DeepSeek AI Research

Engram

Conditional Memory via Scalable Lookup

A New Axis of Sparsity for Large Language Models

Primary Innovation

O(1) Memory Lookup

Model Scale

Up to 40B Parameters

Performance Gain

BBH +5.0, MMLU +3.0

The Problem with Current LLMs

Inefficient Knowledge Retrieval in Transformers

The Fundamental Limitation

Transformers lack a native knowledge lookup primitive. Unlike biological memory systems that can retrieve stored information via constant-time lookups, current LLMs are forced to simulate retrieval through expensive computation.

The Linguistic Duality

1. **Compositional Reasoning**
   Deep, dynamic computation for logic and inference
2. **Knowledge Retrieval**
   Static, local patterns (named entities, formulas)

The Inefficiency

To recognize "Diana, Princess of Wales", an LLM consumes **6+ layers** of attention and FFNs, progressively composing features. This amounts to an **expensive runtime reconstruction** of what should be a static lookup table.

Example: Entity Recognition Waste

L1-2	Wales → Country in the UK
L3	Princess of Wales (unspecific)
L6	**Diana, Princess of Wales**

Table: Diana, Princess of Wales resolution

The Solution Space

Current (MoE): Conditional computation sparsely activates parameters

Proposed (Engram): Conditional memory via sparse lookup operations

Wasted Depth
6+
layers

Desired
O(1)
lookup

Engram: A New Paradigm

Conditional Memory as a Complementary Sparsity Axis

Core Concept

Engram is a conditional memory module that modernizes classic N-gram embeddings for O(1) lookup, serving as a complementary axis of sparsity to MoE's conditional computation.

While **MoE** sparsely activates parameters for dynamic logic, **Engram** uses sparse lookups to retrieve static embeddings for fixed knowledge.

Key Design Principles

Structural Separation
Static pattern storage decoupled from dynamic computation

Constant-Time Access
O(1) retrieval via hashing, independent of memory size

Deep Injection
Strategic placement at specific layers (not just input)

Tokenizer Compression
23%
vocab reduction

Hash Heads
K=8
per N-gram order

Innovation Stack

1. Tokenizer Compression
Collapses semantically equivalent tokens via vocabulary projection

2. Multi-Head Hashing
K distinct hash heads mitigate collisions, concatenate embeddings

3. Context-Aware Gating
Hidden state queries memory, suppresses noise via scalar gate α_t

4. Multi-Branch Integration
Branch-specific gating with shared embedding table

5. Depthwise Convolution
Expands receptive field, enhances non-linearity

Mathematical Formulation

$e_{t,n,k} = E_{n,k}[\varphi_{n,k}(g_{t,n})]$
Hash retrieval for N-gram context

$\alpha_t = \sigma(\mathrm{RMSNorm}(h_t) \cdot \mathrm{RMSNorm}(k_t)/\sqrt{d})$
Context-aware gating

Architecture Overview

Two-Phase Processing: Retrieval and Fusion

Phase 1: Sparse Retrieval

Maps local contexts to static memory entries via deterministic hashing

Step 1: Tokenizer Compression

Raw token ID x_t → Canonical ID x'_t via surjective projection P

$$x'_t = P(x_t)$$

Step 2: Multi-Head Hashing

K hash heads map compressed N-gram to embedding table indices

$$z_{t,n,k} = \varphi_{n,k}(g_{t,n})$$
$$e_{t,n,k} = E_{n,k}[z_{t,n,k}]$$

Lightweight multiplicative-XOR hash function

Phase 2: Context-Aware Fusion

Dynamically modulates and integrates retrieved memory

Step 1: Context-Aware Gating

Hidden state h_t serves as Query; memory e_t as Key/Value

$$k_t = W_K e_t,\ v_t = W_V e_t$$
$$\alpha_t = \sigma(\mathrm{RMSNorm}(h_t) \cdot \mathrm{RMSNorm}(k_t)/\sqrt{d})$$
$$\tilde{v}_t = \alpha_t \cdot v_t$$

Gate $\alpha_t \to 0$ if memory contradicts context (noise suppression)

Step 2: Convolution & Residual

Depthwise causal convolution with SiLU activation

$$Y = \mathrm{SiLU}(\mathrm{Conv1D}(\mathrm{RMSNorm}(\tilde{V}))) + \tilde{V}$$
$$H^{(\ell)} \leftarrow H^{(\ell)} + Y$$

System-Level Design

Training

Tables sharded across GPUs. All-to-All comms

Inference

Offloaded to host memory. Prefetch & overlap

Module Placement Strategy

Engram is **not** applied to every layer. Optimal placement balances:

Early injection: Offload local patterns before backbone waste

Context quality: Allow attention to aggregate global context first

Engram-27B uses layers 2 and 15

System Efficiency

Decoupling Compute and Memory

Core Advantage: Deterministic Access

Unlike MoE's dynamic routing, Engram's retrieval indices depend solely on the input token sequence. This determinism enables speciali zed optimization strategies.

Training Phase

Embedding tables sharded across GPUs. All-to-All comms to gather acti ve rows

Inference Phase

Prefetch-and-overlap via PCIe from host memory. Computation of prece ding blocks masks latency

Multi-Level Cache Hierarchy

Exploits Zipfian distribution of N-grams: small fraction accounts for majority of accesses

Tier 1 **GPU HBM**	Frequently accessed embeddings (hot patterns)
Tier 2 **Host DRAM**	Medium-frequency patterns
Tier 3 **NVMe SSD**	Long tail of rare patterns (high capacity)

Throughput Penalty
<3%
with 100B params

Communication
O(1)
per token

Scalability
Linear
with GPU count

Inference Benchmark

Hardware Setup

NVIDIA H800, 512 sequences
SeqLen ~ Uniform(100, 1024)

4B Dense Backbone

Baseline: 9,031.62 tok/s
+100B Engram: 8,858.28 tok/s
(1.9% overhead)

8B Dense Backbone

Baseline: 6,315.52 tok/s
+100B Engram: 6,140.02 tok/s
(2.8% overhead)

Placement Trade-Off

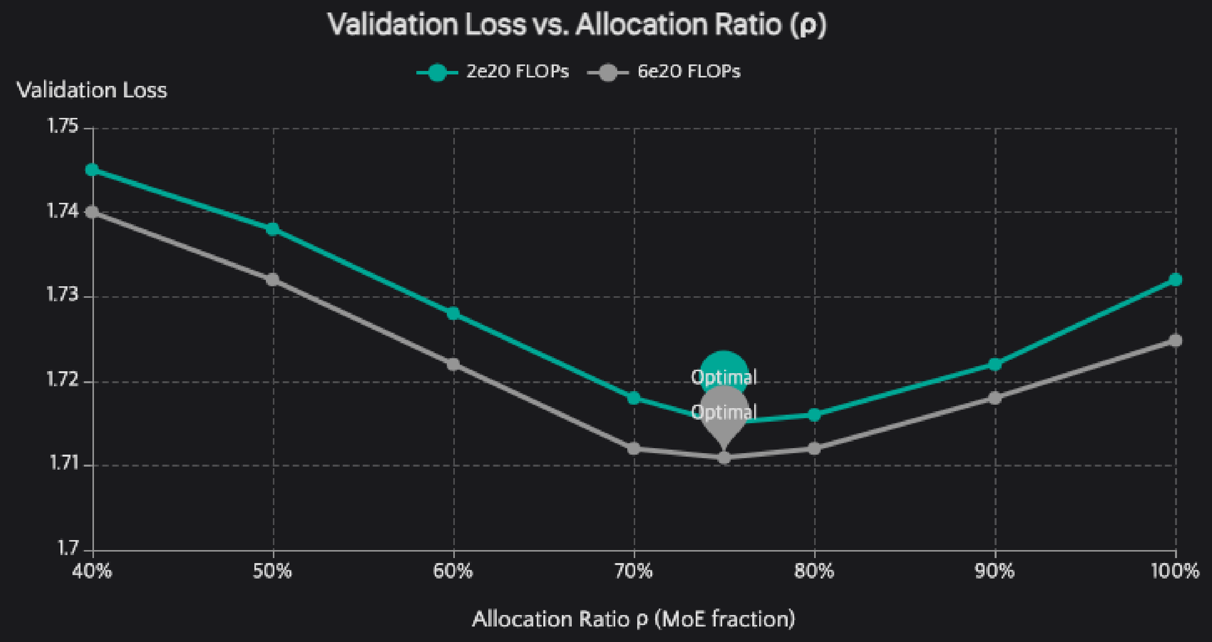
Optimal placement must satisfy both:

Modeling: Early intervention to offload local patterns

System: Sufficient compute window for prefetching

Sparsity Allocation

A U-Shaped Scaling Law



The Sparsity Allocation Problem

Given fixed P_{tot} and P_{act}, how to distribute inactive parameters P_{sparse} between MoE experts and Engram embeddings?

Experimental Protocol

C = 2e20 FLOPs: $P_{tot} \approx 5.7$B, $P_{act} = 568$M

C = 6e20 FLOPs: $P_{tot} \approx 9.9$B, $P_{act} = 993$M

Key Findings

1. U-Shaped Relationship

Pure MoE (ρ=100%) is **not optimal**. Both extremes (ρ→0% and ρ→100%) underperform.

2. Optimal Allocation

ρ ≈ 75-80% to MoE, 20-25% to Engram

3. Stable Across Scales

Optimum location consistent across compute regimes

Quantitative Impact

10B Regime (C = 6e20)

Pure MoE: **1.7248**

Hybrid: 1.7109 (Δ = 0.0139)

Engram Performance

Achieves comparable performance at **ρ ≈ 40%**

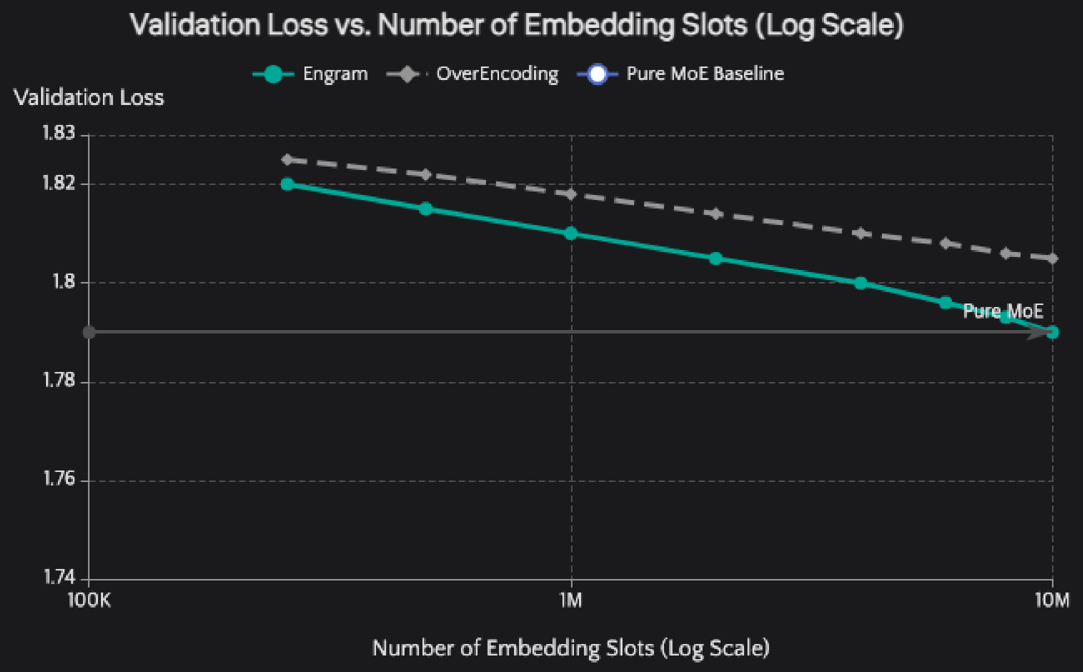
Structural Complementarity

MoE-dominated: Lacks memory for static patterns

Engram-dominated: Loses conditional computation

Infinite Memory Regime

Scaling Beyond Fixed Parameter Budgets



Experimental Protocol

Fixed MoE backbone ($P_{tot} \approx$ 3B, P_{act} = 568M). Sweep Engram slots M from **2.58×105** to **1.0×107** (adding up to ≈ 13B parameters).

Baseline Comparison

vs. OverEncoding (averaging integration). **Engram** unlockes much lar ger scaling potential from the same memory budget.

Power-Law Scaling

Linear in Log-Space

Strict power-law relationship: larger memory continues to pay off without requiring a dditional computation

Predictable Scaling Knob

Engram provides a **reliable mechanism** for capacity expansion

Key Insights

Range Explored

M: 2.58×10^5 → 1.0×10^7 slots

≈ 13B additional parameters

Observation

No sign of saturation within explored range

Implication

Engram scales effectively to massive memory capacities

Validation

Together with the allocation law, these results validate that **conditional memory serves as a distinct, scalable axis** of sparse capacity that complements the conditional computation of MoE.

Large Scale Pre-training

Comprehensive Benchmark Evaluation

Experimental Setup

Four models trained on **262B tokens** with identical data curriculum

Dense-4B: 4.1B total, 3.8B active

MoE-27B: 26.7B total, 3.8B active, 72 routed experts

Engram-27B: 26.7B total, 3.8B active, 55 experts + 5.7B Engram

Engram-40B: 39.5B total, 3.8B active, 55 experts + 18.5B Engram

Key Findings

Surprising Pattern

Gains are **even larger in general reasoning** than knowledge-intensive tasks. Engram relieves early layers from static reconstruction, effectively deepening the network.

Scaling Trend

Engram-40B continues improving, likely under-trained. Loss gap widens toward end of training.

Language Modeling
Pile
loss ↓

Knowledge
MMLU
+3.0

Reasoning
BBH
+5.0

Performance Gains: Engram-27B vs MoE-27B

Knowledge & Reasoning

- MMLU: +3.0
- MMLU-Pro: +1.8
- CMMLU: +4.0
- BBH: +5.0
- ARC-Challenge: +3.7
- DROP: +3.3

Code & Math

- HumanEval: +3.0
- MBPP: +1.6
- GSM8K: +2.2
- MATH: +2.4

Comprehensive Results Table (5-Shot)

Benchmark	MoE-27B	Engram-27B
Pile (loss)	1.950	**1.942**
MMLU	60.4	**63.4**
CMMLU	57.9	**61.9**
BBH	48.2	**53.2**
HumanEval	48.2	**51.2**
GSM8K	58.4	**60.6**
MATH	28.3	**30.7**

Selected benchmarks. All models trained for 262B tokens.

Long Context Performance

Preserving Attention Capacity for Global Context

Core Advantage

By **offloading local dependency modeling** to static lookups, Engram preserves valuable attention capacity for managing global context.

This structural advantage translates into **exceptional performance** in long-range retrieval and reasoning tasks.

Experimental Setup

Context Extension

YaRN on 32k context for 5k steps (30B tokens)

Controlled Comparison

Engram-27B (46k) vs MoE-27B (50k): **Iso-Loss**
Engram-27B (50k) vs MoE-27B (50k): **Iso-FLOPs**

Multi-Query NIAH
97.0
vs 84.2 MoE

Variable Tracking
89.0
vs 77.0 MoE

LongPPL & RULER Results

Iso-Loss Setting (46k vs 50k)

LongPPL	RULER
Book: 4.19 vs 4.38	NIAH: 99.3 vs 84.2
Paper: 2.84 vs 2.91	VT: 88.3 vs 77.0

Iso-FLOPs Setting (50k vs 50k)

LongPPL	RULER
Book: 4.14 vs 4.38	MQ: 97.0 vs 84.2
Code: 2.44 vs 2.49	QA: 40.5 vs 34.5

≈82% Compute (41k vs 50k)

Engram-27B (41k) **matches** MoE-27B (50k) on LongPPL and **surpasses** on RULER

Key Insight

Long-context performance is intrinsically coupled with general modeling ability. Engram's architecture superiority becomes evident when controlling for base capability.

Mechanistic Analysis

Is Engram Functionally Equivalent to Increasing Depth?

Hypothesis

By equipping the model with **explicit knowledge lookup**, Engram effectively mimics an increase in model depth by relieving it of early-stage feature composition.

This allows the network to **reach high-confidence predictions earlier** in the hierarchy.

Methodology

1. LogitLens Analysis

Project intermediate hidden states with final LM Head. Compute **KL divergence** between intermediate and final output distributions.

2. CKA Similarity

Centered Kernel Alignment compares representational structures. Compute pairwise similarity matrix $S \in [0,1]^{L \times L}$.

LogitLens Finding

↓ KL

early layers

CKA Finding

$a_j > j$

off-diagonal

Results: Accelerated Convergence

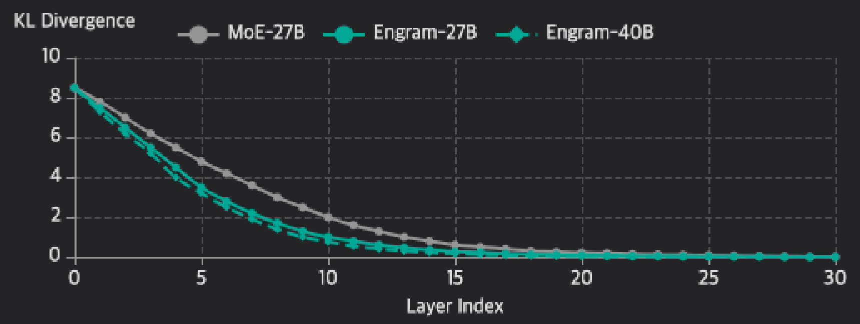


Figure: Layer-wise KL divergence (lower = closer to final prediction)

Results: Representational Alignment

Soft Alignment Index

a_j = weighted centroid of top-k most similar MoE layers

Engram-27B Layer 5 ≈ MoE Layer 12

Consistent off-diagonal shift validates depth equivalence

Validation

Both analyses confirm the central hypothesis: **by bypassing early-stage feature composition via explicit lookups, Engram is functionally equivalent to increasing the model's effective depth**, enabling deeper representations at earlier layers.

Ablation Studies

Component Importance and Sensitivity Analysis

Layer Sensitivity Sweep

Single Engram module (1.6B params) inserted at different layers

Baseline (MoE-3B): **1.808**
Layer 2: **1.770** (best single-layer)
Layer 6: 1.775 | Layer 12: 1.783

Efficacy degrades with deeper insertion

Placement Trade-Off

Early Injection

✓ Offloads patterns before backbone waste
✗ Weaker contextual queries

Late Injection

✓ Rich contextual gating
✗ Backbone already wasted depth

Optimal: Layer 2 (after one attention round)

Layered Design

Two modules (Layers 2 & 6) outperform single injection:

Single at Layer 2: **1.770**
Dual at Layers 2+6: **1.768** (reference)

Reconciles trade-off + enables memory hierarchy

Component Ablation

Removing sub-modules from reference (Val Loss = 1.768)

w/o Multi-Branch	1.775 ↑
w/o Gating	1.780 ↑
w/o Token Compress	1.783 ↑
+ 4-gram	1.778 ↑
w/o Short Conv	1.770 ↑

Critical Components

1. Branch-Specific Fusion
Enables distinct gating behaviors across mHC branches

2. Context-Aware Gating
Dynamic modulation suppresses noise, enables selectivity

3. Tokenizer Compression
Maximizes semantic density (23% vocab reduction)

CONCLUSION

A New Modeling Primitive

Conditional Memory as an Indispensable Component

Complementary Sparsity

Introduces **conditional memory** as a new axis of sparsity, complementing MoE's conditional computation to resolve inefficiency of simulating retrieval

U-Shaped Scaling Law

Uncovers optimal **75-80% MoE + 20-25% Engram** allocation. Hybrid strictly outperforms pure MoE across scales

Infrastructure-Aware Efficiency

Deterministic addressing enables **host memory offloading** with <3% overhead, decoupling storage from compute

Key Achievements

27B	BBH +5.0	97.0	<3%
Parameter Scale	Reasoning Gain	Multi-Query NIAH	Inference Overhead