistic` #27694, manteniendo exacta la distribución de salida. Las ganancias crecen con la temperatura — un target Llama-3.1-8B con un draft de 3B pasa de +0,07% a T=0,2 a +16,6% a T=0,9 en una RTX 5090 — y algunas configuraciones empeoran un poco a temperatura baja, que es por lo que el valor por defecto sigue siendo greedy. Se aplica a drafts simples y a MTP; el autor indica que DFlash no se beneficia. También trajo un bug: los drafts n-gram cerca del final de una respuesta se rechazaban enteros a $T > 0$ hasta #29924, dos días después.

Arquitecturas nuevas, y una que estaba rota sin que se notara

GLM-5.3-Flash llegó el 30 de septiembre #27773, descrito por el autor como un híbrido de texto y visión de 320B con 34 capas de atención lineal KDA y 11 capas DSA, que reutiliza el código de KDA y de

hiperconexiones de modelos anteriores; MTP llegó una semana después #29928. **Qwen3.8-Flash-Next**, cuyo soporte el número 04 reportó como recién llegado, resultó estar "roto en el master actual" según la PR que lo arregló el 1 de octubre #29751. Ese mismo día ganó drafting MTP, reportado en $1,55\times$ de decode global en un DGX Spark, entre $1,43\times$ y $1,67\times$ según la categoría de prompt #29761. Si te formaste una opinión de ese modelo en llama.cpp en septiembre, vuelve a formártela.

Menor, pero toca a todos los modelos híbridos que se usan: la normalización de q/k de Gated DeltaNet difería ligeramente de las implementaciones de referencia, y afectaba a los híbridos Qwen3.5 y 3.8, a Kimi-Linear y a Kimi-K3, entre otros #28068. El cambio medido en divergencia KL es minúsculo; es el tipo de arreglo que obtienes gratis al actualizar.

Lo genuinamente nuevo es una *clase* de modelo: los decision models estilo JEV, que responden preguntas tipadas con una probabilidad por opción en lugar de generar texto (la §3 cuenta la historia de Jev, de TypeSafe, y sus imitadores abiertos). El runtime adoptó el propio formato de Jev, un endpoint de servidor `/v1/systemone`, con Laya, Julia-1, lev, Kev y OpenJev soportados desde la primera PR, cuyo autor llama a los abiertos "envoltorios sofisticados de modelos de embeddings tradicionales" #29818. En nueve días los siguieron Clef, de Cloudflare #29831, pplx-decider, de Perplexity #30044, y los modelos d1 de Liquid AI #30110, entre diez PRs. Diez PRs en nueve días es una categoría formándose, no algo puntual. También se fusionaron un MoE ternario de 20B-A1B, Maple, de momento solo en CPU #27000, y un modelo en bucle, HRM-Text #27625 — digno de mención dado cuánto de la §2 va de modelos en bucle.

Contexto largo, en todos los backends

La sparse flash attention, que el número 04 cubrió al llegar a CUDA y Metal, se extendió a lo largo de la ventana a Vulkan #28105, SYCL #28796 (desactivada por defecto, tras `GGML_SYCL_SPARSE_FA=1`), CUDA para Qwen3.8-Flash-Next #28770 y Vulkan con KV cache cuantizada #29639. La forma que describió el número 04 no ha cambiado: las ganancias reportadas son $1,08\times$ de prefill a 10K y $1,26\times$ a 100K en Qwen3.8-Flash-Next, y nada a profundidad de chat. Algo que conviene saber si usas DeepSeek-V4: #28770 *desactivó* su vía sparse por una puerta que dependía del batch, y el prefill se reactivó dos días después #29298.

El prefill denso de contexto largo se aceleró en tres sitios. La tensor API de Metal mueve ahora la flash attention en el M5, reportado de 174 a 253 t/s en pp512 a 32K de profundidad en un M5 Pro, con el decode sin cambios #29570. Los kernels FA de dos etapas de CUDA planifican ahora tiles completos, de +1,4% a +33,5% en prefill de 64K tokens en un DGX Spark y como mucho +4,8% en una RTX 5090 #29435. Y ROCm en RDNA4 pasó de 164 a 399 t/s a 150K de profundidad en Qwen3.5-27B, en parte por arreglar un bug que impedía seleccionar los kernels de tamaño de cabeza 256 #28102.

En otras partes, una línea cada una: el prefill en CPU sobre k-quants tiene un kernel VNNI por tiles, reportado en $3,8\text{--}7,0\times$ — un microbenchmark de GEMM, no throughput de extremo a extremo #27851; AMD RDNA3 y 4 tienen matmuls en int8 con matrix cores bajo Vulkan, $1,25\times$ en prefill de Llama-3-8B en Strix Halo #27952; el decode de gpt-oss casi se duplicó en Intel Arc Pro bajo SYCL en Windows, en la última build

de la ventana #29809; la carga con direct-IO dejó de mantener cada tensor dos veces, reduciendo a la mitad el pico de RSS en un DGX Spark #29749; y el backend para la NPU Hexagon de Qualcomm es ya una línea de trabajo estable, con 44 PRs en la ventana y builds para Snapdragon en las releases #29007.

Tres cambios que romperán una instalación que funciona

El puerto por defecto del servidor es ahora 9931, no 8080 #30159, desde la build b11521 del 9 de octubre. Las imágenes Docker mantienen 8080 dentro del contenedor. Cualquier cosa con localhost:8080 fijado en el código que actualice una build nativa necesita --port 8080. Los flags obsoletos --mmap, --mlock y --direct-io han desaparecido, sustituidos por --load-mode #28334, así que los scripts de arranque antiguos fallan al parsear desde la v0.4.1. Y las sesiones guardadas por builds anteriores ya no cargan tras un cambio de formato de sesión que llega con la v0.6.0 #28498.

En el servidor, las incorporaciones útiles: los checkpoints de contexto sobreviven ahora a guardar y restaurar un slot, así que restaurar un slot en un modelo híbrido o de ventana deslizante ya no reprocesa el prompt entero #26004; --host acepta una lista de direcciones y sockets Unix #28690; los rerankers basados en LLM causales ya no fallan con documentos más largos que el batch físico #28876; y la interfaz web se convirtió en un gestor de modelos que descarga, carga, descarga de memoria y borra modelos #29583.

SI EXPONES RPC-SERVER, LEE ESTO PRIMERO

Un use-after-free en el servidor RPC, alcanzable por un cliente sin autenticar y que permitía ejecución remota de código, se corrigió el 16 de septiembre #24292 (aviso GHSA-2phh-px2f-2qmx). Se fusionó dos días después de etiquetar la v0.4.1, así que la primera release etiquetada que lo contiene es la v0.5.0.

Una escritura fuera de límites distinta, también alcanzable por RPC, se corrigió el 9 de octubre #30135 y solo está en nightlies. En la misma ventana se endureció el cargador frente a GGUF manipulados (#29598, #29384). El servidor RPC nunca se pensó para una red no fiable; esto recuerda por qué.

Un cambio que permitía al auto-fit dimensionar una KV cache unificada entre slots paralelos se fusionó el 16 de septiembre #28849 y se revirtió el 26 #29437, con un revisor señalando que subía mucho los requisitos de VRAM para quien llena la tarjeta con un modelo. Un cambio de tiling de MoE en Vulkan que daba +10% de prefill en Strix Halo #29182 redujo casi a la mitad el procesamiento de prompts de 8K en Intel Arc y se revirtió solo para Intel tras un bisect #29936.

Dos compromisos deliberados que conviene conocer: MXFP4 en Metal calcula ahora en bf16 por corrección, y el autor acepta una ralentización en chips anteriores al M3, $0,81\times$ de prefill en un M2 Ultra #29770; y un cambio en GLM-5.3-Flash que acelera el decode con flash attention activada hace caer el decode a 32K de 72,3 a 24,6 t/s con `-fa off` y micro-batches pequeños, en la propia tabla del autor #30042. Abiertos al compilar: MTP reduciendo aproximadamente a la mitad el procesamiento de prompts con reparto de capas multi-GPU *issue* #27428, y la salida de razonamiento de Gemma 4 acumulando basura al final *issue* #28827. Otra ventana, otro revert y re-merge; fija las builds de las que dependes.

Las habilidades se comprimen; los hechos, no

Cinco semanas de arXiv, recorridas semana a semana para que ninguna parte de la ventana quedara truncada, más los daily papers de Hugging Face — 1.084 dentro de la ventana. Dos literaturas que no se citan entre sí convergieron en la misma asimetría. La destilación cambia cómo usa un modelo lo que sabe, no lo que sabe. La cuantización agresiva conserva la fluidez y pierde primero los hechos. Y en ambas, la métrica de control estándar mide la parte que sobrevive.

La on-policy distillation reorganiza; no añade

El número 04 se preguntaba si el teacher de OPD hacía algo en absoluto. Esta ventana respondió desde cuatro direcciones independientes, con cuatro métodos distintos, y las respuestas coinciden.

La más limpia es un estudio sintético controlado que varía por separado los *hechos* adicionales del teacher y su *habilidad composicional* adicional. En cuatro modelos de tres familias, la OPD con KL inversa transfiere la habilidad a estructuras de razonamiento no vistas pero "transfiere un conocimiento factual mínimo"; cambiar

a KL directa restaura la transferencia factual, y la misma asimetría aparece en QA factual real y en matemáticas de competición 2610.09639. Un estudio mecanicista con sparse crosscoders llega al mismo sitio desde dentro de la red: OPD "ni crea features ni transmite las del teacher", y deja las tasas de activación de más del 98% de las features que el student usa con frecuencia dentro de un 20% 2609.35210. Tratando al teacher como un modelo de recompensa implícito, un tercer paper concluye que OPD "mejora el rendimiento sin ampliar las capacidades del student" — hace más fácil muestrear respuestas correctas, y cuando la recompensa implícita está desalineada, premia rollouts largos y repetitivos 2610.03185. Y un cuarto atribuye el estancamiento de OPD con teachers grandes a una señal de aprendizaje que se desvanece: cambio relativo de parámetros del 0,025–0,098% y similitud por capa superior a 0,98, frente a reducciones de pérdida mucho mayores cuando el teacher es el propio sucesor del student entrenado con RL 2610.11247.

Los papers de receta se alinean detrás. Una recompensa de ± 1 en los tokens donde la probabilidad del teacher está por encima o por debajo de la del student "reproduce casi el mismo modo de entrenamiento que OPD con KL inversa" 2609.33791. Un estudio sistemático concluye que la política de rollout — lo que pone el "on-policy" en OPD — "no juega necesariamente un papel central"; lo hacen la dirección de la KL y la tasa de aprendizaje, y la ventaja on-policy "no persiste de forma fiable tras un RLVR posterior" 2609.35259. Las leyes de escalado de OPD dentro de una misma familia muestran que la puntuación máxima mejora con el tamaño del teacher "solo hasta aproximadamente la escala del student", que "a igual puntuación, los teachers más pequeños transfieren mejor", y que en todos los pares débil-a-fuerte observados el student superó a su teacher 2609.32722. La OPD con varios teachers no supera a un student enseñado por el mejor especialista individual, porque el feedback de seguimiento de instrucciones está más disperso y domina las actualizaciones 2609.35347.

Nada de eso hace inútil a OPD; la convierte en otra herramienta. Si un modelo está reorganizando habilidades que ya tiene, la OPD barata es la OPD correcta, y tres papers la abarataron mucho: supervisar el 0,1–1% de los tokens iguala a la OPD completa 2609.24432; restringir la OPD entre tokenizers distintos a un top-16 del vocabulario compartido elegido por el student iguala a la versión completa 2610.08448; y SparseOPD reduce la memoria del backward un 70,5% con una similitud coseno de gradiente del 99% 2609.34386. Dos modos de fallo con arreglo que conviene conocer si la usas: student y teacher pueden poner su probabilidad de parada en tokens EOS *distintos* aunque los conjuntos de parada declarados sean idénticos, lo que infla la longitud hasta que se tratan los EOS equivalentes como una sola acción de parada 2609.20511; y el clipping puntual heredado de la receta original de on-policy self-distillation puede empujar las probabilidades de los tokens *lejos* del teacher, produciendo repeticiones que persisten hasta el final de la respuesta 2609.38995.

El paper más votado del tema apunta en la otra dirección, y merece leerse por eso: RIDE trata el cambio del teacher de RL respecto a su checkpoint previo al RL como una dirección en el espacio de estados ocultos y extrapola *más allá* del teacher a lo largo de ella, acercándose al teacher de RL o superándolo en los cuatro pares probados 2609.36484. Fíjate en lo que extrapola: el residuo del RL, una habilidad. Es coherente con el resto de la sección, no un contraejemplo.

Cuantización: el daño está donde la métrica de control no mira

El número 04 terminó con la tercera semana consecutiva en la que 4 bits pasaba el control estándar y fallaba uno de comportamiento. Esta ventana produjo un grupo de auditorías que midió exactamente eso, varias de ellas sobre los propios formatos de llama.cpp.

Saying, Not Knowing audita 27 artefactos GGUF de 13 familias, de 0,35B a 14B, bajando por la escalera publicada hasta Q2_K, con 429 palabras raras del inglés, y hace dos preguntas: ¿puede el modelo todavía *usar* la palabra, y puede todavía *definirla*? En todos los artefactos de menos de 2B en Q2, las definiciones caen un 20–67% por debajo de la propia base del artefacto, normalmente varias veces la pérdida en uso. El número de parámetros predice el daño (Spearman $\rho = 0,72$); el tamaño del vocabulario del tokenizer no ($\rho = 0,12$). La perplejidad en WikiText-2 no distingue los casos: dos artefactos con penalizaciones de perplejidad en Q2 casi idénticas, 44,7% y 47,6%, pierden respectivamente el 3,6% y el 43,6% de sus definiciones. La mitad tranquilizadora: Q4_K_M "sigue léxicamente limpio a partir de 1B" ^{2610.04403}.

La misma forma a otras escalas y en otros comportamientos. Modelos ternarios entrenados desde cero retienen tan solo el 6% de la capacidad factual de un modelo fp16 entrenado igual mientras la perplejidad sube solo 1,4–1,6× — a 50M parámetros o menos, así que es un mecanismo, no un veredicto sobre modelos de la clase BitNet ^{2609.32902}. Midiendo la divergencia sobre todo el vocabulario en lugar de la precisión, una matriz de 120 ejecuciones encuentra la precisión "inestable, no monótona" bajo la cuantización progresiva de llama.cpp, y el Q4_K de precisión mixta en general más cerca de BF16 que el Q4_0 uniforme a tamaño parecido ^{2609.07664}. En uso agéntico de herramientas, tres modelos polacos caen por un precipicio entre 3 y 2 bits — de 0,716 a 0,045 el de 11B — pero fallan de formas distintas: el 7B podado falla a lo largo, el 11B confabula una respuesta final en el primer paso ^{2609.32042}.

EL PANEL DE ADVERTENCIA – TU VEREDICTO DE CUANTIZACIÓN PUEDE SER UN ARTEFACTO DE CÓMO LO PUNTUASTE

Comparando Llama-3.1-8B en 8 y en 4 bits en recuperación de fallos de herramientas, un paper obtiene todas las respuestas posibles a partir de los mismos logs. Puntuado sobre las tareas limpias de cada modelo, 4 bits gana por 17,5 puntos; sobre tareas emparejadas, no hay diferencia; sobre el pipeline completo, 8 bits gana por 28,3 — y un parseo estricto de la salida convierte ese +28,3 en –15,0. En Qwen2.5-7B la diferencia solo entre prompts va de –50 a +35 puntos ^{2610.07781}.

Ponlo al lado de la auditoría GGUF de arriba, cuya conclusión es que "la validación debe hacerse por artefacto", y la regla transferible es incómoda pero barata de seguir: **que un modelo cuantizado pase la perplejidad te dice que es fluido, no que siga sabiendo cosas ni que siga sabiendo usar herramientas**, y un único prompt, conjunto de tareas y política de puntuación tampoco establece un veredicto estable en ningún sentido. Para un cuantizado que despliegas de verdad, prueba el comportamiento del que dependes, con más de un prompt.

Tres hallazgos sobre las *herramientas* de la cuantización, más que sobre sus resultados. Una pérdida de reconstrucción más baja puede degradar un modelo sobre los mismos datos de calibración con los que se minimizó [2610.11226](#). Un estudio de precisión mixta encontró una ejecución que mejoró $4,6\times$ el objetivo propio de un bloque mientras la perplejidad subía $32\times$, y una regla basada en la matriz en la que está cada peso superó a todas las estimaciones de sensibilidad que calcularon — el 96,4% de los pesos a 4,5 bits por parámetro, a menos de un 4,4% de la perplejidad en precisión completa en Mistral-7B [2609.34866](#). Eso hace eco del hallazgo del número 04 de que las teorías intuitivas sobre dónde vive el daño de cuantización no predicen nada. Y el resultado más práctico de la ventana: **los pipelines de exportación documentados de BitNet, Falcon-E y BitCPM convierten los latentes a bf16 antes de ternarizar**, el redondeo de bf16 deja valores exactamente en el umbral, y el redondeo a par los lleva a cero. En los puntos finales de un fine-tune esto baja la precisión estricta greedy en GSM8K del 58,79% al 0,78% en Falcon-E-1B — la exportación revierte el fine-tune en silencio [2610.07853](#).

Cuantización que puedes usar

La ventana también produjo trabajo de cuantización dirigido directamente al runtime de la §1.

Disaggregated Quantization, del IST Austria, parte de que el prefill está limitado por cómputo y premia la aritmética de baja precisión mientras el decode está limitado por memoria y premia pesos compactos, y usa formatos distintos para cada uno. Con decodificadores GGUF de 1 bit publicados de Qwen3.8-27B, entrenar un *prefiller* NVFP4 mejora la precisión 32,5 puntos en MMLU-Pro sin tocar el checkpoint de decode, y leer los pesos de prefill desde SSD da una aceleración reportada de $1,78\times$ en el tiempo hasta el primer token a 8K en llama.cpp [2609.26333](#). **QATFactory** es un framework abierto de entrenamiento y destilación conscientes de la cuantización que exporta directamente al Q4_K de llama.cpp y a vLLM sin una segunda conversión con pérdidas; en Qwen3.5-9B su QAD alcanza un 68,9% de precisión media en NVFP4 frente al 65,4% del mejor método post-entrenamiento [2609.39223](#). **Tetra** sirve codebooks de la red de Leech a unos 2,7 bits por parámetro y, a 4B, puntúa 23,6 puntos por encima del IQ2_XXS de llama.cpp en su benchmark [2609.35465](#). Y Minima, en el número 04, cuantizaba los pesos Gated DeltaNet de los modelos híbridos; **STEPQuant** cuantiza su *estado* recurrente, que se convierte en el cuello de botella de memoria al servir en concurrencia, igualando de cerca la precisión con estado en FP32 a un presupuesto nominal de 6 bits y reduciendo la memoria total de servicio hasta un 68,7% en SGLang [2609.38169](#).

Los modelos en bucle se convierten en un eje de eficiencia

Los modelos que ejecutan el mismo bloque varias veces por token — en bucle, o de profundidad recurrente — pasaron de unos 1,5 a 9,3 papers por cada mil envíos a cs.CL y cs.LG a lo largo de la ventana, y seguían subiendo en la última semana. Los papers han pasado de si el bucle funciona a cómo servirlo. Una ley de escalado que modela juntas recurrencia y dispersión encuentra que un MoE en bucle iguala en razonamiento a uno sin bucle de aproximadamente el doble de tamaño, a igual cómputo de entrenamiento [2609.40316](#). Dejar que solo el primer bucle escriba la KV cache, y que los siguientes la lean, *mejora* la perplejidad usando un 76–79% menos de memoria de contexto entre 150M y 1B [2610.02383](#). Los bucles superficiales pueden hacer de draft para los profundos, una auto-especulación sin entrenamiento reportada en $2,42\times$ en Ouro-

2.6B y $3,54\times$ en Huginn-3.5B [2609.23033](#). Y la intuición de que el error de redondeo se acumula de bucle en bucle "solo se cumple en bucles que nunca se estabilizan" [2609.39277](#).

El paper que lo conecta con el resto de la sección: un estudio controlado encuentra que la recurrencia puede mejorar el razonamiento más allá del horizonte de entrenamiento *mientras degrada el conocimiento* [2609.36636](#). Es un solo paper, y la pata más débil de la tesis de este número — pero es la misma asimetría por tercera vez.

Servir modelos: dos auditorías y un MoE de 35B en 3 GiB

EL SPECULATIVE DECODING "SIN PÉRDIDAS" LO ES EN FP32

Una reproducción independiente de un decodificador especulativo con auto-draft ejecutó 1.190 prompts de 12 dominios y encontró que, en BF16, la salida especulativa coincide exactamente con el decode normal solo en el 45% de las generaciones del checkpoint de los autores y en el 43% de las de un modelo entrenado de forma independiente. En FP32 coincide en todos los prompts, y las puntuaciones posteriores no muestran degradación sistemática [2609.15504](#). La lectura práctica: si comparas la salida especulativa con la normal en bf16 o fp16 como prueba de corrección, la divergencia es esperable y no es un bug. La salvedad es que se probó una sola arquitectura; no está establecido si los drafters tipo EAGLE o DFlash se comportan igual.

La otra auditoría es para quien use un modelo con compresión de KV cache por bloques: comprimir ventanas fijas en menos entradas da a cada token una coordenada nueva, su fase respecto al borde de la ventana, y la recuperación en contexto largo en modelos abiertos grandes puede variar hasta 40 puntos entre fases — "una precisión media alta puede convivir con fallos posicionales sistemáticos" [2609.36322](#). En la misma línea, un paper de metodología sostiene que las evaluaciones actuales de reutilización de KV cache para RAG inflan sistemáticamente lo bien que funciona [2609.31415](#).

Edge0 es el repositorio relevante para este boletín con más estrellas de la ventana, más de 4.000: un predictor por capa adivina el enrutado de la capa siguiente un token antes, y esa predicción es el enrutado, de modo que los expertos traídos desde SSD son exactamente los que se usan. En una sola máquina de 24 GB sirve un MoE de 35B a 20 tok/s dentro de 3 GiB de memoria activa máxima, "a pocos puntos" de su teacher fp16 [2609.18063](#). Cambia el enrutado y necesita un adaptador de recuperación, así que no es un reemplazo directo para un GGUF existente — léelo junto a la caché de expertos de la §1, que resuelve el mismo problema sin reentrenar. Dos resultados de drafting cierran la sección: **DLoop** sigue haciendo draft mientras el drafter está seguro en lugar de verificar tras cada bloque, reportado en un 5–41% más de aceleración real con EAGLE-3, DFlash, DSpark y módulos MTP [2610.07659](#) — la secuela directa del *stranded speed-up* del número 04; y **AgSpec** hace draft a partir de la sesión y el espacio de trabajo en pipelines de agentes de código, reportado en hasta $4,37\times$ con batch 1 [2610.01108](#).

Una clase de modelo nueva, dos precios y menos mandos

Noticias técnicas de los grandes laboratorios, con cada fecha leída en el propio post o entrada de changelog del laboratorio. Cinco semanas trajeron una clase de modelo nueva de una startup, tres modelos Claude nuevos, tres variantes de GPT-6, un Grok, un buque insignia de Mistral, un modelo de código cercano a la frontera construido sobre una base abierta y un Gemini 4 anunciado. Agrupado por lo que cambia para quien construye sobre ellos, no por empresa.

JEV, y una clase de modelo que se formó en tres semanas

El lanzamiento con más consecuencias de la ventana no vino de un gran laboratorio. El 15 de septiembre **TypeSafe** salió del modo oculto con **Jev**, el primero de lo que llama modelos "System One": "estado no estructurado de entrada, decisiones probabilísticas tipadas de salida". Envías un estado y un esquema de preguntas tipadas — una elección, un sí o no, una puntuación — y recibes un valor para cada una con "probabilidades calibradas y puntuaciones de confianza", en una sola pasada y sin generar texto. TypeSafe cobra la entrada a 0,042 \$ por millón de tokens con la salida "GRATIS (demasiado barata para medirla)", y reporta 70–500 ms de extremo a extremo. Solo está alojado, en acceso anticipado con lista de espera, y el post de lanzamiento da múltiplos de velocidad y coste pero ningún benchmark numérico de precisión `post`. Una nota sobre la fecha: el post de TypeSafe lleva ahora una marca del 9 de octubre, pero la cobertura fechada el 16 de septiembre cuenta que la empresa salió del modo oculto "el martes", el 15 `VKTR`, 16 `sep`, y los modelos abiertos de abajo citan a Jev desde el 18 de septiembre. Ambas fechas caen en esta ventana.

Lo que vino después es la historia. Tres días tras el lanzamiento salió **Laya**: un encoder ModernBERT-large de 421M con una cabeza de decisión que expone "la misma forma de petición y respuesta `POST /v1/systemone` que TypeSafe Jev" `model card`. Cinco días después, **OpenJev**, un modelo de pesos abiertos de 27B que declara no estar afiliado a TypeSafe pero sigue la API alojada de Jev, se sitúa "a menos de 2 puntos de la API alojada de Jev" en 10.000 preguntas de texto, 84,0% frente a 85,4%, con licencia no comercial `model card`. A finales de septiembre había adaptadores estilo JEV sobre bases Qwen (Kev, lev), uno multilingüe de 144M (Julia-1), uno de 0,9B cuya card dice que "los clientes escritos para Jev o Kev funcionan sin cambios" `model card`, y **JEV-27B-VL** de AutoTrust, un decision model con visión sobre Qwen3.8-27B que, según su card, quedó primero de 20 en el tablero de visión de un Jev Decision Index gestionado por la comunidad `model card`. Cloudflare, Perplexity, InternLM y Liquid AI siguieron con los suyos (§4).

Los runtimes convergieron en el formato de Jev en lugar de inventar uno. El endpoint `/v1/systemone` de llama.cpp, fusionado el 2 de octubre, "sigue el formato System One introducido con el modelo Jev de TypeSafe, así que los clientes existentes solo necesitan una nueva URL base" [post](#), 2 oct (§1). La **Decisions API** de OpenAI, en beta desde el 6 de octubre sobre GPT-6 Luna, recibe texto e imágenes y devuelve respuestas tipadas, con una velocidad declarada diez veces mayor que su Responses API [changelog](#), 6 oct. El modelo alojado de una startup, más de una docena de decision models abiertos, un runtime abierto y la API de un gran laboratorio, todo en 21 días, es la señal más clara de la ventana de que esto es un componente con el que la gente va a construir: clasificación, enrutado, moderación y elección de la siguiente acción de un agente como una sola pasada con una probabilidad por opción, en lugar de generar texto y parsearlo. Dos cautelas. Una respuesta calibrada y válida según el esquema puede seguir siendo errónea con toda seguridad; la seguridad de tipos elimina fallos de parseo, no errores. Y la evidencia es joven — las comparaciones de precisión son de los propios modelos abiertos, casi siempre frente a Jev, en tableros gestionados por sus autores o por la comunidad.

La frontera converge en dos precios

El nivel medio ya tiene precio de mercado. **Claude Sonnet 5.5** se lanzó el 28 de septiembre a 2 \$ por millón de tokens de entrada y 10 \$ por millón de salida [post](#), 28 sep, el mismo precio que **GPT-6 Sol** de OpenAI el 22 de septiembre y **GPT-6.1 Sol** el 29, ambos con un contexto de 1,05M tokens [changelog](#), 22 y 29 sep, y el mismo que el precio introductorio que Google anunció para Gemini 4 Argon (más abajo). **Grok 4.7**, el 21 de septiembre, rebaja la salida a 2 \$ y 6 \$ con un contexto de 500K [post](#), 21 sep. El nivel pequeño también convergió: **Claude Haiku 5.5** el 7 de octubre y **GPT-6 Luna** el 22 de septiembre cuestan ambos 0,10 \$ de entrada y 0,50 \$ de salida, Haiku para prompts de hasta 100K tokens [post](#), 7 oct. Y **Claude Opus 5.5**, el 22 de septiembre, llegó *más barato* que su predecesor, a 4 \$ y 20 \$, con 1M de contexto por defecto [post](#), 22 sep. En el lado del consumidor, OpenAI empezó el 7 de octubre a desplegar GPT-6 para todos en ChatGPT, con una "Intelligent UI" que responde con elementos visuales e interactivos [post](#), 7 oct.

Los benchmarks de todos ellos están en sus posts, y la mayoría sobre la batería de evaluación de cada fabricante; un detalle merece repetirse porque es inusual: Anthropic reporta Sonnet 5.5 *por encima* de Opus 5.5 en Terminal-Bench 4.0, 70,6% frente a 66,4%. Para quien elige un modelo por defecto, la lectura práctica es que un modelo de 2 \$/10 \$ es ahora donde está el precio de la frontera, y todo lo que esté por encima tiene que justificarse en tu carga de trabajo.

Una base abierta, post-entrenada hasta cerca de la frontera

SWE-2, de **Cognition**, publicado el 10 de septiembre en la app de escritorio y la CLI de Devin, está post-entrenado a partir de Kimi K3, un modelo de pesos abiertos. Cognition lo reporta a un punto de Claude Fable 5.1 en FrontierCode 1.1 Main, 50,0% frente a 50,9%, con un coste un 64% menor, y por delante en DeepSWE 1.1, 73,0% frente a 67,4%. Su propia tabla muestra también el límite: 27,3% en Terminal-Bench 4 frente al 55,8% de Fable [post](#), 10 sep. No hay API ni declaración sobre los pesos. Lo que importa para este boletín es la receta: una empresa sin un preentrenamiento de frontera llevó una base abierta cerca de la

frontera en los benchmarks que buscaba, y la dispersión entre benchmarks dice exactamente cuánto. La fecha del post aparece como "09.10.26", con el mes primero; se discutió el 10 de septiembre.

Menos mandos, otra vez

El número 04 señaló que GPT-6 Astra no aceptaba temperatura. Esa es ahora la dirección de todos los fabricantes, y cada caso es un cambio incompatible para alguien. Opus 5.5 devuelve un 400 si intentas desactivar el razonamiento; Sonnet 5.5 sustituye "disabled" por un nuevo modo `between_tools`; Haiku 5.5 rechaza directamente `temperature`, `top_p`, `top_k`, los presupuestos de razonamiento y el prefill del asistente `notas de versión`; y GPT-6.1 Sol no tiene esfuerzo de razonamiento `none` `ficha del modelo`. Si tu código fija parámetros de muestreo en cada llamada, actualizar de modelo es ahora un cambio de código. Las razones declaradas varían; el efecto es el mismo: la frontera alojada se está convirtiendo en un componente cuyo decoding no controlas, que es un argumento más para mantener un modelo local donde necesites ese control.

El nivel restringido se volvió la forma normal

El número 04 llamó "un modelo público más un hermano restringido para seguridad" a dos casos y se negó a llamarlo tendencia. Cinco semanas después es como publican los laboratorios. Anthropic reestructuró su Cyber Verification Program en tres niveles el 6 de octubre y reporta, en su propio benchmark de escenarios, que Opus 5.5 quedó bloqueado en todas las tareas sin inscripción `post`, 6 oct; añadió un Life Sciences Verification Program el 17 de septiembre `post`, 17 sep, y sus tres modelos 5.5 desvían las peticiones de ciberseguridad y biología de mayor riesgo a un modelo anterior salvo que la cuenta esté inscrita. Gemini 4 Argon, de Google, va *primero* a defensores de ciberseguridad verificados, y a ellos sin salvaguardas de ciberseguridad `post`, 30 sep. El modelo de ciencias de la vida de OpenAI es solo de acceso de confianza `changelog`, 8 sep, y las capacidades de red-team de Grok 4.7 son solo por invitación. Dos consecuencias prácticas: el mismo ID de modelo puede ahora responder distinto según el programa en el que esté tu organización, y Anthropic volvió a facturar desde el 24 de septiembre algunas negativas por salvaguardas que no producen salida `notas de versión`.

Los laboratorios publican runtimes de agentes

La **Agents API** de OpenAI entró en beta pública el 10 de septiembre, con OpenAI gestionando la orquestación de sesiones, la compactación y la recuperación en sandboxes propios o del cliente, y ganó computer use el 29, el mismo día en que GPT-6.1 Sol trajo delegación multiagente dentro de una sola petición a la Responses API `changelog`. Anthropic publicó compactación bajo demanda (14 de septiembre), herramientas que se pueden añadir a mitad de conversación sin invalidar la caché del prompt (22 de septiembre) y flujos multiagente dinámicos para Managed Agents (9 de octubre); la línea que conviene notar para quien se preocupe por la prompt injection es del 7 de octubre, cuando el `web_fetch` de Managed Agents empezó a rechazar URLs que no estuvieran ya en la sesión `notas de versión`. OpenAI también llevó los

agentes al consumidor: **Dots**, lanzados el 29 de septiembre, son asistentes proactivos en ChatGPT que siguen trabajando a lo largo de proyectos y tareas [post](#), 29 sep. Z.ai publicó como código abierto **ZCode**, su entorno de programación — app de escritorio, interfaz web y agente de terminal, con el runtime del agente en el repositorio — bajo Apache 2.0 el 20 de septiembre, y superó las 7.000 estrellas [repo](#). Google sustituyó su agente Antigravity el 17 de septiembre con un *esquema de herramientas integradas cambiado* — herramientas y parámetros renombrados, incompatible para quien parsee las llamadas a herramientas — y apagó la versión antigua el 5 de octubre [changelog](#), 17 sep. Lo más relevante para este boletín: el SDK de Antigravity funciona ahora con modelos locales, Gemma 4 en LiteRT o cualquier servidor compatible con OpenAI, lo que incluye `llama-server` [post](#), 23 sep.

Anunciado, no publicado

Gemini 4 Argon se anunció el 30 de septiembre con un precio introductorio de 2 \$ y 10 \$ y un límite de 1M tokens de salida, pero como lanzamiento escalonado: primero los defensores, luego la API de pago, luego el resto. Al compilar no tiene ID de modelo publicado ni entrada en el changelog de la API de Gemini, así que trátalo como anunciado, no disponible [post](#), 30 sep. **Mistral Large 4**, un MoE de aproximadamente 1T parámetros totales y 52B activos con 1M de contexto, entró en preview pública el 6 de octubre, y Mistral promete pesos abiertos "antes de final de mes", sin licencia declarada [post](#), 6 oct. Si llegan, irán a una futura tabla de publicaciones; hasta entonces son una promesa.

Investigación, y lo demás

Tres resultados de investigación con validación en laboratorio húmedo o por correctores oficiales, algo más raro de lo que debería. Anthropic puso unos 950 agentes Claude durante 21 horas sobre una base de datos de secuencias y encontró una transcriptasa inversa con un gen accesorio y una matriz de repeticiones tipo CRISPR, que luego expresó y caracterizó; su función se desconoce, y el [post](#) lo dice [post](#), 23 sep. Google DeepMind publicó AlphaGenome Atlas, predicciones precalculadas para unos nueve mil millones de variantes de una sola letra del genoma humano, gratis para uso no comercial [post](#), 8 sep, y SynthID Bio, marcas de agua para secuencias y estructuras de proteínas que superaron pruebas en laboratorio húmedo sin perjudicar la unión [post](#), 30 sep. OpenAI publicó nuevos resultados sobre problemas abiertos de matemáticas obtenidos con un modelo de frontera interno, con las formalizaciones en Lean de las demostraciones publicadas en GitHub — las demostraciones formales son la parte que cualquiera puede comprobar [post](#), 6 oct. NVIDIA reporta sistemas Nemotron a nivel de medalla de oro en la IOI 2026, en una ejecución en directo no oficial, y en la IMO 2026, corregida por los correctores oficiales [post](#), 7 oct.

Para quien trabaja con esto, el [post](#) más útil de Hugging Face en la ventana: Transformers puede ahora ejecutar los cuantizados Q4_K_M, Q5_K_M y Q6_K de llama.cpp empaquetados en Apple silicon mediante los kernels Metal de ggml, para modelos de la familia Qwen3.5 y solo en la rama main, con un throughput mostrado como "cercano a llama.cpp" en gráficas que no comparan lo mismo [post](#), 22 sep. Los formatos GGUF se están convirtiendo en una lengua franca más allá del runtime que los definió.

Meta no publicó nada en su blog de IA por tercer número consecutivo; su post más reciente sigue siendo del 27 de julio. **Qwen** solo publicó modelos de imagen, y **Microsoft** ningún Phi nuevo.

§ 4 · PUBLICACIONES

Pesos que se publicaron

Solo pesos abiertos; las especificaciones y cifras son las de las model cards, y los benchmarks son los de los fabricantes. Las fechas son la de creación del repositorio en Hugging Face; cuando la fecha pública del propio fabricante es otra — los repositorios suelen crearse primero en privado — se dan ambas. La última columna es la pregunta que le importa a este boletín: si funciona hoy en llama.cpp upstream.

FECHA	MODELO	POR QUÉ IMPORTA
6 sep	MiniCPM5-2B	Apache 2.0, 2,5B denso, 128K de contexto, publicado con un draft model DSpark de 324M para speculative decoding model card. Creado una hora después de cerrar el número 04, así que pertenece aquí. La card reclama resultados de clase 4B en código, matemáticas y uso de herramientas. Funciona upstream , con GGUF oficiales para el modelo y el drafter GGUF.
10 sep	DeepSeek-V4.1-Flash	MIT, 763B almacenados con 8B activos en prefill y 16B en decode , 1M de contexto, imagen y texto model card. La publicación de la ventana. Un diseño encoder-decoder causal más reutilización de KV entre capas y KV cache en FP4 reducen la KV cache global a 890 bytes por token, más o menos una cuarta parte que V4-Flash 2609.19969, y sus tablas de memoria condicional "Engram" son búsquedas dispersas que pueden vivir en SSD — una build de la comunidad las lee del disco en un solo Mac de 128 GB GGUF. No está en llama.cpp upstream : la PR de soporte lleva abierta desde el día del lanzamiento #28696.
12 / 21 sep	AliceAI-Foundation-80B-A3B-Base	Apache 2.0, 81B totales con 3B activos , 262K de contexto, un modelo base de Yandex entrenado desde cero con una disposición híbrida de atención lineal KDA model card. Repositorio del 12 de septiembre, anuncio del 21 post. Solo base, así que es para quien hace fine-tuning; el más fuerte, según el fabricante, en QA factual en ruso. Todavía sin PR en llama.cpp.

FECHA	MODELO	POR QUÉ IMPORTA
16 sep	Xing4.0-29B-A4B	Apache 2.0 , 29B con 4B activos , 256K de contexto, MLA más MTP y, según la card, entrenado íntegramente en NPU Ascend <code>model card</code> . Reporta 75,0 en SWE-bench Verified; el GGUF IQ4_NL del fabricante ocupa unos 18 GB, una GPU de consumo. El soporte upstream es una PR abierta #29012; mientras tanto el fabricante distribuye su propia build de llama.cpp.
16 sep	Ternary-Bonsai-2-27B	Apache 2.0 , Qwen3.8-27B con pesos ternarios a 1,72 bits por peso de extremo a extremo , 5,95 GB frente a unos 54 GB en FP16 <code>model card</code> . La historia de compresión de la ventana, reportada al 98,2% de FP16 en 14 benchmarks en modo razonamiento. Lee la §2 antes de fiarte: el conocimiento es exactamente lo que, según las auditorías de la ventana, pierden los modelos de pocos bits mientras las medias aguantan. Necesita el fork de llama.cpp de Prism ; las PRs upstream están abiertas #29672.
17 / 22 sep	FrogNano-4B	Un agente de código de Microsoft de 4,7B sobre la arquitectura híbrida de Qwen3.5-4B, entrenado solo con RL sobre unas 1.500 tareas de software sintéticas y, según la card, sin trayectorias de ningún modelo más fuerte <code>model card</code> . La licencia es ambigua — los metadatos de la card dicen MIT, su tabla dice Apache 2.0 y el repositorio no tiene fichero de licencia. Funciona upstream con GGUF de la comunidad GGUF.
18 sep – 5 oct	Decision models estilo JEV	La respuesta abierta a Jev, el modelo alojado de TypeSafe (§3): modelos que devuelven una probabilidad calibrada por opción ante preguntas tipadas en lugar de generar texto, sobre el formato <code>/v1/systemone</code> de Jev. Laya (18 sep, Apache 2.0, 421M, el más votado de la ola) <code>model card</code> ; OpenJev (20 sep, 27B, CC BY-NC, reportado a menos de 2 puntos de Jev alojado) <code>model card</code> ; Kev-9B, lev e Intern-Decision-4B como adaptadores o fine-tunes de modelos Qwen; y JEV-27B-VL de AutoTrust (30 sep, Apache 2.0, Qwen3.8-27B más una cabeza de decisión, lee imágenes) <code>model card</code> . Laya, Julia-1, lev, Kev y OpenJev funcionan upstream desde la primera PR de decision models #29818.
20 sep	Mellum2.1-12B-A2.5B	Apache 2.0 , 12B con 2,5B activos , 128K de contexto; el modelo de código de JetBrains, con casi todo el trabajo nuevo en post-training con RL sobre repositorios reales <code>model card</code> . 2,5B activos es decode de clase CPU para un agente de código. Funciona upstream , con GGUF oficial GGUF.
21 sep	Familia MiMo-V2.6	MIT. Los MoE omni de Xiaomi: Flash con 309B y 15B activos, Pro con 1,02T y 42B activos, 1M de contexto <code>model card</code> . El que puede ejecutar casi todo el mundo es Distill-Qwen-9B , un SFT denso de 9,4B sobre Qwen3.5-9B que reporta 44,6 en SWE Pro frente a 32,0 de su base <code>model card</code> . Funciona upstream para texto #29257, con GGUF de ggml-org.

FECHA	MODELO	POR QUÉ IMPORTA
30 sep / 1 oct	Clef, Clef-Flash	Apache 2.0 , post-trains de 27B y 9B sobre modelos Qwen que no generan texto: envías un estado y un esquema de preguntas tipadas, y recibes una probabilidad calibrada para cada opción en una sola pasada <code>model card post</code> . El modelo estilo JEV de la empresa más grande hasta ahora. Funciona upstream en el nuevo endpoint <code>/v1/systemone</code> (§1) GGUF.
2 / 3 oct	Kolibri-1	Apache 2.0 , 78B con 3,5B activos , 262K de contexto nativo validado hasta 1M; el MoE europeo de pesos abiertos de Aleph Alpha, entrenado desde cero con 20T tokens, alrededor de una cuarta parte en alemán <code>model card post</code> . No está upstream ; los GGUF de la comunidad necesitan un parche incluido, y no hay PR.
5 / 7 oct	d1-3B, d1-omni-600M	Los decision models de Liquid AI para el edge, bajo la LFM Open License: la variante omni de 600M recibe texto, imágenes y hasta 30 s de audio, y el fabricante reporta 8 ms por decisión en una RTX 4090 <code>model card post</code> . Funciona upstream #30114.
14 sep / 6 oct	EmbeddingGemma 2	Apache 2.0 , 740M, un único espacio de 768 dimensiones para texto, código, imágenes, vídeo y audio, con el encoder de cada modalidad cargable por separado <code>model card post</code> . El repositorio es de septiembre pero el modelo se lanzó el 6 de octubre. La pieza de RAG local de la ventana, y soportado upstream el mismo día #30054.
6–8 oct	LightOnOCR-3	Apache 2.0 , modelos de OCR de extremo a extremo de 0,8B, 1B y 4B con un modo de grounding que devuelve cajas delimitadoras etiquetadas <code>model card post</code> . Los de 0,8B y 4B pasan a la arquitectura Qwen3.5-VL, así que deberían funcionar por la vía multimodal de llama.cpp — pero no hay GGUF oficial y no lo hemos verificado.

De catorce filas, ocho funcionan hoy en llama.cpp estándar — tres porque reutilizan una arquitectura que ya conocía, cinco porque el soporte llegó después del lanzamiento, los decision models entre ellas. Las tres publicaciones más interesantes — DeepSeek-V4.1-Flash, Ternary Bonsai 2 y Kolibri-1 — necesitan un fork, un parche o una PR abierta. Fuera a propósito: Qwen3.8, GLM-5.3 y Kimi-K3 siguen en tendencia pero se publicaron antes de la ventana; los grandes post-trains de bases abiertas existentes, como Nex-N2.5 e Intern-S2, son publicaciones reales pero quedan fuera de un boletín sobre lo que funciona en local; y Qwen-Image-2.1 queda fuera del ámbito de los LLM.

Qué haría yo

Ordenado por madurez, no por lo interesante que sea.

ADOPTAR

Actualiza llama.cpp — después de leer los tres cambios incompatibles, y hoy mismo si expones `rpc-server`. El puerto es ahora 9931 #30159, los flags de carga antiguos han desaparecido #28334 y las sesiones guardadas por builds anteriores no cargarán #28498. Nada de eso es razón para quedarse atrás: el arreglo del use-after-free de RPC está en la v0.5.0 y posteriores #24292, y Qwen3.8-Flash-Next estaba roto antes del 1 de octubre #29751. Pasa `--port 8080` explícitamente, cambia `--mmap` por `--load-mode` y regenera las sesiones guardadas.

ADOPTAR

En un Mac M1–M4, vuelve a activar el speculative decoding. Si este verano descartaste DFlash2 en Apple silicon, estabas midiendo un problema de kernel que ya está arreglado: de 30,2 a 110,0 t/s en código en un M3 Ultra, con el decode secuencial sin cambios #29869. En cualquier hardware, si muestreas con temperatura, prueba `--spec-draft-sampling probabilistic` con un draft simple o MTP #27694 — y déjalo desactivado cerca de temperatura cero, donde puede costar unos puntos porcentuales.

ADOPTAR

Deja de aceptar la perplejidad como control de un cuantizado que despliegas. Las auditorías de la ventana coinciden en que la perplejidad y la precisión media pasan mientras fallan las definiciones 2610.04403, la capacidad factual 2609.32902 y el uso de herramientas 2609.32042 — y en que el veredicto puede invertirse con el prompt y el criterio de puntuación 2610.07781. En concreto: por debajo de 2B parámetros, no bajas de Q4_K_M; a igual tamaño, prefiere Q4_K a Q4_0 2609.07664; y prueba el comportamiento del que dependes, con más de un prompt, sobre el artefacto exacto que publicas. Cuatro números seguidos con 4 bits o menos pasando el control estándar y fallando uno de comportamiento ya no es una coincidencia; trátalo como lo esperable.

VIGILAR

La caché de expertos MoE, hasta que llegue a un tag. Si ejecutas un MoE más grande que tu VRAM en una sola tarjeta de consumo para uso interactivo, `--moe-cache-mib` en una nightly merece una tarde: 1,57–2,20× de decode reportado #29887. Si procesas prompts largos repartidos en dos tarjetas, la propia tabla del autor muestra el procesamiento del prompt a 0,73–0,91× #30112, así que mide tu mezcla antes de quedártela.

VIGILAR

Los decision models estilo JEV, como sustituto de "generar y luego parsear". Jev, el modelo alojado de TypeSafe [post](#), la Decisions API de OpenAI [changelog](#), 6 oct, un endpoint en un runtime abierto con el mismo formato [#29818](#) y más de una docena de modelos abiertos llegaron en tres semanas. Para enrutado, moderación y elección de la siguiente acción de un agente, una sola pasada que devuelve una probabilidad por opción es más barata y más fácil de umbralizar que parsear texto, y lo abierto ya está cerca: OpenJev reporta un 84,0% frente al 85,4% de Jev alojado en su propia prueba [model card](#), aunque con licencia no comercial. Si quieres probarlo en local, Laya, con 421M y Apache 2.0, es el sitio barato para empezar [model card](#). El formato tiene semanas y ya hay abierta en llama.cpp una PR "Decision 2.0", así que construye un prototipo, no una dependencia — y mantén honestos los umbrales, porque una respuesta calibrada puede seguir siendo errónea.

VIGILAR

DeepSeek-V4.1-Flash y Ternary Bonsai 2, hasta que lleguen upstream. El primero es la arquitectura de la ventana y el segundo un 27B en 6 GB, pero ambos necesitan un fork o una PR abierta ([#28696](#), [#29672](#)). En el caso de Bonsai, lee su "98% de FP16" junto a la §2: una media entre benchmarks es exactamente la métrica que, según las auditorías de la ventana, oculta lo que pierden los modelos ternarios. Si haces fine-tuning de un modelo ternario, revisa la exportación: convertir a bf16 antes de ternarizar puede revertir el fine-tune en silencio [2610.07853](#).

VIGILAR

Si destilas para añadir conocimiento, OPD con KL inversa es la herramienta equivocada. Afila las habilidades que el student ya tiene [2610.09639](#) [2609.35210](#), y las versiones baratas igualan a las caras [2609.24432](#). Para hechos, el único resultado controlado dice que la KL directa restaura la transferencia; hasta que se replique, mete el conocimiento nuevo con SFT o recuperación y usa OPD para ajustar el comportamiento. Ya puestos, revisa tus tokens EOS [2609.20511](#).

SOLO LABORATORIO

Modelos en bucle. Su cuota de papers se multiplicó por seis a lo largo de la ventana, y los trucos para servirlos son reales — KV compartida entre bucles [2610.02383](#), auto-especulación [2609.23033](#) — pero hay pocos checkpoints abiertos en bucle que merezca la pena ejecutar, el soporte en llama.cpp acaba de empezar con HRM-Text [#27625](#), y el único estudio controlado de lo que sacrifican dice que el conocimiento [2609.36636](#).

Qué establecieron cinco semanas y qué no. Dos tendencias sobreviven a normalizar por volumen: la cuota de la on-policy distillation en los envíos a cs.CL y cs.LG aproximadamente se triplicó y se asentó en torno al doble de donde empezó, y los modelos en bucle subieron unas seis veces desde una base pequeña y seguían subiendo. Los recuentos brutos del 27 de septiembre al 4 de octubre están inflados en todos los temas — los envíos totales de esa semana fueron más o menos el doble que en las demás — y no deben leerse como crecimiento. La tesis se apoya de forma desigual: la pata de OPD tiene detrás cuatro grupos independientes, la de cuantización varias auditorías mayoritariamente a 14B o menos, y la de los modelos en bucle un solo paper. Sin establecer: si "OPD no añade conocimiento" se sostiene a escala con KL directa, si el daño léxico de los GGUF se replica más allá de palabras raras del inglés, y si el speculative decoding "sin pérdidas" falla la coincidencia exacta en bf16 con drafters distintos del probado. Un mes basta para ver convergencia; no basta para ver replicación.

Fuentes

LLAMA.CPP §1

Las pull requests fusionadas entre el mediodía del 6 de septiembre y el 10 de octubre se enumeraron con la API de GitHub (718 fusionadas, builds `b10822` – `b11540`), y cada PR citada abajo se consultó individualmente para confirmar que se fusionó; las PRs e issues abiertas se marcan como tales.

[llama.cpp v0.4.1](#) (notas de la release)

[llama.cpp v0.5.0](#) (notas de la release)

[llama.cpp v0.6.0](#) (notas de la release)

[#29887 llama](#) : add a GPU cache for MoE experts kept in host memory

[#30112 llama](#): support MoE cache over multiple GPUs

[Niko1221/Strata](#)

[#29869 metal](#) : few-row MMA mat-mul and batched copies for speculative decoding

[#30065 metal](#) : few-row MMA mat-mul for the remaining src0 types

[#27694](#) Make the drafter probabilistic and the target verify by rejection sampling for simple draft and MTP

[#29924](#) spec : fix n-gram drafts rejected at temp > 0 after truncation

[#27773](#) add GLM-5.3-Flash (GLM5-Next) support

[#29928](#) feat: add GLM5Next MTP, optimize

[#29751](#) llama: fix qwen4exp

[#29761](#) Qwen4Exp: add MTP

[#28068](#) models: fix GDN normalization from `max` to `rsqrt`

[#29818](#) llama, server: add /v1/systemone API (models: layla, julia-1, lev, openjev, kev)

[#29831](#) model: add support for clef decision model (text-only)

[#30044](#) models: support pplx-decider

[#30110](#) model : add LiquidAI/d1-3B decision model

[#27000](#) llama: add Maple 20B-A1B ternary MoE architecture (CPU)

[#27625](#) model : add support for HrmTextForCausalLM (DFM Mimir 1B)

[#28105](#) vulkan: support sparse Flash Attention

[#28796](#) [SYCL] support sparse FA

[#28770](#) CUDA: enable sparse fa for qwen4

[#29639](#) vulkan: sparse flash attention for quantized K/V

[#29298](#) CUDA: enable sparse-fa for dsv4 prefill (again)

[#29570](#) metal : add tensor API flash attention kernel for F16 KV

[#29435](#) CUDA: prefer whole-tile FlashAttention scheduling for efficient two-stage kernels

[#28102](#) CUDA/HIP: Flash Attention tuning (gfx1201)

[#27851](#) ggml-cpu: tiled mul_mat for k-quants

[#27952](#) vulkan: int8 coopmat1 matmul implementation for AMD RDNA3 and RDNA4

[#29809](#) sycl : accelerate MXFP4 MoE with arithmetic decoding and weight reordering

[#29749](#) Avoid a second full-size copy of each tensor with direct-io

[#29007](#) ci: publish snapdragon builds in release workflow

[#30159](#) server: use port 9931 by default

[#28334](#) args: remove mmap/mlock/dio flags from arg parser

[#28498](#) kv-cache: fix restoring mismatched KV cache rotation by saving exact rotation metadata

[#26004](#) server : preserve context checkpoints across slot save/restore

[#28690](#) server: Add support for binding to multiple addresses

[#28876](#) server : allow RANK pooling batch splitting for causal LLM rerankers (ie. Qwen3 and Qwen3-VL)

[#29583](#) ui : add the models manager

[#24292](#) rpc : fix UAF in graph_recompute leading to remote code execution

[#30135](#) ggml: fix OOB write in ggml_acc / ggml_set with negative offset

[#29598](#) ggml : speed up model loading

[#29384](#) Ggml integer overflow

[#28849](#) Change max context length for auto-fitting with unified KV

[#29437](#) Revert "Change max context length for auto-fitting with unified KV"

[#29182](#) vulkan: MOE aware mul_mat_id tile selection

[#29936](#) vulkan: revert mul_mat_id tile selection PR [#29182](#)

[#29770](#) metal : use bf16 math for mxfp4 mul-mat

[#30042](#) llama: remove the gather path of the glm5-next sparse attention

[#27428](#) draft-mtp roughly halves prompt processing on multi-GPU layer split (issue [abierta](#))

[#28827](#) gemma4 thinking emits progressively long trailing garbage (issue [abierta](#))

PAPERS §2

[2610.09639](#) On-Policy Distillation Teaches New Skills but Not New Knowledge

[2609.35210](#) Understanding On-Policy Distillation: A Mechanistic Interpretability Perspective via Sparse Crosscoders

[2610.03185](#) Gains and Collapse in On-Policy Distillation:A Reinforcement Learning Perspective

[2610.11247](#) Why On-Policy Distillation Sometimes Fails: Vanishing Learning Signals

[2609.33791](#) Do We Really Need KL Divergence for On-Policy Distillation of Large Language Models?

[2609.35259](#) On-Policy or Off-Policy Learning? A Systematic Study of Distillation Dynamics

[2609.32722](#) Scaling Properties of Same-Family On-Policy Distillation

[2609.35347](#) Beyond Teacher Assignment: Domain-Normalized Multi-Teacher On-Policy Distillation

[2609.24432](#) 1% of Tokens Can Be Enough: On Gradient Estimation in On-Policy Distillation

[2610.08448](#) Rethinking Cross-Tokenizer On-Policy Distillation: From Alignment Coverage to Supervision Reliability

[2609.34386](#) Look Before You Select: Rethinking Vocabulary Sparsification in On-Policy Distillation

[2609.20511](#) When EOS Tokens Disagree: Understanding Length Inflation in On-Policy Distillation

[2609.38995](#) When Clipping Reverses Correction: Failure Dynamics of Pointwise Forward-KL On-Policy Self-Distillation

[2609.36484](#) The Teacher Is a Direction, Not a Destination: Extrapolating RL-Induced Representation Residuals in On-Policy Distillation

[2610.04403](#) Saying, Not Knowing: Aggressively GGUF-Quantized Small Language Models Still Write Rare Words They Can No Longer Define

[2609.32902](#) Linger and Lose: Knowledge Collapse in Low-Bit Language Models

[2609.07664](#) Accuracy is Not Enough: A Divergence-Based Approach to Evaluate Fidelity Loss in Quantized LLMs

[2609.32042](#) Quantization Thresholds Replicate, Failure Modes Do Not: A Three-Model Study of Agentic Tool Use in Polish from 8-bit to 2-bit

[2610.07781](#) Quantization Effects on Tool-Failure Recovery Vary Across Prompts and Evaluation Designs

[2610.11226](#) When Lower Reconstruction Loss Hurts: Distributionally Robust Refinement for Low-Bit LLM Quantization

[2609.34866](#) From Attention Sensitivity to Layer Role: Revisiting Mixed-Precision Quantization of Transformers

[2610.07853](#) Lost in the bf16 Cast: Exporting Ternary Language Models Can Revert Most Low-Learning-Rate Code Changes

[2609.26333](#) Disaggregated Quantization: Specializing LLM Prefill and Decode

[2609.39223](#) QATFactory: A Versatile, Deployment-Aligned Framework for Quantization-aware Training and Distillation of LLMs

[2609.35465](#) Tetra: Serving Leech-Lattice Quantized LLMs at 2.7 Bits per Parameter

[2609.38169](#) STEPQuant: When and Where Errors Matter in Delta-Rule Recurrent State Quantization

[2609.40316](#) Scaling Laws for Looped Mixture of Experts

[2610.02383](#) The Surprising Effectiveness of Shared Memory in Looped Transformers

[2609.23033](#) WaveFront Decoding: Parallelized Self-Speculative Decoding for Looped Language Models

[2609.39277](#) A Tilted Bowl Is Not a Slippery Slope: Compressing Looped Models

[2609.36636](#) What Makes Recurrence Effective in Looped Language Models?

[2609.15504](#) Evaluating Losslessness in Speculative Decoding Under Finite-Precision Inference

[2609.36322](#) Periodic Weak Spots: Phase Sensitivity from Chunked KV-Cache Compression

[2609.31415](#) Evaluating the accuracy of KV cache reuse techniques

[2609.18063](#) The Other Half of the Memory Wall: Serving 35B MoEs from SSD with Trained Routing Prediction

[2610.07659](#) DLoop: Looped Speculative Decoding

[2610.01108](#) AgSpec: Pushing the Limits of Retrieval-Based Speculative Decoding in Coding Agent Pipelines

LABORATORIOS §3

[TypeSafe — Introducing System One Models & Jev \(launched 15 Sep; post now timestamped 9 oct\)](#)

[VKTR — TypeSafe launch coverage \(16 sep\), cited for the launch date](#)

[convaiinnovations/laya](#)

[openjev/openjev](#)

[IFM/K2-Type-0.9B](#)

[autotrust/JEV-27B-VL](#)

[ggml-org — decision models en llama.cpp \(2 oct\)](#)

[Changelog de la API de OpenAI \(entradas del 8 sep al 8 oct\)](#)

[Anthropic — Claude Sonnet 5.5 \(28 sep\)](#)

[xAI — Grok 4.7 \(21 sep\)](#)

[Anthropic — Claude Haiku 5.5 \(7 oct\)](#)

[Anthropic — Claude Opus 5.5 \(22 sep\)](#)

[OpenAI — GPT-6 and Intelligent UI for everyone \(7 oct\)](#)

[Cognition — SWE-2 \(10 sep\)](#)

[Claude API](#)

[OpenAI — ficha del modelo GPT-6.1 Sol](#)

[Anthropic — Cyber Verification Program \(6 oct\)](#)

[Anthropic — Life Sciences Verification Program \(17 sep\)](#)

[Google — Gemini 4 Argon \(30 sep\)](#)

[OpenAI — Introducing Dots \(29 sep\)](#)

[zai-org/ZCode](#)

[Changelog de la API de Gemini](#)

[Google — modelos locales en el SDK de Antigravity \(23 sep\)](#)

[Mistral — Mistral Large 4 \(6 oct\)](#)

[Anthropic — sistema enzimático descubierto por agentes \(23 sep\)](#)

[Google DeepMind — AlphaGenome Atlas \(8 sep\)](#)

[Google DeepMind — SynthID Bio \(30 sep\)](#)

[OpenAI — Sharing AI progress in mathematics \(6 oct\)](#)

[NVIDIA — Nemotron en la IOI y la IMO 2026 \(7 oct\)](#)

[Hugging Face — Transformers ejecuta cuantizados de llama.cpp \(22 sep\)](#)

PUBLICACIONES §4

[openbmb/MiniCPM5-2B](#)

[openbmb/MiniCPM5-2B-GGUF](#)

[deepseek-ai/DeepSeek-V4.1-Flash](#)

[2609.19969 DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression](#)

[antirez/deepseek-v4.1-flash-gguf](#)

[#28696 model : add DeepSeek V4.1 \(abierta\)](#)

[yandex/AliceAI-Foundation-80B-A3B-Base](#)

[habr.com](#)

[XingChen-AGI/Xing4.0-29B-A4B](#)

[#29012 Add Support for Xing4.0 \(abierta\)](#)

[prism-ml/Ternary-Bonsai-2-27B-gguf](#)

[#29672 ggml: add PTQ1_0, ternary at group 128 \(abierta\)](#)

[microsoft/FrogNano-4B-2609](#)

[bartowski/FrogNano-4B-2609-GGUF](#)

[JetBrains/Mellum2.1-12B-A2.5B-Thinking](#)

[JetBrains/Mellum2.1-12B-A2.5B-Thinking-GGUF](#)

[XiaomiMiMo/MiMo-V2.6-Flash-RL](#)

[XiaomiMiMo/MiMo-V2.6-Distill-Qwen-9B](#)

[#29257 convert: add MiMo-V2.6 support](#)

[Cloudflare/clef](#)

[blog.cloudflare.com](#)

[ggml-org/Clef-GGUF](#)

[Aleph-Alpha/Kolibri-1](#)

aleph-alpha.com

LiquidAI/d1-omni-600M

www.liquid.ai

[#30114](#) model : add LiquidAI/d1-omni-600M decision model

google/embeddinggemma-2

blog.google

[#30054](#) model: support embeddinggemma2 (text+vision+audio)

lightonai/LightOnOCR-3-4B

huggingface.co

Compilado el 10 de octubre de 2026 a partir del feed de releases y la API de GitHub de llama.cpp, la API de arXiv filtrada a esta ventana y recorrida semana a semana, los daily papers curados por Hugging Face, la API de modelos y las model cards de Hugging Face, y las salas de prensa y changelogs de los propios laboratorios. Las cifras de aceleración, puntuaciones de benchmark o tasas de error son las reportadas por los autores o fabricantes en sus propios benchmarks — trátalas como afirmaciones, no como mediciones, hasta que las reproduzcamos. La ventana empieza donde terminó la compilación del número 04, el mediodía del domingo 6 de septiembre, y cierra con la de este, el sábado 10 de octubre; todo lo fechado fuera de ella se marca en línea como contexto. Los términos se definen en la [página del glosario](#). Corregido más tarde el 10 de octubre para nombrar Jev, de TypeSafe, y los modelos estilo JEV, que la primera versión cubría solo como decision models genéricos, y para añadir Gemini 4 Argon a los puntos de arriba; y de nuevo tras una comprobación de cobertura contra Hacker News, Hugging Face y GitHub, para añadir seis elementos de la ventana que esta sacó a la luz.



LLM Radar lo publica semanalmente [Montevive](#) · [Feed Atom](#) · ai@montevive.ai · [Privacidad](#)