deepseek

Conditional Memory via Scalable Lookup: A New Axis of Sparsity for Large Language Models

Xin Cheng[1,2], Wangding Zeng[2], Damai Dai[2], Qinyu Chen[2], Bingxuan Wang[2], Zhenda Xie[2], Kezhao Huang[2], Xingkai Yu[2], Zhewen Hao[2], Yukun Li[2], Han Zhang[2], Huishuai Zhang[1], Dongyan Zhao[1], Wenfeng Liang[2]

[1]**Peking University** [2]**DeepSeek-AI**

{zhanghuishuai, zhaody}@pku.edu.cn

{chengxin, zengwangding, damai.dai}@deepseek.com

Abstract

While Mixture-of-Experts (MoE) scales capacity via conditional computation, Transformers lack a native primitive for knowledge lookup, forcing them to inefficiently simulate retrieval through computation. To address this, we introduce conditional memory as a complementary sparsity axis, instantiated via **Engram**, a module that modernizes classic N-gram embedding for $O(1)$ lookup. By formulating the *Sparsity Allocation* problem, we uncover a U-shaped scaling law that optimizes the trade-off between neural computation (MoE) and static memory (Engram). Guided by this law, we scale Engram to 27B parameters, achieving superior performance over a strictly iso-parameter and iso-FLOPs MoE baseline. Most notably, while the memory module is expected to aid knowledge retrieval (e.g., MMLU +3.4; CMMLU +4.0), we observe even larger gains in general reasoning (e.g., BBH +5.0; ARC-Challenge +3.7) and code/math domains (HumanEval +3.0; MATH +2.4). Mechanistic analyses reveal that Engram relieves the backbone's early layers from static reconstruction, effectively deepening the network for complex reasoning. Furthermore, by delegating local dependencies to lookups, it frees up attention capacity for global context, substantially boosting long-context retrieval (e.g., Multi-Query NIAH: 84.2 → 97.0). Finally, Engram establishes infrastructure-aware efficiency: its deterministic addressing enables runtime prefetching from host memory, incurring negligible overhead. We envision conditional memory as an indispensable modeling primitive for next-generation sparse models. Code available at: https://github.com/deepseek-ai/Engram

1. Introduction

Sparsity is a recurring design principle for intelligent systems, spanning from biological neural circuits (Lennie, 2003; Olshausen and Field, 1997) to modern Large Language Models (LLMs). Currently, this principle is primarily realized through Mixture-of-Experts (MoE) (Dai et al., 2024; Shazeer et al., 2017), which scales capacity via conditional computation. Owing to its ability to drastically increase model size without proportional increases in compute, MoE has become the de facto standard for frontier models (Comanici et al., 2025; Guo et al., 2025; Team et al., 2025).

Despite the success of this conditional computation paradigm, the intrinsic heterogeneity of linguistic signals suggests significant room for *structural optimization*. Specifically, language modeling entails two qualitatively different sub-tasks: compositional reasoning and knowl-

edge retrieval. While the former demands deep, dynamic computation, a substantial portion of text—such as named entities and formulaic patterns—is local, static, and highly stereotyped (Constant et al., 2017; Erman, 2000). The effectiveness of classical N-gram models (Brants et al., 2007; Liu et al., 2024b; Nguyen, 2024) in capturing such local dependencies implies that these regularities are naturally represented as computationally inexpensive lookups. Since standard Transformers (Vaswani et al., 2017) lack a native knowledge lookup primitive, current LLMs are forced to **simulate retrieval through computation**. For instance, resolving a common multi-token entity requires consuming multiple early layers of attention and feed-forward networks (Ghandeharioun et al., 2024; Jin et al., 2025) (see Table 3). This process essentially amounts to an expensive runtime reconstruction of a static lookup table, wasting valuable sequential depth on trivial operations that could otherwise be allocated to higher-level reasoning.

To align model architecture with this linguistic duality, we advocate for a complementary axis of sparsity: conditional memory. Whereas conditional computation sparsely activates parameters to process dynamic logic (Bengio et al., 2013; Shazeer et al., 2017), conditional memory relies on sparse lookup operations to retrieve static embeddings for fixed knowledge. As a preliminary exploration of this paradigm, we revisit N-gram embeddings (Bojanowski et al., 2017) as a canonical instantiation: local context serves as a key to index a massive embedding table via constant-time $O(1)$ lookups (Huang et al., 2025a; Pagnoni et al., 2025; Tito Svenstrup et al., 2017; Yu et al., 2025). Our investigation reveals that, perhaps surprisingly, this static retrieval mechanism can serve as an ideal complement to modern MoE architecture—but only if it is properly designed. In this paper, we propose **Engram**, a conditional memory module grounded in the classic N-gram structure but equipped with modern adaptations such as tokenizer compression, multi-head hashing, contextualized gating, and multi-branch integration (detailed in Section 2).

To quantify the synergy between these two primitives, we formulate the *Sparsity Allocation* problem: given a fixed total parameter budget, how should capacity be distributed between MoE experts and Engram memory? Our experiments uncover a distinct U-shaped scaling law, revealing that even simple lookup mechanisms, when treated as a first-class modeling primitive, act as essential complements to neural computation. Guided by this allocation law, we scale Engram to a 27B-parameter model. Compared to a strictly iso-parameter and iso-FLOPs MoE baseline, Engram-27B achieves superior efficiency across diverse domains. Crucially, the gains are not limited to knowledge-intensive tasks (e.g., MMLU: +3.4; CMMLU: +4.0; MMLU-Pro: +1.8), where memory capacity is intuitively beneficial; we observe even more significant improvements in general reasoning (e.g., BBH: +5.0; ARC-Challenge: +3.7; DROP: +3.3) and code/math domains (e.g., HumanEval: +3.0; MATH: +2.4; GSM8K: +2.2).

Mechanistic analysis via LogitLens (nostalgebraist, 2020) and CKA (Hendrycks et al., 2021a) reveals the source of these gains: Engram relieves the backbone from reconstructing static knowledge in early layers, thereby increasing effective depth available for complex reasoning. Furthermore, by delegating local dependencies to lookups, Engram frees up attention capacity to focus on global context, enabling exceptional performance in long-context scenarios—substantially outperforming baselines on LongPPL (Fang et al.) and RULER (Hsieh et al.) (e.g., Multi-Query NIAH: 97.0 vs. 84.2; Variable Tracking: 89.0 vs. 77.0).

Finally, we establish infrastructure-aware efficiency as a first-class principle. Unlike MoE's dynamic routing, Engram employs deterministic IDs to enable runtime prefetching, overlapping communication with computation. Empirical results show that offloading a 100B-parameter table to host memory incurs negligible overhead ($< 3\%$). This demonstrates that Engram effectively bypasses GPU memory constraints, facilitating aggressive parameter expansion.

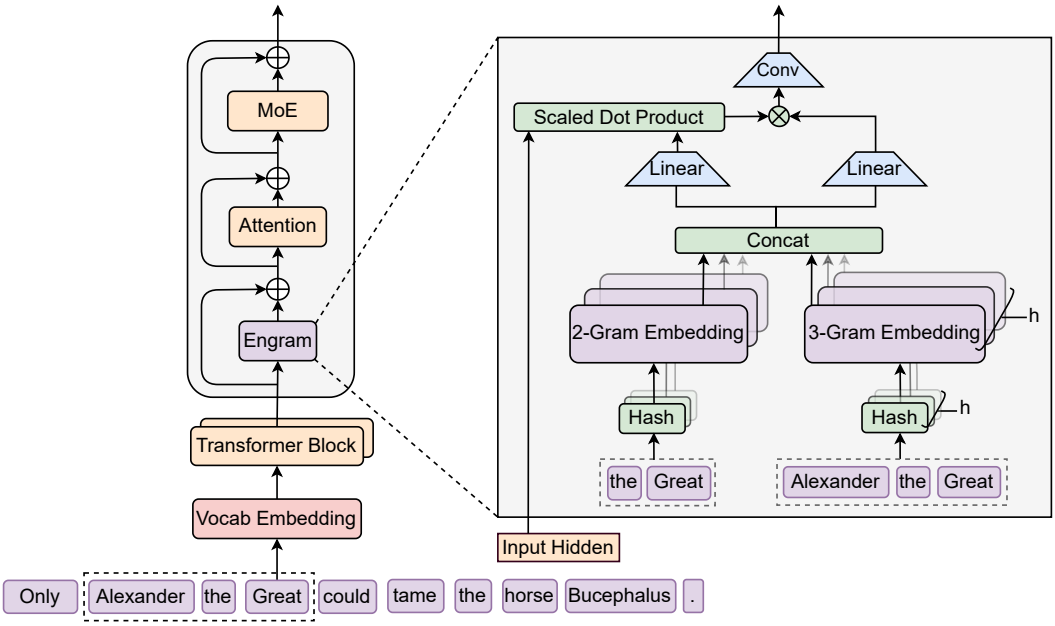

Figure 1 | **The Engram Architecture.** The module augments the backbone by retrieving static N-gram memory and fusing it with dynamic hidden states via context-aware gating. This module is applied only to specific layers to decouple memory from compute, leaving the standard input embedding and un-embedding module intact.

2. Architecture

2.1. Overview

As shown in Figure 1, Engram is a conditional memory module designed to augment the Transformer backbone by structurally separating static pattern storage from dynamic computation. Formally, given an input sequence $X = (x_1, \ldots, x_T)$ and hidden states $\mathbf{H}^{(\ell)} \in \mathbb{R}^{T \times d}$ at layer ℓ, the module processes each position t in two functional phases: **retrieval and fusion**. First, as detailed in Section 2.2, we extract and compress suffix N-grams to deterministically retrieve static embedding vectors via hashing. Subsequently, in Section 2.3, these retrieved embeddings are dynamically modulated by the current hidden state and refined via a lightweight convolution. Finally, we discuss the integration with multi-branch architectures in Section 2.4 and the system-level design in Section 2.5.

2.2. Sparse Retrieval via Hashed N-grams

The first phase maps local contexts to static memory entries, involving tokenizer compression and retrieving embeddings via deterministic hashing.

Tokenizer Compression While N-gram models typically operate directly on tokenizer outputs, standard subword tokenizers prioritize *lossless reconstruction*, often assigning disjoint IDs to semantically equivalent terms (e.g., `Apple` vs. `␣apple`) (Kudo and Richardson, 2018; Li et al., 2023b). To maximize semantic density, we implement a vocabulary projection layer. Specifically, we pre-compute a surjective function $\mathcal{P} : V \to V'$ that collapses raw token IDs into canonical

identifiers based on normalized textual equivalence (using NFKC (Whistler, 2025), lowercasing, etc.). In practice, this process achieves a 23% reduction in the effective vocabulary size for a 128k tokenizer (see Appendix C). Formally, for a token at position t, we map its raw ID x_t to a canonical ID $x'_t = \mathcal{P}(x_t)$ to form the suffix N-gram $g_{t,n} = (x'_{t-n+1}, \dots, x'_t)$.

Multi-Head Hashing. Directly parameterizing the combinatorial space of all possible N-grams is intractable. Following Tito Svenstrup et al. (2017), we adopt a hashing-based approach. To mitigate collisions, we employ K distinct hash heads for each N-gram order n. Each head k maps the compressed context to an index within an embedding table $\mathbf{E}_{n,k}$ (of prime size $M_{n,k}$) via a deterministic function $\varphi_{n,k}$:

$$z_{t,n,k} \triangleq \varphi_{n,k}(g_{t,n}), \quad \mathbf{e}_{t,n,k} = \mathbf{E}_{n,k}[z_{t,n,k}]. \tag{1}$$

In practice, $\varphi_{n,k}$ is implemented as a lightweight multiplicative-XOR hash. We construct the final memory vector $\mathbf{e}_t \in \mathbb{R}^{d_{\text{mem}}}$ by concatenating all retrieved embeddings:

$$\mathbf{e}_t \triangleq \Big\Vert_{n=2}^{N} \Big\Vert_{k=1}^{K} \mathbf{e}_{t,n,k}. \tag{2}$$

2.3. Context-aware Gating

The retrieved embeddings $\mathbf{e}_t$ serve as context-independent priors. Being static, however, they inherently lack contextual adaptability and may suffer from noise due to hash collisions or polysemy (Haber and Poesio, 2024). To enhance expressivity and resolve this ambiguity, we employ a context-aware gating mechanism inspired by Attention (Bahdanau et al., 2015; Vaswani et al., 2017). Specifically, we utilize the current hidden state $\mathbf{h}_t$—which has aggregated global context via preceding attention layers—as a dynamic Query, while the retrieved memory $\mathbf{e}_t$ serves as the source for both Key and Value projections:

$$\mathbf{k}_t = \mathbf{W}_K \mathbf{e}_t, \quad \mathbf{v}_t = \mathbf{W}_V \mathbf{e}_t \tag{3}$$

where $\mathbf{W}_K, \mathbf{W}_V$ are learnable projection matrices. To ensure gradient stability (Dehghani et al., 2023), we apply RMSNorm (Zhang and Sennrich, 2019) to the Query and Key before computing the scalar gate $\alpha_t \in (0, 1)$:

$$\alpha_t = \sigma\left(\frac{\text{RMSNorm}(\mathbf{h}_t)^\top \text{RMSNorm}(\mathbf{k}_t)}{\sqrt{d}}\right). \tag{4}$$

The gated output is defined as $\tilde{\mathbf{v}}_t = \alpha_t \cdot \mathbf{v}_t$. This design enforces semantic alignment: if the retrieved memory $\mathbf{e}_t$ contradicts the current context $\mathbf{h}_t$, the gate α_t tends toward zero, effectively suppressing the noise.

Finally, to expand the receptive field and enhance the model's non-linearity, we introduce a short, depthwise causal convolution (Gu et al., 2022; Peng et al., 2023). Let $\tilde{\mathbf{V}} \in \mathbb{R}^{T \times d}$ denote the sequence of gated values. Using a kernel size w (set to 4), dilation δ (set to the max N-gram order) and SiLU activation (Elfwing et al., 2018), the final output $\mathbf{Y}$ is computed as:

$$\mathbf{Y} = \text{SiLU}\left(\text{Conv1D}(\text{RMSNorm}(\tilde{\mathbf{V}}))\right) + \tilde{\mathbf{V}}, \tag{5}$$

The Engram module is integrated into the backbone via a residual connection: $\mathbf{H}^{(\ell)} \leftarrow \mathbf{H}^{(\ell)} + \mathbf{Y}$, followed by the standard Attention and MoE. Crucially, Engram is not applied to every layer; its specific placement is governed by the system-level latency constraints detailed in Section 2.5.

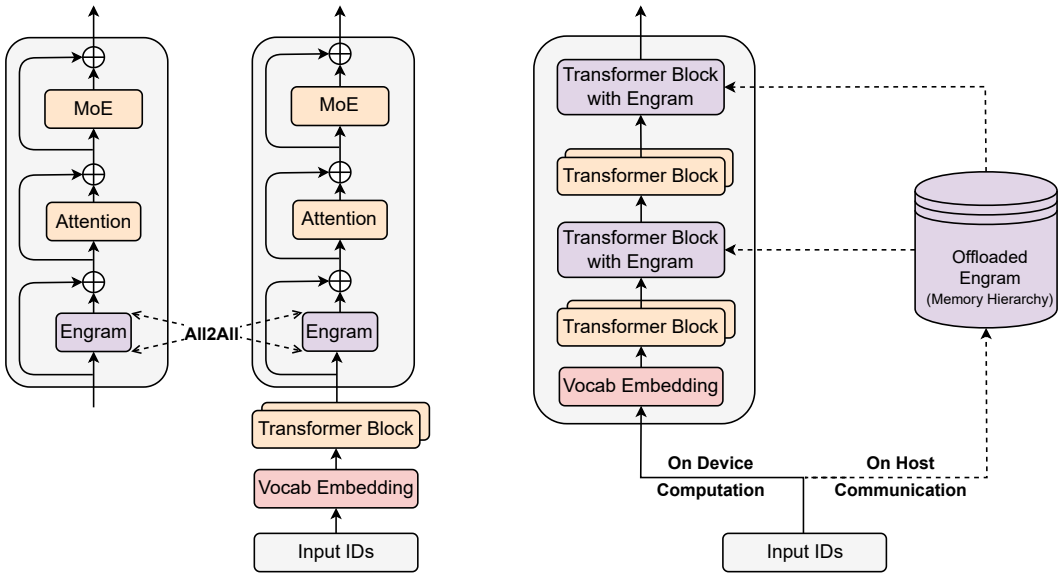

Figure 2 | **System implementation of Engram.** (a) Training Phase: The massive embedding tables are sharded across available GPUs. An All-to-All communication primitive is employed to retrieve active embedding rows across devices. (b) Inference Phase: Engram tables are offloaded to host memory. By exploiting the deterministic retrieval logic, the host asynchronously prefetches and transfers embeddings, overlapping communication with the on-device computation of preceding Transformer blocks.

2.4. Integration with Multi-branch Architecture

In this work, rather than standard single-stream connections (He et al., 2016), we adopt the advanced multi-branch architecture as our default backbone, chosen for its superior modeling capabilities (Larsson et al., 2017; Szegedy et al., 2015; Xie et al., 2025; Zhu et al., 2025). A defining characteristic of this architecture is the expansion of the residual stream into M parallel branches, where information flow is modulated by learnable connection weights.

Although the Engram module is inherently topology-agnostic, adapting it to this multi-branch framework necessitates structural optimization to balance efficiency and expressivity. Specifically, we implement a parameter-sharing strategy: a single sparse embedding table and a Value projection matrix $\mathbf{W}_V$ are shared across all M branches, whereas M distinct Key projection matrices $\{\mathbf{W}_K^{(m)}\}_{m=1}^{M}$ are employed to enable branch-specific gating behaviors. For the m-th branch with hidden state $\mathbf{h}_t^{(m)}$, the branch-specific gating signal is computed as:

$$\alpha_t^{(m)} = \sigma\left(\frac{\mathrm{RMSNorm}(\mathbf{h}_t^{(m)})^\top \mathrm{RMSNorm}(\mathbf{W}_K^{(m)}\mathbf{e}_t)}{\sqrt{d}}\right). \tag{6}$$

The retrieved memory is then modulated by these independent gates applied to the shared value vector: $\mathbf{u}_t^{(m)} = \alpha_t^{(m)} \cdot (\mathbf{W}_V \mathbf{e}_t)$. This design allows the linear projections (one $\mathbf{W}_V$ and M distinct $\mathbf{W}_K^{(m)}$) to be fused into a single dense FP8 matrix multiplication, maximizing the compute utilization of modern GPUs. Unless otherwise stated, all experiments utilize this integration with Manifold-Constrained Hyper-Connections ($M = 4$) (Xie et al., 2025).

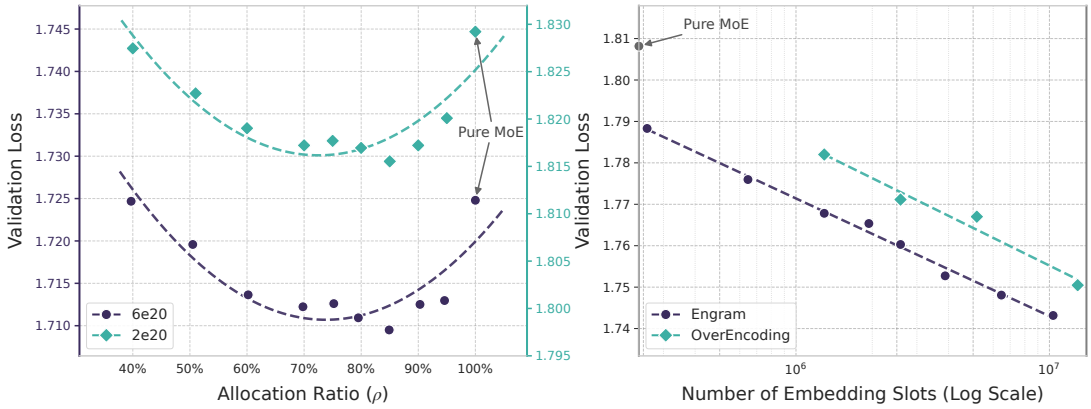

Figure 3 | **Sparsity allocation and Engram scaling.** Left: Validation loss across allocation ratios ρ. Two compute budgets are shown (2e20 and 6e20 FLOPs). Both regimes exhibit a U-shape, with hybrid allocation surpassing Pure MoE. Right: Scaling behavior in the infinite-memory regime. Validation loss exhibits a log-linear trend with respect to the number of embeddings.

2.5. System Efficiency: Decoupling Compute and Memory

Scaling memory-augmented models is often constrained by the limited capacity of GPU high-bandwidth memory (HBM). However, the deterministic retrieval mechanism of Engram naturally supports the decoupling of parameter storage from computational resources. Unlike MoE, which relies on runtime hidden states for dynamic routing, Engram's retrieval indices depend solely on the input token sequence. This predictability facilitates specialized optimization strategies for both training and inference, as illustrated in Figure 2.

During training, to accommodate large-scale embedding tables, we employ standard model parallelism by sharding the tables across available GPUs. An All-to-All communication primitive is used to gather active rows in the forward pass and dispatch gradients in the backward pass, enabling the total memory capacity to scale linearly with the number of accelerators.

During inference, this deterministic nature enables a prefetch-and-overlap strategy. Since memory indices are known prior to the forward pass, the system can asynchronously retrieve embeddings from abundant host memory via PCIe. To effectively mask communication latency, the Engram module is placed at specific layers within the backbone, leveraging the computation of preceding layers as a buffer to prevent GPU stalls. This necessitates a hardware-algorithm co-design strategy: while placing Engram deeper extends the compute window available for hiding latency, our ablation in Section 6.2 shows that modeling performance favors early intervention to offload local pattern reconstruction. Therefore, the optimal placement must simultaneously satisfy both modeling and system latency constraints.

Furthermore, natural language N-grams inherently follow a Zipfian distribution (Chao and Zipf, 1950; Piantadosi, 2014), where a small fraction of patterns accounts for the vast majority of memory accesses. This statistical property motivates a Multi-Level Cache Hierarchy: frequently accessed embeddings can be cached in faster storage tiers (e.g., GPU HBM or Host DRAM), while the long tail of rare patterns resides in slower, high-capacity media (e.g., NVMe SSD). This stratification allows Engram to scale to massive memory capacities with minimal impact on effective latency.

3. Scaling Laws and Sparsity Allocation

Engram, as an instantiation of conditional memory, is structurally complementary to the conditional computation provided by MoE experts. This section investigates the scaling properties of this duality and how to optimally allocate sparse capacity. Specifically, two key questions drive our research:

1. **Allocation under Finite Constraints.** When total parameters and training compute are fixed (Iso-parameter and Iso-FLOPs), how should we split the sparse capacity between MoE experts and Engram embeddings?
2. **Infinite Memory Regime.** Considering the non-scaling $O(1)$ overhead of Engram, if the memory budget is relaxed or scaled aggressively, what scaling behavior does Engram exhibit by itself?

3.1. Optimal Allocation Ratio Between MoE and Engram

Compute-matched formulation. We analyze the trade-off using three parameter metrics:

- P_{tot}: total trainable parameters, excluding vocabulary embedding and LM head.
- P_{act}: activated parameters per token. This quantity determines the training cost (FLOPs).
- $P_{\text{sparse}} \triangleq P_{\text{tot}} - P_{\text{act}}$: the *inactive* parameters, which represents the "free" parameter budget available for scaling model size without incurring computational cost (e.g., unselected experts or unretrieved embeddings).

We keep P_{tot} and P_{act} fixed within each FLOPs budget, so that models have the same number of parameters and the same per-token FLOPs. For MoE, P_{act} is determined by the top-k selected experts, while the parameters of non-selected experts contribute to P_{sparse}. For Engram, only a constant number of slots are retrieved per token, so scaling the number of embedding slots increases P_{tot} without increasing per-token FLOPs.

Allocation ratio. We define the allocation ratio $\rho \in [0, 1]$ as the fraction of the inactive-parameter budget assigned to MoE expert capacity:

$$P_{\text{MoE}}^{(\text{sparse})} = \rho\, P_{\text{sparse}}, \qquad P_{\text{Engram}} = (1 - \rho)\, P_{\text{sparse}}. \tag{7}$$

Intuitively:

- $\rho = 1$ corresponds to a pure MoE model (all inactive parameters are routed experts).
- $\rho < 1$ reduces the number of routed experts and reallocates the freed parameters to Engram embedding slots.

Experimental protocol. We evaluate this trade-off at two compute regimes and maintain a constant sparsity ratio $P_{\text{tot}}/P_{\text{act}} \approx 10$ across both settings:

- $C = 2 \times 10^{20}$ FLOPs: $P_{\text{tot}} \approx 5.7$B and $P_{\text{act}} = 568$M. The baseline ($\rho = 1$) has a total of 106 experts.
- $C = 6 \times 10^{20}$ FLOPs: $P_{\text{tot}} \approx 9.9$B and $P_{\text{act}} = 993$M. The baseline ($\rho = 1$) has a total of 99 experts.

For different ρ, we construct the corresponding model only by adjusting the number of routed experts and the number of Engram embedding slots. All runs use the identical training pipeline and optimization hyperparameters.

Results and Analysis. Figure 3 (left) reveals a consistent U-shaped relationship between validation loss and the allocation ratio ρ. Remarkably, the Engram model achieves comparable performance to the pure MoE baseline ($\rho = 100\%$) even when the MoE allocation is reduced to just $\rho \approx 40\%$ (i.e., a total of 46 experts for the 5.7B model and 43 experts for the 9.9B model). Furthermore, the pure MoE baseline proves suboptimal: reallocating roughly 20%–25% of the sparse parameter budget to Engram yields the best performance. Quantitatively, in the 10B regime ($C = 6 \times 10^{20}$), validation loss improves from 1.7248 (at $\rho = 100\%$) to 1.7109 near the optimum of $\rho \approx 80\%$ ($\Delta = 0.0139$). Crucially, the location of this optimum is stable across regimes ($\rho \approx 75\%$–80%), suggesting a robust allocation preference across the examined scales (under fixed sparsity). This observed U-shape confirms the structural complementarity between the two modules:

- **MoE-dominated ($\rho \to 100\%$):** The model lacks dedicated memory for static patterns, forcing it to inefficiently reconstruct them through depth and computation.
- **Engram-dominated ($\rho \to 0\%$):** The model loses conditional computation capacity, hurting tasks that require dynamic, context-dependent reasoning; memory cannot replace computation in this regime.

3.2. Engram under Infinite Memory Regime

In Section 3.1, we optimized the allocation under a fixed parameter budget. We now explore the complementary setting: aggressive memory scaling. This investigation is motivated by Engram's unique ability to decouple storage from compute detailed in Section 2.5.

Experimental protocol. We utilize a fixed MoE backbone with $P_{\text{tot}} \approx 3\text{B}$ and $P_{\text{act}} = 568\text{M}$, trained for 100B tokens to ensure convergence. On top of this backbone, we attach an Engram table and sweep the number of slots M from 2.58×10^5 to 1.0×10^7 (adding up to ≈ 13 billion parameters). For baselines, we compare against OverEncoding (Huang et al., 2025a), which integrates N-gram embeddings via averaging with the vocabulary embedding. We note that while other work such as SCONE (Yu et al., 2025) also investigates large-scale embeddings, it is primarily inference-focused and includes extra module (*f-gram model*) and additional training FLOPs, rendering it incompatible with the strict iso-compute constraints of this study.

Results. Figure 3 (right) demonstrates that scaling the number of memory slots yields a clear and consistent improvement in validation loss. Across the explored range, the curve follows a strict power law (linear in log-space), indicating that Engram provides a predictable scaling knob: larger memory continues to pay off without requiring additional computation. Crucially, regarding scaling efficiency: while the direct averaging approach of OverEncoding benefits from larger memory tables, Engram unlocks much larger scaling potential from the same memory budget. Together with the allocation law in Section 3.1, these results validate that conditional memory serves as a distinct, scalable axis of sparse capacity that complements the conditional computation of MoE.

4. Large Scale Pre-training

With the proposed Engram architecture and the empirically derived allocation law, we scale Engram to the multi-billion parameter to validate its efficacy in real-world language model pre-training. Specifically, we train four models: (1) **Dense-4B** (4.1B total parameters), (2) **MoE-27B**

Table 1 | **Pre-training performance comparison between dense, MoE, and Engram models**. All models are trained for 262B tokens and are matched in activated parameters (3.8B). Engram-27B is iso-parameters with MoE-27B by reallocating parameters from routed experts (72 → 55) to a 5.7B-parameter Engram memory. Engram-40B further increases Engram memory (18.5B parameters) while keeping the activated-parameter budget fixed. Full training-time benchmark trajectories are reported in Appendix B.

	Benchmark (Metric)	**# Shots**	**Dense-4B**	**MoE-27B**	**Engram-27B**	**Engram-40B**
	# Total Params		4.1B	26.7B	26.7B	39.5B
	# Activated (w/o token embed)		3.8B	3.8B	3.8B	3.8B
	# Trained Tokens		262B	262B	262B	262B
	# Experts (shared + routed, top-k)		-	2 + 72 (top-6)	2 + 55 (top-6)	2 + 55 (top-6)
	# Engram Params		-	-	5.7B	18.5B
Language Modeling	Pile (loss)	-	2.091	1.960	**1.950**	1.942
	Validation Set (loss)	-	1.768	1.634	**1.622**	1.610
Knowledge & Reasoning	MMLU (Acc.)	5-shot	48.6	57.4	**60.4**	60.6
	MMLU-Redux (Acc.)	5-shot	50.7	60.6	**64.0**	64.5
	MMLU-Pro (Acc.)	5-shot	21.1	28.3	**30.1**	31.3
	CMMLU (Acc.)	5-shot	47.9	57.9	**61.9**	63.4
	C-Eval (Acc.)	5-shot	46.9	58.0	**62.7**	63.3
	AGIEval (Acc.)	0-shot	29.1	38.6	**41.8**	45.9
	ARC-Easy (Acc.)	25-shot	76.8	86.5	**89.0**	90.1
	ARC-Challenge (Acc.)	25-shot	59.3	70.1	**73.8**	76.4
	TriviaQA (EM)	5-shot	33.0	48.8	**50.7**	51.8
	TriviaQA-ZH (EM)	5-shot	62.8	74.8	**76.3**	77.9
	PopQA (EM)	15-shot	15.1	19.2	**19.4**	21.2
	CCPM (Acc.)	0-shot	72.2	79.6	**87.1**	87.7
	BBH (EM)	3-shot	42.8	50.9	**55.9**	57.5
	HellaSwag (Acc.)	0-shot	64.3	71.8	**72.7**	73.1
	PIQA (Acc.)	0-shot	63.8	71.9	**73.5**	76.5
	WinoGrande (Acc.)	5-shot	64.0	67.6	**67.8**	68.1
Reading Comprehension	DROP (F1)	1-shot	41.6	55.7	**59.0**	60.7
	RACE-Middle (Acc.)	5-shot	72.4	80.9	**82.8**	83.3
	RACE-High (Acc.)	5-shot	66.0	75.4	**78.2**	79.2
	C3 (Acc.)	0-shot	57.7	60.1	**63.6**	61.8
Code & Math	HumanEval (Pass@1)	0-shot	26.8	37.8	**40.8**	38.4
	MBPP (Pass@1)	3-shot	35.4	46.6	**48.2**	46.2
	CruxEval-i (EM)	0-shot	27.6	30.7	**32.2**	36.2
	CruxEval-o (EM)	0-shot	28.7	34.1	**35.0**	35.3
	GSM8K (EM)	8-shot	35.5	58.4	**60.6**	62.6
	MGSM (EM)	8-shot	27.0	46.8	**49.4**	52.4
	MATH (EM)	4-shot	15.2	28.3	**30.7**	30.6

(26.7B total parameters), (3) **Engram-27B** (26.7B total parameters), and (4) **Engram-40B** (39.5B total parameters). All models are trained using an identical data curriculum (same token budget and order) and are strictly matched in the number of activated parameters.

4.1. Experimental Setup

Training Data and Model Configurations All models are pre-trained on a corpus of 262 billion tokens and we utilize the tokenizer from DeepSeek-v3 (Liu et al., 2024a) with a vocabulary size of 128k. For modeling, to ensure a controlled comparison, we adhere to a consistent default setting across all models unless explicitly stated otherwise. We utilize a 30-block Transformer with a

hidden size of 2560. Each block integrates a Multi-head Latent Attention (MLA) (DeepSeek-AI et al., 2024) with 32 heads, connected to FFNs via mHC (Xie et al., 2025) with an expansion rate of 4. All models are optimized using Muon (Jordan et al., 2024; Team et al., 2025); detailed hyperparameters are listed in the Appendix A. We instantiate four distinct models:

- **Dense-4B** serves as the baseline model. It utilizes the backbone architecture described above, incorporating a standard dense FFN into every block.
- **MoE-27B** replaces the standard dense FFN with a DeepSeekMoE module (Dai et al., 2024). Configured with 72 routed experts and 2 shared experts (activating the top-k = 6 routed experts per token), this model scales to 26.7B total parameters while maintaining the same activated parameters as Dense-4B.
- **Engram-27B** is strictly derived from the **MoE-27B** architecture to ensure fair comparison. We reduce the number of routed experts from 72 to 55 and reallocate the freed parameters to a 5.7B-parameter embedding module ($\rho = 74.3\%$), keeping the total model size constant at 26.7B. Regarding the Engram configuration, we instantiate the module at layers 2 and 15 and set the maximum N-gram size to 3, the number of heads to 8, and the dimension to 1280. For optimization, the embedding parameters are updated using Adam (Kingma, 2014) with a learning rate scaled by 5× and no weight decay, while the convolution parameters are initialized to zero to strictly preserve the identity mapping at the start of training.
- **Engram-40B** retains the same backbone and computation budget as Engram-27B but scales the sparse embedding module to 18.5B parameters (totaling 39.5B parameters). This model is designed to investigate the scaling properties of Engram.

Evaluation Protocol We evaluate models on a diverse suite of benchmarks spanning language modeling, knowledge, reasoning, reading comprehension, and code/math. For each benchmark, we follow standard prompting protocols and evaluation metrics.

- **Language Modeling:** We report loss on the test set of The Pile (Gao et al., 2020) and an validation set drawn from the same distribution as the training data.
- **Knowledge & Reasoning:** MMLU (Hendrycks et al., 2021a), MMLU-Redux (Gema et al., 2025), MMLU-Pro (Wang et al., 2024b), CMMLU (Li et al., 2024), C-Eval (Huang et al., 2023), AGIEval (Zhong et al., 2024), ARC-Easy/Challenge (Clark et al., 2018), TriviaQA (Joshi et al., 2017), TriviaQA-ZH (internal), PopQA (Mallen et al., 2023), CCPM (Li et al., 2021), BBH (Suzgun et al., 2023), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021).
- **Reading Comprehension:** DROP (Dua et al., 2019), RACE (Middle/High) (Lai et al., 2017), and C3 (Sun et al., 2020).
- **Code & Math:** HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), CruxEval (Gu et al., 2024), GSM8K (Cobbe et al., 2021), MGSM (Shi et al., 2023), and MATH (Hendrycks et al., 2021b).

4.2. Experimental Results

Table 1 summarizes the main results. First, consistent with prior literature (Borgeaud et al., 2022; He, 2024; Shazeer et al., 2017), sparse architectures demonstrate superior scaling laws compared to dense models. Under the same training compute budget, all three sparse variants (MoE-27B, Engram-27B/40B) significantly outperform the iso-FLOPs Dense-4B baseline across all benchmarks.

Table 2 | **Long-context performance comparison.** Parenthetical values (e.g. *(50k, 1.62)*) denote the pre-training steps and the corresponding loss prior to the long-context extension. Two key findings: (1) With only 82% of the pre-training FLOPs (41k vs. 50k), Engram-27B matches the baseline's LongPPL (Fang et al.) performance while achieving significantly higher accuracy on RULER (Hsieh et al.); (2) Under both iso-pretraining-loss (46k) and iso-pretraining-FLOPs (50k) settings, Engram-27B substantially outperforms the baseline across all metrics. **Bold** indicates the best and underline the second.

Model	LongPPL (32k)				RULER (32k)							
	Perplexity (↓)				NIAH Accuracy (↑)				Other Tasks (↑)			
	Book	Paper	Code	L-CoT	S	MK	MV	MQ	VT	CWE	FWE	QA
MoE-27B *(50k, 1.63)*	4.38	2.91	2.49	14.16	**100.0**	88.0	92.7	84.2	77.0	4.5	73.0	34.5
Engram-27B *(41k, 1.66)*	4.37	2.92	2.50	14.26	99.6	88.3	93.0	89.5	83.2	3.8	99.6	**44.0**
Engram-27B *(46k, 1.63)*	4.19	2.84	2.45	13.59	97.6	89.0	95.5	**97.0**	87.2	4.3	98.6	37.5
Engram-27B *(50k, 1.62)*	**4.14**	**2.82**	**2.44**	**13.41**	99.3	**89.3**	**96.5**	**97.0**	**89.0**	**5.9**	**99.3**	40.5

More importantly, Engram-27B consistently improves over the iso-parameter and iso-FLOPs MoE-27B baseline. Interestingly, these gains are not limited to knowledge-intensive tasks (e.g., MMLU: +3.0, MMLU-Pro: +1.8, CMMLU: +4.0), where memory capacity is intuitively beneficial. We observe even more significant improvements in general-reasoning domains (e.g., BBH: +5.0, ARC-Challenge: +3.7, DROP: +3.3), as well as code and mathematical reasoning (e.g., HumanEval: +3.0, MBPP: +1.6, GSM8K: +2.2, MATH: +2.4). To reduce the impact of benchmark noise and to visualize training dynamics, we provide full benchmark trajectories during pre-training in Appendix B. These results support our hypothesis that introducing a dedicated knowledge lookup primitive improves representation efficiency beyond what can be achieved by allocating the entire sparse budget to conditional computation.

Finally, scaling to Engram-40B further reduces pre-training loss and improves performance across most benchmarks. Although it does not yet strictly dominate Engram-27B on every task, this is likely an artifact of under-training. We observe that the training loss gap between Engram-40B and the baselines continues to widen towards the end of training, suggesting that the expanded memory capacity has not yet fully saturated within the current token budget.

5. Long Context Training

By offloading local dependency modeling to static lookups, the Engram architecture preserves valuable attention capacity for managing global context. In this section, we empirically verify this structural advantage by conducting long-context extension training (Gao et al., 2025; Peng et al., 2024). Through a rigorous evaluation protocol that isolates architectural contributions from base model capabilities, we demonstrate that Engram yields significant gains in long-range retrieval and reasoning tasks.

5.1. Experimental Setup

Training Details. To enable long-context capabilities, we adopt the context expansion strategy introduced in DeepSeek-V3 (Liu et al., 2024a). Following the pre-training stage, we apply YaRN (Peng et al., 2024) for context window extension in a 32768-token context training stage for 5,000 steps (30B tokens of high-quality, long-context data). The hyper-parameters are scale $s = 10, \alpha = 1, \beta = 32$ and the scaling factor $f = 0.707$.

Model Configurations. We compare context extensions across four distinct model configurations. We utilize the final pre-training checkpoints (50k steps) for both MoE-27B and Engram-27B. Additionally, to rigorously benchmark architectural efficiency, we select two intermediate checkpoints for Engram-27B at 41k and 46k steps. Despite differing initialization stages, all variants undergo *the exact same* context extension training protocol. Crucially, Engram-27B (46k) is selected because it exhibits the same pre-training loss as the fully trained MoE-27B (50k). This creates a controlled "Iso-Loss" setting, ensuring that any performance divergence during context extension is attributable to the architecture rather than the starting quality of the model.

Evaluation Benchmarks. We assess long-context performance using LongPPL (Fang et al.) and RULER (Hsieh et al.). For LongPPL, we construct evaluation sets spanning four categories: long books, research papers, code repositories, and long chain-of-thought (CoT) trajectories. For RULER, we evaluate on 14 subsets aggregated into 8 categories: Single (S), Multi-keys (MK), Multi-values (MV) and Multi-queries (MQ) Needle-in-a-Haystack; Multi-hop Variable Tracking (VT), Common Words Extraction (CWE), Frequent Words Extraction (FWE), and Question Answering (QA).

5.2. Experimental Results

The evaluation results are summarized in Table 2. To accurately assess the contribution of the Engram architecture, our analysis proceeds in two steps: first, decoupling the impact of base model capability from architectural design, and second, conducting a controlled analysis.

1. Long-Context Capability Beyond Attention Mechanics. While attention mechanisms and positional encoding provide the structural basis for context processing (Press et al., 2022; Su et al., 2024; Xiao et al., 2024; Yang et al., 2025), our results indicate that long-context performance is not solely determined by architectural priors. Observing the trajectory of Engram (41k $\rightarrow$ 50k), we find that long-context performance improves monotonically with pre-training progression, even when controlling for identical model architecture and a fixed computational budget during the context extension stage. This suggests that long-context performance is intrinsically coupled with the general modeling ability of the base model. Consequently, a rigorous architectural comparison must control for this confounding variable by aligning base model loss, rather than merely aligning training steps.

2. Architectural Superiority under Controlled Settings. Guided by the principle above, we benchmark Engram against the MoE baseline. When controlling for base capability, the efficiency gains of the Engram module become evident:

- **Iso-Loss Setting (46k vs. Baseline):** This setting strictly isolates architectural efficiency. When comparing Engram-27B (46k) against the fully trained MoE-27B (50k)—models aligned on pre-training loss—Engram demonstrates significant gains. Specifically, it outperforms the baseline on complex retrieval tasks (e.g., Multi-Query NIAH: 97.0 vs. 84.2; VT: 87.2 vs. 77.0).
- **Iso-FLOPs Setting (50k vs. Baseline):** Under the standard iso-compute budget, Engram-27B (50k) further widens this gap, establishing the highest performance across the board.
- **Extreme Setting ($\approx$ 82% Compute):** Even the early-stopped Engram-27B (41k) remains highly competitive against the fully trained MoE-27B (50k). It matches the baseline on LongPPL and surpasses it on RULER, underscoring the intrinsic superiority of the Engram architecture.

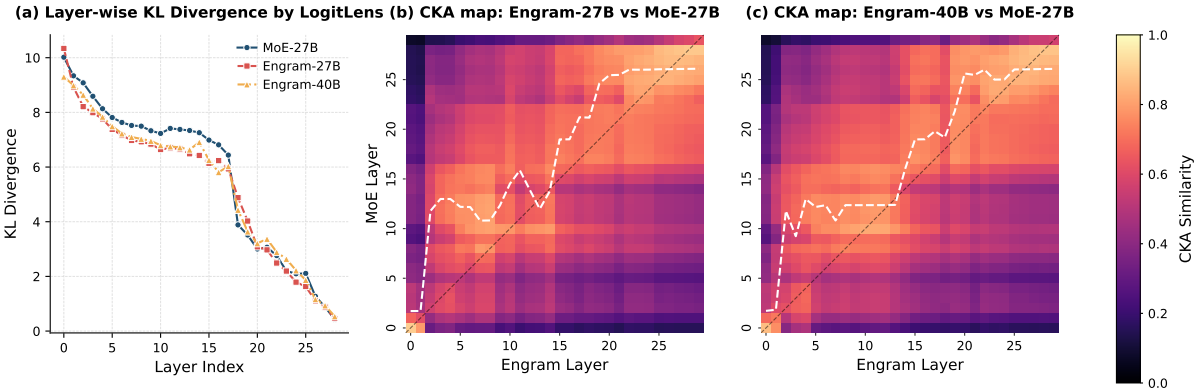

Figure 4 | **Analysis of representational alignment and convergence speed.** (a) Layer-wise KL Divergence via LogitLens (nostalgebraist, 2020). The consistently lower divergence in early layers indicates that Engram accelerates prediction convergence. (b-c) Similarity heatmap computed by CKA (Kornblith et al., 2019). The distinct upward shift of the high-similarity diagonal demonstrates that =Engram's shallow layers are functionally equivalent to deeper layers of the MoE model, effectively increasing the model's depth.

6. Analysis

In this section, we investigate the internal mechanisms of Engram, including its effective depth (Section 6.1), core module design (Section 6.2), and parametric sensitivity (Section 6.3). Additionally, we evaluate the inference throughput with offloading (Section 6.4) and conclude with a case study (Section 6.5).

6.1. Is Engram functionally equivalent to increasing the model's depth?

Current LLMs lack a dedicated knowledge lookup primitive and they rely on computation to simulate memory recall. As shown in Table 3, to recognize the entity `"Diana, Princess of Wales"`, an LLM must consume multiple layers of Attention and FFNs to progressively compose features (Ghandeharioun et al., 2024; Jin et al., 2025; Li and Subramani, 2025), a process that could theoretically be identified via a knowledge lookup operation.

Given this, we posit that by equipping the model with an explicit knowledge lookup capability, Engram effectively mimics an increase in model depth by relieving the model of the early stages of feature composition. To validate this hypothesis, we employ two mechanistic interpretability tools: LogitLens (Belrose et al., 2023; nostalgebraist, 2020) and Centered Kernel Alignment analysis (CKA) (Davari et al., 2023; Kornblith et al., 2019).

6.1.1. Accelerated Prediction Convergence

We first analyze the evolution of predictions across layers using LogitLens (nostalgebraist, 2020). By projecting each intermediate layer's hidden state with the final LM Head, we compute the Kullback–Leibler divergence (Kullback and Leibler, 1951) between the intermediate output distribution and the model's final output distribution. This metric quantifies how close a latent representation is to being "prediction-ready" (Belrose et al., 2023; Csordás et al., 2025).

Figure 4 (a) reports the layer-wise KL divergence. Compared to the MoE baseline, both Engram variants exhibit systematically smaller KL divergence, with the most pronounced gap

Table 3 | **Entity resolution example reproduced from** Ghandeharioun et al. (2024). This table illustrates how LLMs gradually integrate context tokens through layers of attention and FFNs to construct the internal representation of the entity: `"Diana, Princess of Wales"`. The "Latent State Translation" column displays the automatically generated text for the last token: `"Wales"` by PatchScope (Ghandeharioun et al., 2024), while the "Explanation" column presents the manual interpretation provided by the original authors.

Layer	Latent State Translation	Explanation
1-2	: Country in the United Kingdom	**Wales**
3	: Country in Europe	**Wales**
4	: Title held by female sovereigns in their own right or by queens consort	**Princess of Wales** (unspecific)
5	: Title given to the wife of the Prince of Wales (and later King)	**Princess of Wales** (unspecific)
6	: Diana, Princess of Wales (1961-1997), the first wife of Prince Charles, Prince of Wales, who was famous for her beauty and humanitarian work	**Diana, Princess of Wales**

appearing in the early blocks. The steeper descent in the Engram curves indicates that the model finishes feature composition much faster. This observation aligns with our hypothesis: by accessing external knowledge explicitly, Engram reduces the computational steps required, thereby reaching high-confidence, valid predictions earlier in the network hierarchy.

6.1.2. Representational Alignment and Effective Depth

To further investigate whether Engram layers semantically correspond to deeper layers of the baseline, we employ Centered Kernel Alignment (CKA), a widely established metric for comparing representational structures (Kornblith et al., 2019; Kriegeskorte et al., 2008). Given two sets of representations X and Y (e.g., activations from different models or layers), CKA is defined as:

$$\mathrm{CKA}(K, L) = \frac{\mathrm{HSIC}(K, L)}{\sqrt{\mathrm{HSIC}(K, K)\mathrm{HSIC}(L, L)}} \tag{8}$$

where $K = XX^\top$ and $L = YY^\top$ denote the Gram matrices (using a linear kernel) and HSIC is Hilbert-Schmidt Independence Criterion (Gretton et al., 2005). We employ a minibatch implementation with an unbiased estimator of HSIC (Davari et al., 2023) and evaluate on the Few-NERD dataset (Ding et al., 2021), extracting hidden states corresponding to the final token of named entities.

To rigorously quantify the layer-wise correspondence, we first compute the pairwise CKA similarity matrix $S \in [0, 1]^{L \times L}$, where L is the number of layers. We then introduce a soft alignment index a_j, defined as the weighted centroid of the top-k most similar MoE layers for each Engram layer j:

$$a_j = \frac{\sum_{i \in \mathcal{I}_j} S_{i,j} \cdot i}{\sum_{i \in \mathcal{I}_j} S_{i,j}}, \quad \text{where } \mathcal{I}_j = \operatorname*{argtop}_i k(S_{i,j}). \tag{9}$$

Here, $S_{i,j}$ denotes the similarity score between MoE layer i and Engram layer j. The index a_j

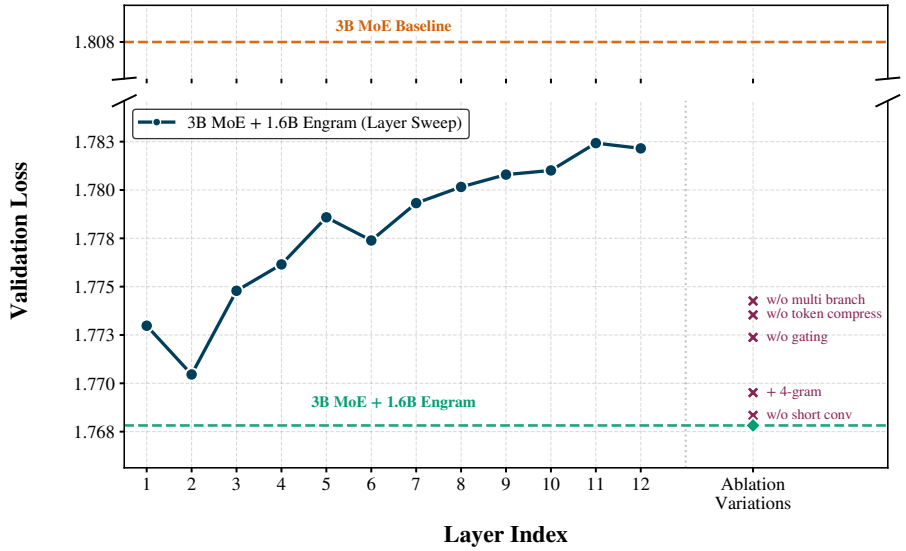

Figure 5 | **Architecture ablation results.** We compare the 3B MoE baseline against Engram variations in two settings: (1) Layer Sensitivity (dark blue curve): Sweeping the insertion depth of a single Engram module confirms that early injection (Layer 2) is optimal, whereas efficacy degrades in deeper layers. (2) Component Ablation (Right Markers): Removing sub-modules from the reference configuration demonstrates the importance of multi-branch integration, tokenizer compression, and context-aware gating.

serves as a robust proxy for the "effective MoE depth" corresponding to Engram layer j, utilizing top-k filtering (with $k = 5$) to mitigate low-similarity noise.

Figure 4 (b)–(c) visualize the similarity heatmaps overlayed with the soft alignment curve (dashed white line). We observe a distinct upward shift from the diagonal, meaning that $a_j > j$ for a wide range of layers. For instance, the representations formed at layer 5 of Engram-27B align most closely with those at approximately layer 12 of the MoE baseline.

The consistent off-diagonal shift, which aligns with the LogitLens results (Section 6.1.1), confirms that Engram achieves deeper representations at earlier layers. This validates our central hypothesis: by bypassing early-stage feature composition via explicit lookups, Engram is functionally equivalent to increasing the model's effective depth.

6.2. Structural Ablation and Layer Sensitivity

In this section, we ablate Engram under a controlled setting to investigate the effectiveness of each key module design. Unless otherwise specified, the backbone is a 12-layer 3B MoE model (0.56B activated parameters) trained for 100B tokens. Figure 5 reports validation loss. The dashed orange line denotes the 3B MoE baseline (Val Loss = 1.808).

Reference configuration. We augment the backbone with a fixed 1.6B-parameter Engram memory. Our reference model uses $\{2, 3\}$-grams and inserts Engram at Layers 2 and 6, achieving Val Loss = 1.768, a substantial improvement over the MoE baseline ($\Delta = 0.04$). All structural ablations below are defined relative to this reference.

Where should memory be injected? To study depth sensitivity, we keep the Engram budget fixed (1.6B) but consolidate it into a single Engram module, and sweep its insertion layer from 1 to 12 (dark blue "Layer Sweep" curve in Figure 5). This experiment exposes an inherent trade-off in Engram placement.

A placement trade-off. Injecting Engram early allows it to offload local pattern reconstruction before the backbone expends computational depth, aligning with the backbone's natural hierarchical processing (Ghandeharioun et al., 2024; Jin et al., 2025; Li and Subramani, 2025; Tenney et al., 2019). However, this incurs a cost in gating precision: early hidden states have not yet aggregated sufficient global context via attention, and the parallel branches lack the representational divergence required for fine-grained modulation (Xie et al., 2025; Zhu et al., 2025). Consequently, optimal placement requires balancing (i) offloading static local patterns early and (ii) utilizing stronger contextual queries for gating later.

The sweep shows that Layer 2 achieves the best single-layer performance (Val Loss = 1.770), outperforming Layer 1 and degrading as the insertion point moves deeper. This indicates that one round of attention is already sufficient to provide a meaningfully contextualized $\mathbf{h}_t$ for gating, while still being early enough to replace the backbone's bottom-layer local aggregation.

While Layer 2 is optimal under a single injection constraint, we find that dividing the same 1.6B memory into two smaller modules (achieved by reducing the embedding dimension d_{mem}) and placing them at Layers 2 and 6 performs even better (Val Loss = 1.768). This layered design reconciles the trade-off by combining early intervention with rich, late-stage contextual gating. More importantly, layered insertion also provides a practical system advantage, enabling better utilization of the memory hierarchy as discussed in Section 2.5.

Which components matter? Starting from the reference configuration, we ablate individual design choices while keeping the Engram parameter budget fixed. Results are denoted by markers in Figure 5. We find that three components yield the most significant gains: (i) branch-specific fusion within the multi-branch backbone, (ii) context-aware gating, and (iii) tokenizer compression. Removing any of these causes the largest regressions in validation loss. Specifically, for the "w/o multi branch" ablation, we retain the mHC backbone structure but replace the branch-specific gating with a single Engram fusion applied to the hidden states after the pre-mapping $\mathcal{H}^{pre}$ (Xie et al., 2025).

Other changes have smaller effects: removing the lightweight depthwise convolution only marginally degrades performance. Allocating capacity to 4-grams is slightly suboptimal under a fixed 1.6B budget—likely because it dilutes capacity from the more frequent 2/3-gram patterns—though we do not rule out that higher-order N-grams become beneficial at larger memory scales.

6.3. Sensitivity Analysis

To characterize the functional contribution of the Engram module, we evaluate the model by completely suppressing the sparse embedding output during inference while keeping the backbone unchanged. Crucially, this post-hoc ablation induces a **training–inference inconsistency**, potentially introducing noise in complex, mixed-capability tasks. Consequently, we prioritize the analysis of *Factual Knowledge* and *Reading Comprehension*—the two extremes of the sensitivity spectrum—which exhibit the highest signal-to-noise ratio under this stress test.

As shown in Figure 6, the results reveal a sharp functional dichotomy. Factual knowledge

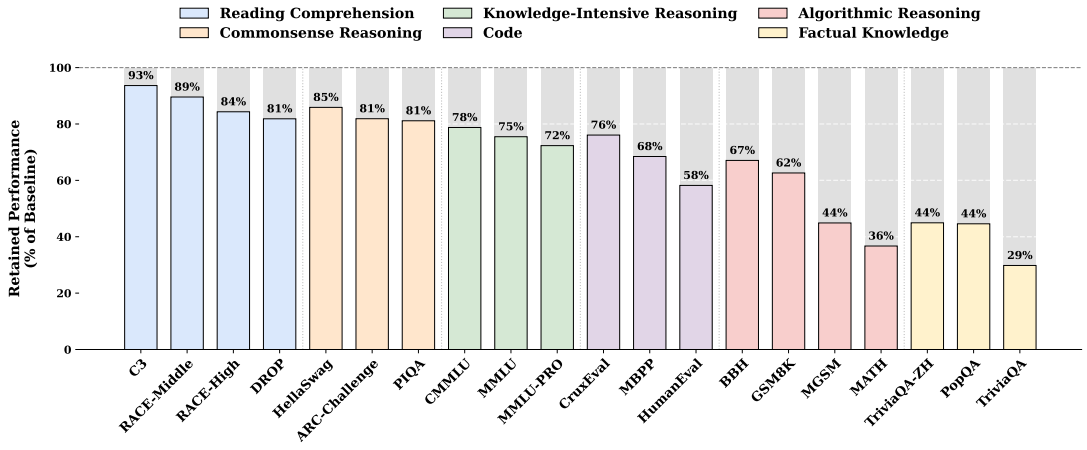

Figure 6 | **Retained performance under Engram ablation.** Factual knowledge relies heavily on the Engram module, whereas reading comprehension is largely preserved by the backbone.

benchmarks suffer a catastrophic collapse, retaining only 29–44% of the original performance (e.g., TriviaQA at 29%), confirming that the Engram module acts as the primary repository for parametric knowledge. Conversely, reading comprehension tasks are remarkably resilient, retaining 81–93% (e.g., C3 at 93%), suggesting that context-grounded tasks rely primarily on the backbone's attention mechanism rather than Engram.

6.4. System Efficiency

A pivotal system advantage of Engram over routing-based MoE is that its sparse activations are addressed by explicit, static hash IDs. This yields a strictly deterministic memory access pattern: indices for the next Engram lookup are fixed once the token sequence is known and can be computed before the corresponding layer executes.

Experimental Setup. We implemented an inference harness based on nano-vLLM[1]—a streamlined prototype of the industry-standard vLLM engine (Kwon et al., 2023). To obtain a clean latency baseline without the confounding communication patterns of Expert Parallel in MoE, we benchmark on two dense backbones (Dense-4B and Dense-8B). We insert a massive 100B-parameter Engram layer into the second Transformer block, with the entire embedding table resident in host DRAM. During inference, the system prefetches embeddings for the Engram layer asynchronously, overlapping the PCIe transfer with the computation of the first block.

Results. As detailed in Table 4, offloading a 100B-parameter embedding table incurs a negligible throughput penalty, peaking at only 2.8% on the 8B backbone. This confirms that the compute intensity of early dense blocks provides a sufficient temporal window to mask the retrieval latency. Crucially, the effective communication volume per step scales with the number of activated slots rather than the total embedding table size.

Crucially, this experiment serves as a conservative baseline. While the hierarchical design in Section 2.5 exploits Zipfian locality to cache frequent items in HBM, our experimental setup forces *all* retrievals to traverse the PCIe bus from host memory. The fact that this baseline

[1] https://github.com/GeeeekExplorer/nano-vllm

Table 4 | **End-to-end Inference Throughput**. We measure infernece throughput with a 100B-parameter Engram layer entirely offloaded to host memory.

Experimental Setup		
Hardware		NVIDIA H800
Workload		512 Sequences
Sequence Length		Uniform(100, 1024)
Throughput Results		
Base Model	**Configuration**	**Throughput (tok/s)**
4B-Dense	Baseline	9,031.62
	+ 100B Engram (CPU Offload)	8,858.28
8B-Dense	Baseline	6,315.52
	+ 100B Engram (CPU Offload)	6,140.02

retrieval strategy yields minimal overhead strongly suggests that a fully optimized, locality-aware implementation would incur negligible throughput penalty.

6.5. Case Study: Gating Visualization

In Section 2.3, we introduced the context-aware gating mechanism, designed to dynamically modulate the integration of retrieved static memory into the backbone. To empirically validate whether Engram behaves as intended, we visualize the gating scalar α_t of Engram-27B[2] across various samples in Figure 7.

The results demonstrate a distinct pattern of selectivity. The gating mechanism consistently activates (shown in red) upon completing local, static patterns. In English, we observe strong activations on multi-token named entities (e.g., "Alexander the Great", "the Milky Way") and formulaic phrases (e.g., "By the way", "Princess of Wales"). This behavior generalizes effectively across languages. In the Chinese examples, Engram identifies and retrieves distinct idiomatic expressions and historical entities, such as "Four Great Inventions" (四大发明) and "Zhang Zhongjing" (张仲景). These qualitative results confirm that Engram successfully identifies and handles stereotyped linguistic dependencies, effectively relieving the Transformer backbone from memorizing these static associations.

7. Related Work

N-gram Modeling and Embedding Scaling. Originating from Shannon's framework (Shannon, 1948), N-gram models rely on local history to predict tokens, traditionally employing smoothing techniques (Katz, 1987; Kneser and Ney, 1995) to mitigate data sparsity. Despite the paradigm shift toward neural architectures (Bengio et al., 2003) for capturing long-range dependencies, the computational efficiency of N-gram lookups has been preserved in modern representation learning, as exemplified by seminal works like FastText (Bojanowski et al., 2017).

[2]As detailed in our architecture setup, this model utilizes a mHC ($M = 4$) with Engram modules inserted at layers 2 and 15. Consequently, for any given token, the model computes a total of 8 distinct gating scalars. We observe that not every branch encodes interpretable activation patterns. For the clarity of this visualization, we select and display the gating values most strongly correlated with semantic pattern matching.

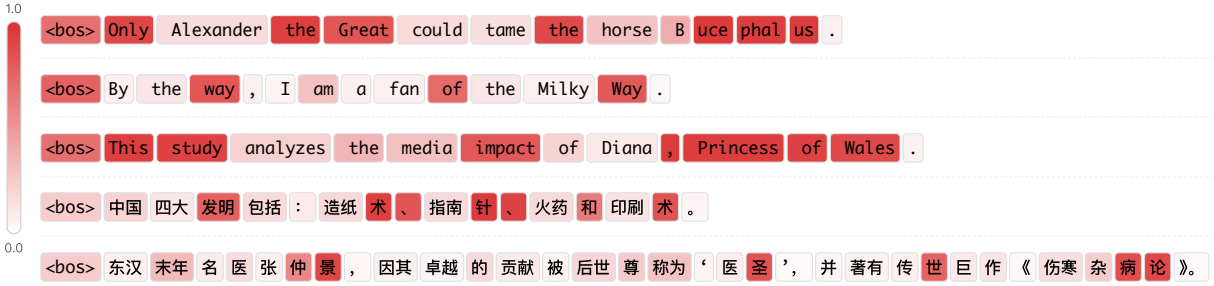

Figure 7 | **Visualization of the gating mechanism of Engram.** The heatmap intensity corresponds to the magnitude of the gating scalar $\alpha_t \in [0, 1]$, where darker red indicates stronger activation. Because Engram operates on suffix N-grams (here $N = 3$), a high activation on a specific token x_t implies that the preceding tokens culminating in that token (e.g., the phrase ending at t) are recognized as a static pattern effectively retrieved from memory.

Recently, this paradigm has resurged as *embedding scaling*. While architectures such as Per-Layer Embeddings (Team, 2025) and DeepEmbed (RWKV Team, 2025) expand capacity via massive tables, a distinct line of pioneering research—most relevant to our approach—integrates compositional N-gram structures directly into the representation space. SuperBPE (Liu et al., 2025) and SCONE (Yu et al., 2025) explicitly target high-frequency patterns: the former by merging multi-word expressions into "superword" tokens, and the latter via an auxiliary encoding model. In parallel, OverEncoding (Huang et al., 2025a) and Byte Latent Transformer (BLT) (Pagnoni et al., 2025) adopt hash N-gram embeddings to capture local dependencies at the token and byte levels, respectively. These studies collectively demonstrate the efficacy of scaling parameters through N-gram representations with minimal computational overhead. While these approaches offer significant gains in their respective settings, our work diverges fundamentally in two key dimensions.

- First, regarding modeling and evaluation protocols. Prior approaches often treat N-gram embeddings as external augmentations without validating their efficiency under strictly fair comparison protocols. For instance, SCONE (Yu et al., 2025) is inference-focused and relies on auxiliary modules that incur additional training FLOPs. Similarly, OverEncoding (Huang et al., 2025a) fails to yield meaningful improvements on sparse MoE backbones even under a non-isoparametric setting. In contrast, we treat conditional memory as a first-class modeling primitive instantiated via the carefully designed Engram module. By rigorously evaluating this design within our *Sparsity Allocation* framework, we demonstrate its clear advantage over strictly iso-parameter and iso-FLOPs MoE baselines.
- Second, from a system perspective, we advocate for algorithm-system co-design. Existing approaches place embeddings strictly at the input layer (Layer 0), which inherently serializes memory access and computation (Huang et al., 2025a; Yu et al., 2025). Engram, conversely, strategically injects memory into deeper layers to enable communication-computation overlap. Furthermore, by exploiting the inherent Zipfian distribution of N-grams, we could maximize the utility of the hardware memory hierarchy. This holistic design allows Engram to scale to massive parameters with negligible inference overhead.

Mixture-of-Experts. MoE architectures decouple model capacity from computational cost by conditionally activating a sparse subset of experts per token, a paradigm introduced by Shazeer et al. (2017). Subsequent innovations such as GShard (Lepikhin et al., 2020), BASE (Lewis et al., 2021), Switch Transformer (Fedus et al., 2022) and GLaM (Du et al., 2022) enabled super-

linear parameter scaling while maintaining constant inference costs. More recently, DeepSeek-MoE (Dai et al., 2024) demonstrated superior efficiency, significantly outperforming dense models with equivalent active parameters via fine-grained expert segmentation and shared expert isolation. Adopting this architecture, state-of-the-art models such as DeepSeek-V3 (Liu et al., 2024a) and Kimi-k2 (Team et al., 2025) have further pushed total parameters to hundreds of billions scale.

Memory Network. Research on memory-augmented networks aims to expand model capacity without a proportional increase in computational cost, broadly categorized into parametric and non-parametric approaches. Parametric memory methods, such as PKM (Lample et al., 2019), PEER (He, 2024), Selfmem (Cheng et al., 2023b), Memory+ (Berges et al., 2025) and UltraMem (Huang et al., 2025b,c), integrate large-scale, sparse key-value stores directly into the model layers, thereby significantly increasing capacity with negligible impact on FLOPs. Conversely, non-parametric memory approaches like REALM (Guu et al., 2020), RETRO (Borgeaud et al., 2022; Wang et al., 2023), and PlugLM (Cheng et al., 2023a) decouple knowledge storage from model processing, treating the external memory as an editable and scalable key-value store that allows the model to adapt to evolving information without retraining.

Mechanisms of Knowledge Storage. Parallel to capacity scaling, substantial research has scrutinized the internal mechanisms governing how Transformers encode and retrieve factual knowledge. The Feed-Forward Networks (FFNs) are widely hypothesized to function as Key-Value memories (Geva et al., 2021). Under this framework, the first layer acts as a pattern detector ("keys") while the second layer projects specific information into the residual stream ("values"). This modularity is evidenced by the identification of specific "knowledge neurons" responsible for storing distinct facts (Dai et al., 2022). Further validation is provided by causal tracing methodologies, which map the information flow of factual recall to specific FFN layers (Meng et al., 2022). These insights have enabled precise model editing algorithms such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), which allow for the direct update of factual associations without retraining. Moreover, investigations into internal representations, such as those in Othello-GPT (Li et al., 2023a), suggest that these storage mechanisms may facilitate the emergence of structured "world models" rather than mere statistical memorization.

8. Conclusion

In this work, we introduce **conditional memory** as a complementary sparsity axis to the prevailing conditional computation paradigm (MoE), aiming to resolve the inefficiency of simulating knowledge retrieval through dynamic computation. We instantiate this concept via Engram, a module that modernizes classic N-gram embeddings to enable scalable, constant-time $O(1)$ lookups for static patterns

By formulating the *Sparsity Allocation* problem, we uncover a U-shaped scaling law, demonstrating that a hybrid allocation of sparse capacity between MoE experts and Engram memory strictly outperforms pure MoE baselines. Guided by this law, we scale Engram to 27B parameters, achieving superior performance across diverse domains. Notably, while the memory module intuitively aids knowledge retrieval, we observe even larger gains in general reasoning, code, and mathematics.

Our mechanistic analysis reveals that Engram effectively "deepen" the network by relieving early layers from static reconstruction tasks, thereby freeing up attention capacity to focus

on global context and complex reasoning. This architectural shift translates into substantial improvements in long-context capabilities, as evidenced by performance gains in LongPPL and RULER. Finally, Engram advocates for infrastructure-aware efficiency as a first-class design principle. Its deterministic addressing allows for the decoupling of storage and compute, enabling the offloading of massive parameter tables to host memory with negligible inference overhead. We envision conditional memory functions as an indispensable modeling primitive for next-generation sparse models.

References

J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1409.0473.

N. Belrose, Z. Furman, L. Smith, D. Halawi, I. Ostrovsky, L. McKinney, S. Biderman, and J. Steinhardt. Eliciting latent predictions from transformers with the tuned lens. arXiv preprint arXiv:2303.08112, 2023.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. J. Mach. Learn. Res., 3:1137–1155, 2003. URL https://jmlr.org/papers/v3/bengio03a.html.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

V. Berges, B. Oguz, D. Haziza, W. Yih, L. Zettlemoyer, and G. Ghosh. Memory layers at scale. In Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025. OpenReview.net, 2025. URL https://openreview.net/forum?id=ATqGm1WyDj.

X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. arXiv preprint arXiv:2401.02954, 2024.

Y. Bisk, R. Zellers, J. Gao, Y. Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 7432–7439, 2020.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5:135–146, 2017.

S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, et al. Improving language models by retrieving from trillions of tokens. In International conference on machine learning, pages 2206–2240. PMLR, 2022.

T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In J. Eisner, editor, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 858–867, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/D07-1090/.

Y. R. Chao and G. K. Zipf. Human behavior and the principle of least effort: An introduction to human ecology. Language, 26:394, 1950. URL https://api.semanticscholar.org/CorpusID:10182796.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

X. Cheng, Y. Lin, X. Chen, D. Zhao, and R. Yan. Decouple knowledge from paramters for plug-and-play language modeling. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, Findings of the Association for Computational Linguistics: ACL 2023, pages 14288–14308, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.901. URL https://aclanthology.org/2023.findings-acl.901/.

X. Cheng, D. Luo, X. Chen, L. Liu, D. Zhao, and R. Yan. Lift yourself up: Retrieval-augmented text generation with self-memory. Advances in Neural Information Processing Systems, 36: 43780–43799, 2023b.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457, 2018.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261, 2025.

M. Constant, G. Eryiğit, J. Monti, L. Van Der Plas, C. Ramisch, M. Rosner, and A. Todirascu. Survey: multiword expression processing: a survey. Computational Linguistics, 43(4):837–892, 2017.

R. Csordás, C. D. Manning, and C. Potts. Do language models use their depth efficiently? arXiv preprint arXiv:2505.13898, 2025.

D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei. Knowledge neurons in pretrained transformers. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8493–8502, 2022.

D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. arXiv preprint arXiv:2401.06066, 2024.

M. Davari, S. Horoi, A. Natik, G. Lajoie, G. Wolf, and E. Belilovsky. Reliability of CKA as a similarity measure in deep learning. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/forum?id=8HRvyxc606.

DeepSeek-AI, A. Liu, B. Feng, B. Wang, B. Wang, B. Liu, C. Zhao, C. Dengr, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Xu, H. Yang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Chen, J. Yuan, J. Qiu, J. Song, K. Dong, K. Gao, K. Guan, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Zhu, Q. Chen, Q. Du, R. J. Chen, R. L. Jin, R. Ge, R. Pan, R. Xu, R. Chen, S. S. Li, S. Lu, S. Zhou, S. Chen, S. Wu, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Zheng, T. Wang, T. Pei, T. Yuan, T. Sun, W. L. Xiao, W. Zeng, W. An, W. Liu, W. Liang, W. Gao, W. Zhang, X. Q. Li, X. Jin, X. Wang, X. Bi, X. Liu, X. Wang, X. Shen, X. Chen, X. Chen, X. Nie, X. Sun, X. Wang, X. Liu, X. Xie, X. Yu, X. Song, X. Zhou, X. Yang, X. Lu, X. Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zhao, Y. Sun, Y. Li, Y. Wang, Y. Zheng, Y. Zhang, Y. Xiong, Y. Zhao, Y. He, Y. Tang, Y. Piao, Y. Dong, Y. Tan, Y. Liu, Y. Wang, Y. Guo, Y. Zhu, Y. Wang, Y. Zou, Y. Zha, Y. Ma, Y. Yan, Y. You, Y. Liu, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Huang, Z. Zhang, Z. Xie, Z. Hao, Z. Shao, Z. Wen, Z. Xu, Z. Zhang, Z. Li, Z. Wang, Z. Gu, Z. Li, and Z. Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, R. Jenatton, L. Beyer, M. Tschannen, A. Arnab, X. Wang, C. Riquelme Ruiz, M. Minderer, J. Puigcerver, U. Evci, M. Kumar, S. V. Steenkiste, G. F. Elsayed, A. Mahendran, F. Yu, A. Oliver, F. Huot, J. Bastings, M. Collier, A. A. Gritsenko, V. Birodkar, C. N. Vasconcelos, Y. Tay, T. Mensink, A. Kolesnikov, F. Pavetic, D. Tran, T. Kipf, M. Lucic, X. Zhai, D. Keysers, J. J. Harmsen, and N. Houlsby. Scaling vision transformers to 22 billion parameters. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pages 7480–7512. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/dehghani23a.html.

N. Ding, G. Xu, Y. Chen, X. Wang, X. Han, P. Xie, H. Zheng, and Z. Liu. Few-nerd: A few-shot named entity recognition dataset. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 3198–3213, 2021.

N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In International conference on machine learning, pages 5547–5569. PMLR, 2022.

D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In J. Burstein, C. Doran, and T. Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT

2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 2368–2378. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1246. URL https://doi.org/10.18653/v1/n19-1246.

S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural networks, 107:3–11, 2018.

B. Erman. The idiom principle and the open choice principle. Text-Interdisciplinary Journal for the Study of Discourse, 2000.

L. Fang, Y. Wang, Z. Liu, C. Zhang, S. Jegelka, J. Gao, B. Ding, and Y. Wang. What is wrong with perplexity for long-context language modeling? In The Thirteenth International Conference on Learning Representations.

W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research, 23(120):1–39, 2022.

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027, 2020.

T. Gao, A. Wettig, H. Yen, and D. Chen. How to train long-context language models (effectively). In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7376–7399, 2025.

A. P. Gema, J. O. J. Leang, G. Hong, A. Devoto, A. C. M. Mancino, R. Saxena, X. He, Y. Zhao, X. Du, M. R. G. Madani, et al. Are we done with mmlu? In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 5069–5096, 2025.

M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5484–5495, 2021.

A. Ghandeharioun, A. Caciularu, A. Pearce, L. Dixon, and M. Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In International Conference on Machine Learning, pages 15466–15490. PMLR, 2024.

A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In International conference on algorithmic learning theory, pages 63–77. Springer, 2005.

A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

A. Gu, B. Rozière, H. J. Leather, A. Solar-Lezama, G. Synnaeve, and S. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. In Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=Ffpg52swvg.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. Retrieval augmented language model pre-training. In International conference on machine learning, pages 3929–3938. PMLR, 2020.

J. Haber and M. Poesio. Polysemy—Evidence from linguistics, behavioral science, and contextualized language models. Computational Linguistics, 50(1):351–417, Mar. 2024. doi: 10.1162/coli_a_00500. URL https://aclanthology.org/2024.cl-1.10/.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

X. O. He. Mixture of a million experts. arXiv preprint arXiv:2407.04153, 2024.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In J. Vanschoren and S. Yeung, editors, Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, and B. Ginsburg. Ruler: What's the real context size of your long-context language models? In First Conference on Language Modeling.

H. Huang, D. Zhu, B. Wu, Y. Zeng, Y. Wang, Q. Min, and X. Zhou. Over-tokenized transformer: Vocabulary is generally worth scaling. In Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025. OpenReview.net, 2025a. URL https://openreview.net/forum?id=gbeZKej40m.

Y. Huang, Y. Bai, Z. Zhu, J. Zhang, J. Zhang, T. Su, J. Liu, C. Lv, Y. Zhang, Y. Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. Advances in Neural Information Processing Systems, 36:62991–63010, 2023.

Z. Huang, Y. Bao, Q. Min, S. Chen, R. Guo, H. Huang, D. Zhu, Y. Zeng, B. Wu, X. Zhou, et al. Ultramemv2: Memory networks scaling to 120b parameters with superior long-context learning. arXiv preprint arXiv:2508.18756, 2025b.

Z. Huang, Q. Min, H. Huang, Y. Zeng, D. Zhu, R. Guo, and X. Zhou. Ultra-sparse memory network. In The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net, 2025c. URL https://openreview.net/forum?id=zjeHLSiNv1.

M. Jin, Q. Yu, J. Huang, Q. Zeng, Z. Wang, W. Hua, H. Zhao, K. Mei, Y. Meng, K. Ding, F. Yang, M. Du, and Y. Zhang. Exploring concept depth: How large language models acquire knowledge and concept at different layers? In O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, editors, Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025, pages 558–573. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-main.37/.

K. Jordan, Y. Jin, V. Boza, J. You, F. Cesista, L. Newhouse, and J. Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In R. Barzilay and M. Kan, editors, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pages 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/V1/P17-1147. URL https://doi.org/10.18653/v1/P17-1147.

S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Trans. Acoust. Speech Signal Process., 35(3):400–401, 1987. doi: 10.1109/TASSP.1987.1165125. URL https://doi.org/10.1109/TASSP.1987.1165125.

D. P. Kingma. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In 1995 international conference on acoustics, speech, and signal processing, volume 1, pages 181–184. IEEE, 1995.

S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In International conference on machine learning, pages 3519–3529. PMlR, 2019.

N. Kriegeskorte, M. Mur, and P. A. Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. Frontiers in systems neuroscience, 2:249, 2008.

T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In E. Blanco and W. Lu, editors, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018, pages 66–71. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-2012. URL https://doi.org/10.18653/v1/d18-2012.

S. Kullback and R. A. Leibler. On information and sufficiency. The annals of mathematical statistics, 22(1):79–86, 1951.

W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th symposium on operating systems principles, pages 611–626, 2023.

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. H. Hovy. RACE: large-scale reading comprehension dataset from examinations. In M. Palmer, R. Hwa, and S. Riedel, editors, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, pages 785–794. Association for Computational Linguistics, 2017. doi: 10.18653/V1/D17-1082. URL https://doi.org/10.18653/v1/d17-1082.

G. Lample, A. Sablayrolles, M. Ranzato, L. Denoyer, and H. Jégou. Large memory layers with product keys. Advances in Neural Information Processing Systems, 32, 2019.

G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1VaB4cex.

P. Lennie. The cost of cortical computation. Current biology, 13(6):493–497, 2003.

D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. arXiv preprint arXiv:2006.16668, 2020.

M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer. Base layers: Simplifying training of large, sparse models. In M. Meila and T. Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 6265–6274. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/lewis21a.html.

H. Li, Y. Zhang, F. Koto, Y. Yang, H. Zhao, Y. Gong, N. Duan, and T. Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. In Findings of the Association for Computational Linguistics: ACL 2024, pages 11260–11285, 2024.

K. Li, A. K. Hopkins, D. Bau, F. B. Viégas, H. Pfister, and M. Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023a. URL https://openreview.net/forum?id=DeG07_TcZvT.

M. Li and N. Subramani. Echoes of bert: Do modern language models rediscover the classical nlp pipeline?, 2025. URL https://arxiv.org/abs/2506.02132.

R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. M. V, J. T. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries. Starcoder: may the source be with you! Trans. Mach. Learn. Res., 2023, 2023b. URL https://openreview.net/forum?id=KoFOg41haE.

W. Li, F. Qi, M. Sun, X. Yi, and J. Zhang. Ccpm: A chinese classical poetry matching dataset. arXiv preprint arXiv:2106.01979, 2021.

A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437, 2024a.

A. Liu, J. Hayase, V. Hofmann, S. Oh, N. A. Smith, and Y. Choi. SuperBPE: Space travel for language models. In Second Conference on Language Modeling, 2025. URL https://openreview.net/forum?id=lcDRvffeNP.

J. Liu, S. Min, L. Zettlemoyer, Y. Choi, and H. Hajishirzi. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. In First Conference on Language Modeling, 2024b. URL https://openreview.net/forum?id=u2vAyMeLMm.

A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9802–9822, 2023.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in gpt. Advances in neural information processing systems, 35:17359–17372, 2022.

K. Meng, A. S. Sharma, A. J. Andonian, Y. Belinkov, and D. Bau. Mass-editing memory in a transformer. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

T. Nguyen. Understanding transformers via n-gram statistics. Advances in neural information processing systems, 37:98049–98082, 2024.

nostalgebraist. interpreting gpt: the logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? Vision research, 37(23):3311–3325, 1997.

A. Pagnoni, R. Pasunuru, P. Rodriguez, J. Nguyen, B. Muller, M. Li, C. Zhou, L. Yu, J. E. Weston, L. Zettlemoyer, et al. Byte latent transformer: Patches scale better than tokens. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9238–9258, 2025.

B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, L. Derczynski, X. Du, M. Grella, K. K. GV, X. He, H. Hou, P. Kazienko, J. Kocon, J. Kong, B. Koptyra, H. Lau, J. Lin, K. S. I. Mantri, F. Mom, A. Saito, G. Song, X. Tang, J. S. Wind, S. Wozniak, Z. Zhang, Q. Zhou, J. Zhu, and R. Zhu. RWKV: reinventing rnns for the transformer era. In H. Bouamor, J. Pino, and K. Bali, editors, Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023, pages 14048–14077. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.936. URL https://doi.org/10.18653/v1/2023.findings-emnlp.936.

B. Peng, J. Quesnelle, H. Fan, and E. Shippole. Yarn: Efficient context window extension of large language models. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

S. T. Piantadosi. Zipf's word frequency law in natural language: A critical review and future directions. Psychonomic bulletin & review, 21(5):1112–1130, 2014.

O. Press, N. A. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

RWKV Team. Rwkv architecture history. https://wiki.rwkv.com/basic/architecture.html, 2025. Section "RWKV-V8's DeepEmbed", accessed 2025-12-09.

K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. Communications of the ACM, 64(9):99–106, 2021.

C. E. Shannon. A mathematical theory of communication. The Bell system technical journal, 27 (3):379–423, 1948.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538, 2017.

F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, D. Das, and J. Wei. Language models are multilingual chain-of-thought reasoners. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/forum?id=fR3wGCk-IXp.

J. Su, M. H. M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 568:127063, 2024. doi: 10.1016/J.NEUCOM.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

K. Sun, D. Yu, D. Yu, and C. Cardie. Investigating prior knowledge for challenging chinese machine reading comprehension. Transactions of the Association for Computational Linguistics, 8:141–155, 2020.

M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. Le, E. Chi, D. Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In Findings of the Association for Computational Linguistics: ACL 2023, pages 13003–13051, 2023.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

G. Team. Gemma 3n. 2025. URL https://ai.google.dev/gemma/docs/gemma-3n.

K. Team, Y. Zhang, Z. Lin, X. Yao, J. Hu, F. Meng, C. Liu, X. Men, S. Yang, Z. Li, et al. Kimi linear: An expressive, efficient attention architecture. arXiv preprint arXiv:2510.26692, 2025.

I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2019.

D. Tito Svenstrup, J. Hansen, and O. Winther. Hash embeddings for efficient word representations. Advances in neural information processing systems, 30, 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

B. Wang, W. Ping, P. Xu, L. McAfee, Z. Liu, M. Shoeybi, Y. Dong, O. Kuchaiev, B. Li, C. Xiao, et al. Shall we pretrain autoregressive language models with retrieval? a comprehensive study. In Proceedings of the 2023 conference on empirical methods in natural language processing, pages 7763–7786, 2023.

L. Wang, H. Gao, C. Zhao, X. Sun, and D. Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024a. URL https://arxiv.org/abs/2408.15664.

Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. Advances in Neural Information Processing Systems, 37:95266–95290, 2024b.

K. Whistler. Unicode standard annex #15: Unicode normalization forms. Unicode Standard Annex 15, The Unicode Consortium, July 2025. URL https://www.unicode.org/reports/tr15/tr15-57.html. Version Unicode 17.0.0, Revision 57. Accessed 2026-01-04.

G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Z. Xie, Y. Wei, H. Cao, C. Zhao, C. Deng, J. Li, D. Dai, H. Gao, J. Chang, L. Zhao, S. Zhou, Z. Xu, Z. Zhang, W. Zeng, S. Hu, Y. Wang, J. Yuan, L. Wang, and W. Liang. mhc: Manifold-constrained hyper-connections, 2025. URL https://arxiv.org/abs/2512.24880.

S. Yang, Y. Shen, K. Wen, S. Tan, M. Mishra, L. Ren, R. Panda, and Y. Kim. Path attention: Position encoding via accumulating householder transformations. arXiv preprint arXiv:2505.16381, 2025.

D. Yu, E. Cohen, B. Ghazi, Y. Huang, P. Kamath, R. Kumar, D. Liu, and C. Zhang. Scaling embedding layers in language models. arXiv preprint arXiv:2502.01637, 2025.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? In A. Korhonen, D. R. Traum, and L. Màrquez, editors, Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

B. Zhang and R. Sennrich. Root mean square layer normalization. Advances in neural information processing systems, 32, 2019.

W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen, and N. Duan. Agieval: A human-centric benchmark for evaluating foundation models. In Findings of the Association for Computational Linguistics: NAACL 2024, pages 2299–2314, 2024.

D. Zhu, H. Huang, Z. Huang, Y. Zeng, Y. Mao, B. Wu, Q. Min, and X. Zhou. Hyper-connections. In The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net, 2025. URL https://openreview.net/forum?id=9FqARW7dwB.

Appendices

A. Detailed Model Architecture and Hyper Parameters

	Dense-4B	MoE-27B	Engram-27B	Engram-40B
Total Params	4.1B	26.7B	26.7B	39.5B
Active Params		3.8B		
Total Tokens		262B		
Layers		30		
Dimension		2560		
Leading Dense Layers	-	1	1	1
Routed Experts	-	72	55	55
Active Experts	-	6	6	6
Shared Experts	-	2	2	2
Load Balancing Method	-	Loss Free (Wang et al., 2024a)		
Attention module		MLA (DeepSeek-AI et al., 2024)		
RoPE θ		10000		
mHC Expansion Rate		4		
Sequence Length		4096		
Vocab Size		129280		
Batch Size		1280		
Training Steps		50000		
Backbone Optimizer		Muon (Jordan et al., 2024)		
Embedding Optimizer		Adam (Kingma, 2014)		
Base Learning Rate		4e-4		
Lr Scheduler		Step Decay (Bi et al., 2024)		
Weight Decay		0.1		
Engram Dim d_{mem}	-	-	1280	1280
Engram Vocab Size	-	-	2262400	7239680
Engram Num Head	-	-	8	8
Engram Layer	-	-	[2,15]	[2,15]
Engram N-gram	-	-	[2,3]	[2,3]
Engram combine mHC	-	-	True	True
Engram tokenizer compression	-	-	True	True
Engram Conv Zero Init	-	-	True	True
Engram Lr Multipler	-	-	x5	x5
Engram Weight Decay	-	-	0.0	0.0
Engram Optimizer (Embed. only)	-	-	Adam (Kingma, 2014)	

Table 5 | Detailed model architecture information and training hyper parameters.

B. Full Benchmark Curves

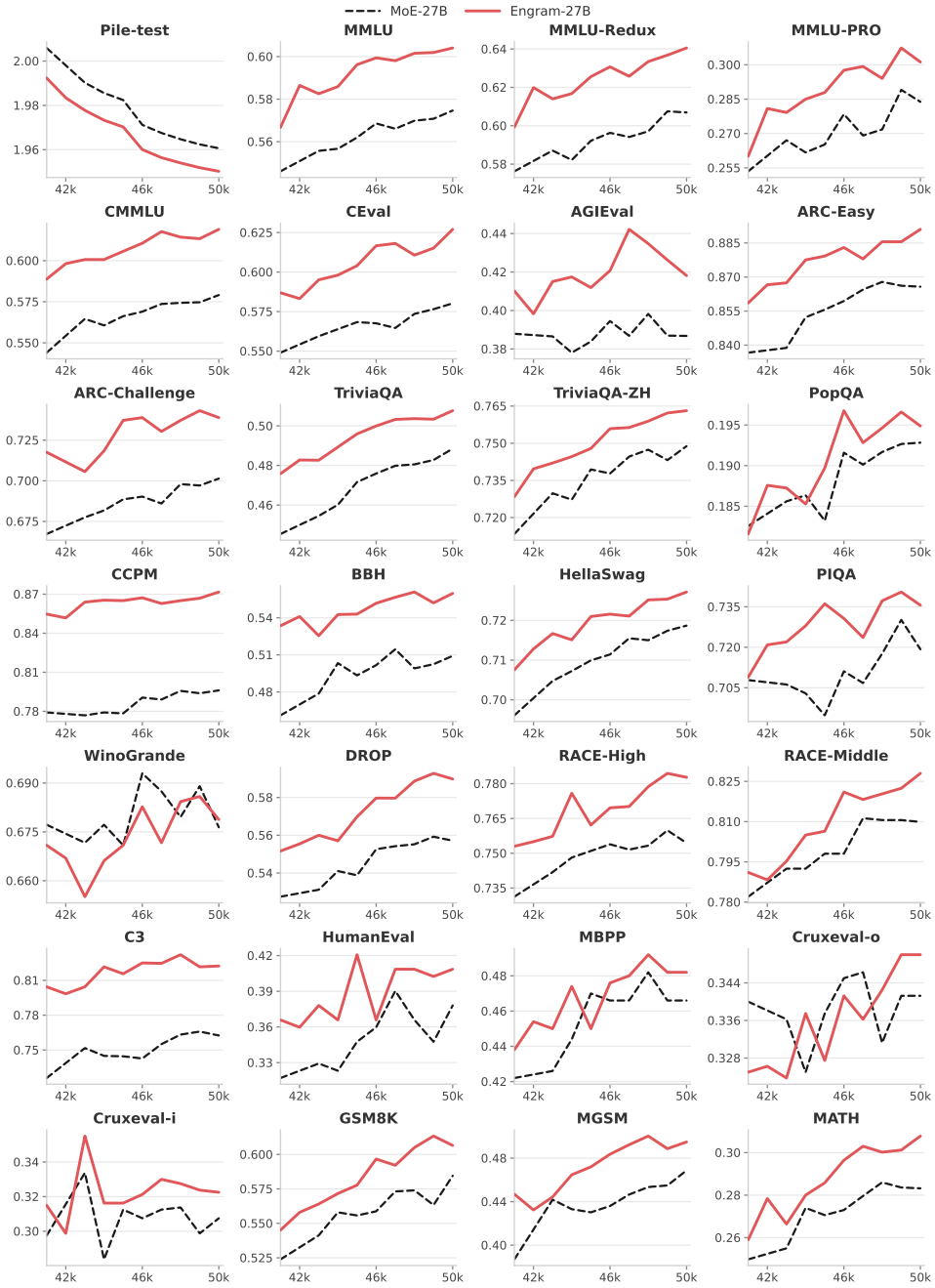

Figure 8 | Last 10k pre-training benchmark curve.

C. Case Study of Tokenizer Compression

Rank	Merge Count	Normalized Token	Original Tokens
1	163	’␣’	’\t’,’\n’,’\r’,’␣’,’␣␣’,’\n\n’,’␣␣␣’,’␣\n’,...
2	54	’a’	’A’,’a’,’␣a’,’␣A’,’á’,’ä’,’ã’,’ą’,’␣à’,’␣å’,’â’,...
3	40	’O’	’O’,’o’,’␣o’,’␣O’,’ó’,’ö’,’ô’,’õ’,’ő’,’ò’,...
4	35	’e’	’E’,’e’,’␣e’,’␣E’,’é’,’è’,’␣é’,’ę’,’ě’,’ê’,...
5	30	’i’	’I’,’i’,’␣I’,’␣i’,’í’,’ì’,’î’,’ī’,’ï’,...

Table 6 | The table illustrates Top-5 merged tokens by *Tokenizer Compression* and the overall compression ratio is 23.43% for our 128k tokenizer.