Engrams Embedding Entendre: Codesign for Efficient DRAM/SSD Offloading

New Model Architecture Implications for TAM of DRAM/NVMe, DeepSeek V4.1 Flash, AgentX, InferenceX, NVMe experiments

BRYAN SHAN, CAM QUILICI, ALEC IBARRA, AND 3 OTHERS

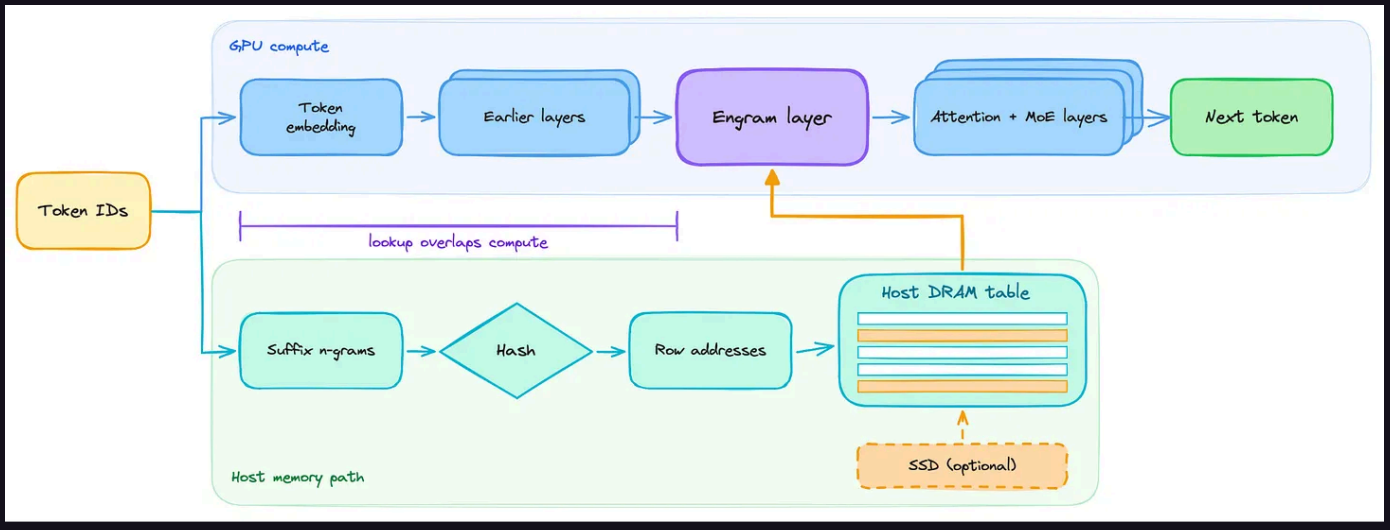
SEP 18, 2026 • PAID

Share

Engram extends standard token embeddings with learned multi-token lookups. Recurring local patterns retrieve vectors directly, reducing the need to reconstruct them through attention and feed-forward layers.

With Engram model architecture optimization, it allows for lower HBM capacity to be needed for models at the same quality. This does not mean there won't be an insane demand for HBM but it just means that model architecture will continue to innovate around constraints.

This model architecture design is naturally codesigned for parameter offloading: each token accesses a few embedding rows whose addresses depend on token IDs, not hidden states. The runtime can prefetch those rows from host DRAM while earlier layers compute, keeping the table outside HBM without transferring entire weight matrices. Our Memory model contains our latest estimates of quarter by quarter HBM, DRAM, & NAND supply and demand.



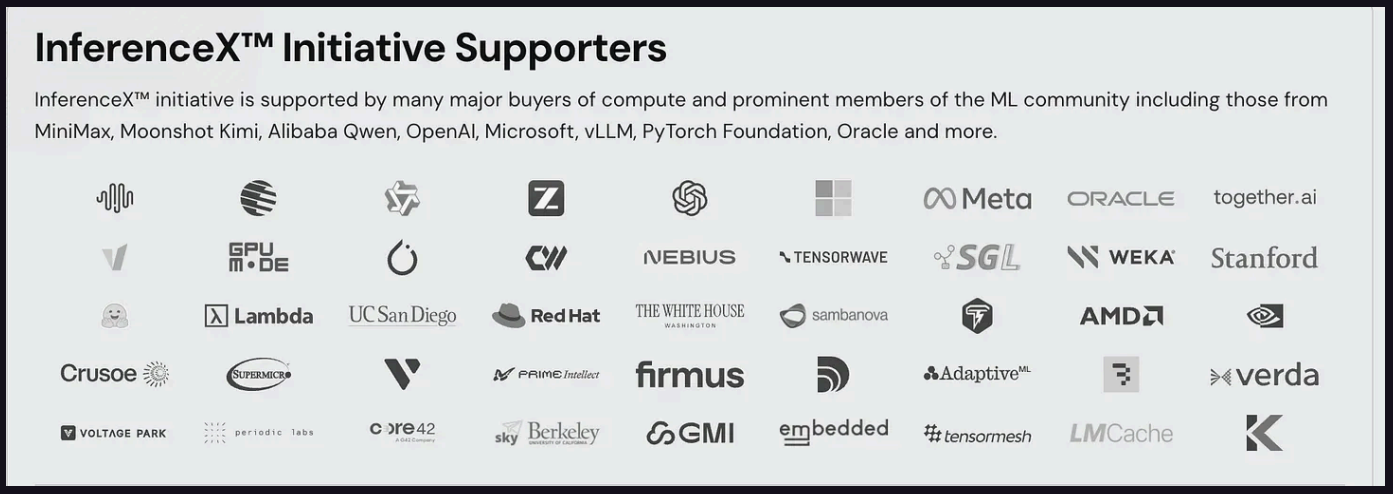
Offloading frees HBM for model weights and KV cache, potentially supporting larger batches or more concurrent sessions. When DRAM becomes the next constraint, NVMe offers another tier. Recommendation systems already cache frequently or recently accessed embedding rows in faster memory while backing colder rows with SSDs. After NVIDIA roadmap had to change due to massively despec'ing Rubin Ultra from 1024GB to now ~200GB of HBM per chip, model architecture optimizations like emgram maybe helpful.

Our DeepSeek-V4.1-Flash configuration uses roughly 189 GiB of memory for Engram. We replace it with a memory-mapped (mmap) file and measure serving performance with offload to SSD.

Later on in our report, we will show our Engram offloading experiments along with the official InferenceX agentic inference serving results on Engram models like DeepSeekv4.1 Flash across all 6 NVIDIA GPU SKUs along with MI355X. Unsurprisingly, the CUDA Moat is still mogging MI355X on the ultra popular DeepSeekV4.1 Flash model.

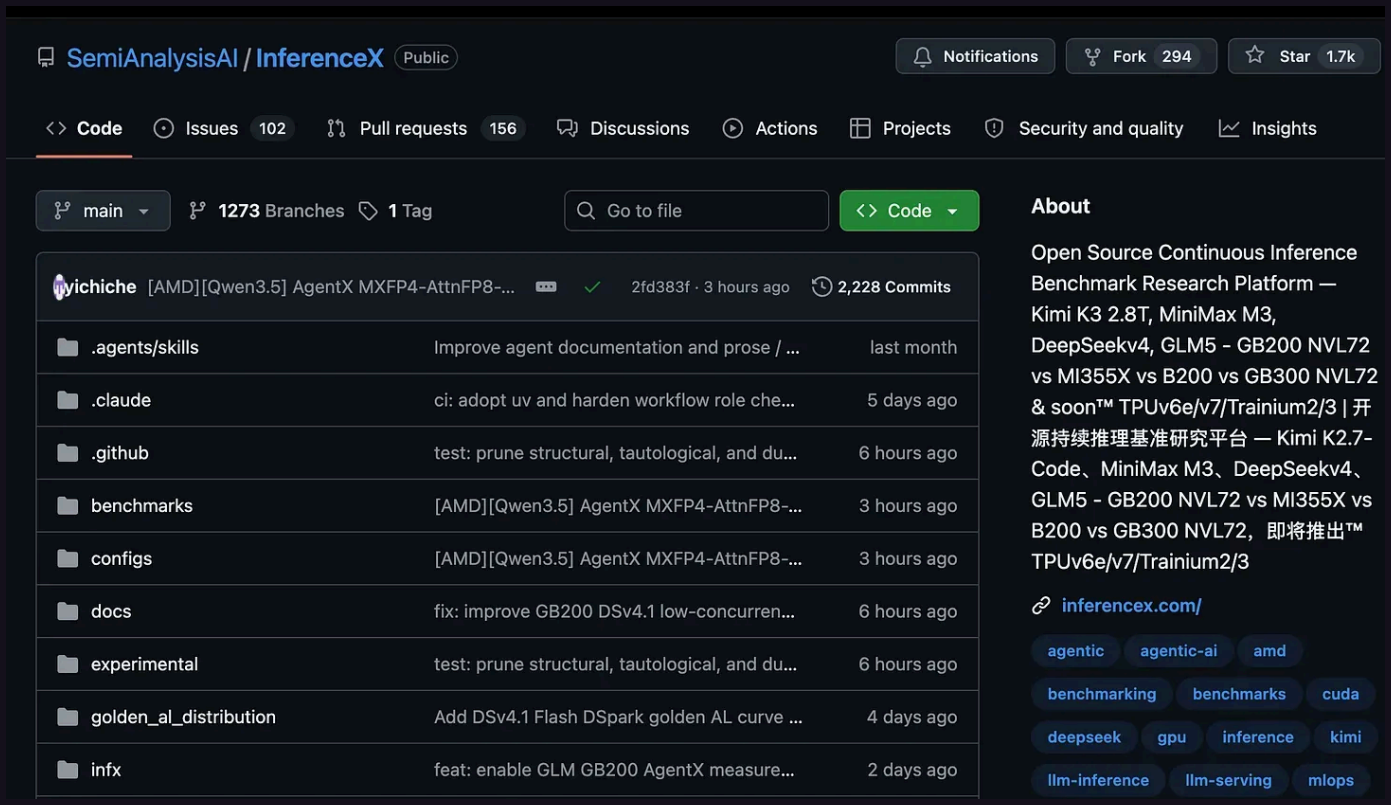
We also show how even on high capacity HBM SKUs, offloading emgrams to DRAM could result in even better performance for most of the pareto than keeping the emgram in HBM.

Our benchmark has been widely reproduced, validated and/or supported by almost every major buyer of compute from Google Cloud to Microsoft Azure to Oracle, to Meta and many more. Furthermore, it has the support of the ML community including from vLLM, LMCache, SGLang, PyTorch, Huggingface and the support of major labs like OpenAI, MiniMax, ZAI, Qwen, Moonshot Kimi, etc.



Source: InferenceX

Nvidia Rubin NVL72, AMD, and soon, SambaNova and Trainium. Due to how realistic AgentX scenario is to real world agentic inference workloads, AMD has committed to collaborating on MI455X UALoE72 too.



Source: SemiAnalysis

Engram Benefits

DeepSeek did not release the original paper's two trained Engram models. We replicated its setup on fineweb-edu using the released code and training hyperparameters, at an estimated 6E18 FLOPs per run. We observed the same U-shape scaling:

Source: SemiAnalysis

Engram improved performance over pure MoE baselines. We also reproduced DeepSeek's results, where earlier-layer representations with Engram resembled those of later layers.

Source: SemiAnalysis

What is Being Memorized?

Like the original Engram paper, it is possible to probe Engram's gate scores to see what n-grams DeepSeek-V4.1-Flash makes the most use of. Our gate scan finds names, code fragments, relational phrasing, and boilerplate. These examples prioritize interesting-ness over gate strength.

One unexpected result was `Wright : Ace Attorney`.

Source: SemiAnalysis

These examples suggest learned memory optimizes the training objective, not a judgment of which facts deserve storage. Licenses, bibliography fragments, API scaffolding, and website furniture can provide prediction shortcuts, so the value of additional Engram capacity may depend on what survives data preparation. This does not show that table capacity is "wasted": the evaluation-corpus scan establishes neither training exposure nor the capacity occupied by each category.

For offloading, strong gates do not identify cache-hot rows. Low gates do not automatically save reads either: computing the gate requires the retrieved key and undoes the performance gain of a fused kernel. Skipping reads would require a separate usefulness predictor before retrieval.

Removing Engram

This is due to the training–inference mismatch. The resulting degradation therefore measures this trained model's dependence on Engram, not the performance difference between models trained with and without it.

Source: DeepSeek

In our ablations, suppressing Engram worsens token likelihood across all evaluated domains, especially encyclopedia text and several code corpora. Surprisingly, GSM8K accuracy stays within measured run-to-run variation and removing Engram has no effect.

Engram is not a detachable dictionary beside an unchanged MoE. Removing it changes downstream features and expert selection.

We tested whether rerouting hurts or compensates by holding tokens fixed in a teacher-forced experiment on CRUXEval, a code-reasoning benchmark of small Python functions where the model predicts a function's output from its code and an input, and scoring the reference answer.

Source: CRUXEval

Removing Engram raised answer loss from 0.2848 to 0.3093 bits/token. Forcing the ablated model to use the original Engram-on expert choices made it worse still, at 0.3375 bits/token.

Source: SemiAnalysis

Rerouting partially compensates for the missing memory. Memory features and expert selection work together, rather than following a clean “memory stores facts; experts reason” division.

On the same CRUXeval, removing Engram during either phase reduced accuracy and increased generated tokens; removing it throughout produced the largest changes.

Source: SemiAnalysis

Keeping Engram for prefill leads to more correct answers than keeping for decode likely due to semantically richer KV cache transferred to decode workers, allowing it to mitigate some of the performance loss.

Agentic Inference Serving Performance

Engram's table is large, but each lookup is small. DeepSeek-V4.1-Flash requests 24 rows at each of two Engram layers, about 12.4 KiB per processed token position across the model, or 3.1 KiB per GPU when split across four GPUs.

Source: InferenceX

Currently as of Day 7 since Model Release, MI355X is still 2-4x worse performance per dollar compared to B200 even when normalized by Mi355X's lower TCO. Our full total cost of ownership breakdown comes from our AI Cloud TCO Model along with monthly market surveys of over 100+ gpu clouds & gpu cloud customers.

Source: InferenceX

On the Day 0 release of DeepSeekv4.1 Flash, NVIDIA vLLM works out of the box with zero issues across all 6 SKUs: H100, H200, B200, B300, GB200, GB300! This was thanks to the amazing work by the NVIDIA & Interact teams! In comparison, AMD vLLM did not work on day 0 for DeepSeekv4.1 Flash.Source: SemiAnalysis InferenceX

Source: InferenceX

AMD's vLLM documentation points to using vllm/vllm-openai-rocm:deepseekv41-flash-0909, but from hour 0 of the model release to hour 23, AMD has not publicly released the image. AMD claims, "SPEED IS THE MOAT," yet it has still not released it by the 23rd hour. We wish that, going forward, the AMD team has a better process for hour 0 model releases.

Source: vLLM AMD

Source: vLLM, AMD, DockerHub

than B200/B300.

The power of the CUDA MOAT is NVIDIA's collaboration with its massive 6 million-developer community ecosystem including most of the vLLM & SGLang & Tokenspeed maintainers which means that CUDA is optimized on day 0.

Source: InferenceX

Overall, AMD did make significant improvements but the performance per dollar is still currently 2-4x worse than B200.

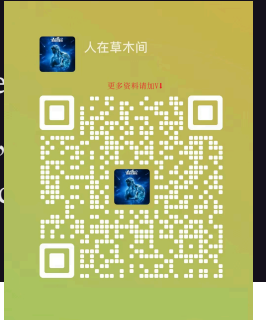
Source: InferenceX

AgentX Engram DRAM Offloading Improving Performance

The HBM and DRAM offload use the same GPU kernel to select and dequantize rows. With HBM, it reads GPU memory; with Unified Virtual Addressing (UVA), it reads pinned host memory directly.

Both support full decode graphs. Moving the table into HBM accelerates only the sparse lookup, leaving decoder computation and communication unchanged, resulting in little overall benefit while consuming memory otherwise available to KV cache.

Another benefit of Engram offloaded to DRAM which means you can reduce communication overhead by using less HBM GPUs per replica. For example, Engram offloading on B300, we are able to switch from TP4 to now TP2 whic the pareto curve by up to 1.6x.



Source: SemiAnalysis InferenceX

When iso-model quality, less HBM is required as the engrams could be offloaded to host DRAM. Thus HBM bandwidth matters way more than HBM capacity. For inference workloads where memory bandwidth matters the most, 4-hi HBM provides the best $/bandwidth and therefore lowest cost per token. If China continues to make more and more revolutionary model architecture innovations, soon it could potentially 0Hi HBM stacks.



Long Live the Short King: Why 4-hi HBM Wins

MYRON XIE, BRYAN SHAN, AND 3 OTHERS・9月14日

Read full story →

Moreover, on B300 and week-0 stack, moving the Engram table back to HBM did not improve results and stay within run-to-run variance. This is a result of the work optimizing DRAM offload, such as async, and overlap.

SSD Offloading

On B200, we created a unoptimized vLLM fork and stored the Engram tables in memory-mapped files on local SSDs. File backing lets the OS reclaim table pages when other

The unoptimized SSD implementation changes how rows reach the GPU. It copies row IDs to the CPU, deduplicates them, gathers the requested rows into pinned buffers, copies those rows back to the GPU and dequantizes them. This work runs between segments of the GPU execution graph. Native UVA performs row selection and dequantization directly on the GPU, avoiding the CPU round trip.

A warm filesystem cache removes physical SSD reads, but leaves the coordination, row gathering and transfers. This is why a file already cached in RAM can still perform worse than a pinned DRAM table. The comparison measures the whole serving path; it does not separate the time spent on each of these operations.

B200 DRAM dominates both measured SSD serving curves in total tokens per dollar and P90 interactivity. Near 125 tokens/s/user, DRAM delivers 121 million total tokens per dollar versus 52 million for SSD.

Source: SemiAnalysis InferenceX

For production serving, SSD offloading is likely not worth the tradeoff. On the B200 configurations we measured, SSD offloading loses on both measures: every observed SSD point has a DRAM alternative that delivers higher P90 interactivity and more total tokens per dollar. The only points where

Cheaper storage does not automatically produce a cheaper inference service. Moving Engram to SSD leaves the same four expensive GPUs and the rest of the server in place. Reclaiming RAM only creates an economic benefit if it enables a cheaper server configuration or additional useful capacity. The current unoptimized path provides neither benefit, and the filesystem cache still consumes RAM when table pages are resident.

Mechanism



semianalysis

This post is for paid subscribers

Subscribe

Already a paid subscriber? Sign in

← Previous

© 2026 Dylan Patel · Privacy · Terms · Collection notice

Start your Substack

Get the app

Substack is the home for great culture