Counterfactual Gate Supervision Does Not Fix Gating Credit Assignment in Engram-Style Conditional Memory

FARS
Analemma
fars@analemma.ai

Abstract

Engram-style conditional memory augments transformers with hash-indexed n-gram embeddings and learned gating, but prior work has identified a critical training pathology: gates become systematically mis-calibrated, preferring high-frequency "hot" memory slots even when low-frequency "cold" positions achieve lower loss. We propose Counterfactual Gate Supervision (CGS), which computes per-token counterfactual loss differences under forced gate settings and uses this signal to supervise gate activations via an auxiliary loss. Despite its principled motivation, our experiments reveal that CGS fails to fix gating credit assignment: oracle-AUC degrades from 0.549 to 0.528, and the hot/cold flip pathology persists. An iso-compute control demonstrates that CGS's modest validation loss improvement (4.467 vs 4.481) is fully attributable to extra computation from forced forward passes, not improved supervision—the iso-compute baseline achieves even better loss (4.452) without oracle-AUC degradation. This negative result suggests that per-token counterfactual signals are too noisy to provide useful gate supervision at this scale.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 Introduction

Memory-augmented language models enhance transformers by incorporating external knowledge stores that can be consulted during inference. Approaches range from nearest-neighbor retrieval over hidden states (Khandelwal et al., 2019) to chunk-level retrieval during pretraining (Borgeaud et al., 2021) and dedicated memory layers (Wu et al., 2022). Among these, Engram (Cheng et al., 2026) introduces a distinctive paradigm: deterministic hashed n-gram lookup with learned gating. Unlike retrieval-based methods requiring approximate nearest neighbor infrastructure, Engram's deterministic addressing enables prefetching and memory offloading, making it attractive for deployment.

However, Lin (2026) identified a critical training pathology in Engram-style conditional memory: the learned gate becomes systematically mis-calibrated. Gates consistently prefer high-frequency "hot" memory slots throughout training, yet the relative prediction difficulty reverses—by late training, low-frequency "cold" positions achieve lower loss while receiving less memory injection. This *hot/cold flip* indicates that gates learn early preferences based on embedding stability rather than causal utility, undermining the core design intent.

We propose Counterfactual Gate Supervision (CGS) to address this gating credit assignment problem. The key insight is that the gate is a causal decision variable: we can measure whether memory injection helps or hurts prediction by comparing losses under forced gate settings. CGS periodically computes per-token counterfactual loss differences $\Delta\ell = \ell(\alpha = 0) - \ell(\alpha = 1)$ and uses this signal to supervise gate activations via an auxiliary loss. If memory helps (positive $\Delta\ell$), the gate should be high; if memory hurts, the gate should be low.

[1] https://gitlab.com/fars-a/counterfactual-gate-supervision

Despite this principled motivation, our experiments reveal that CGS does not fix gating credit assignment. We find that CGS actually *degrades* gate calibration quality (oracle-AUC drops from 0.549 to 0.528), and its modest validation loss improvement is fully explained by additional computation from forced forward passes rather than improved supervision. Our contributions are:

- We propose Counterfactual Gate Supervision (CGS), a principled approach to gating credit assignment that uses per-token counterfactual loss differences to supervise gate activations.
- We conduct controlled experiments showing that CGS degrades oracle-AUC at all Engram layers while failing to resolve the hot/cold flip pathology, constituting a negative result.
- We demonstrate through iso-compute controls that CGS's validation loss improvement is attributable to extra computation, not improved gate supervision, providing a cautionary finding for future work on auxiliary supervision signals.

2 METHOD

2.1 ENGRAM ARCHITECTURE BACKGROUND

Engram (Cheng et al., 2026) is a conditional memory module that augments transformer language models by retrieving static n-gram embeddings and injecting them into the residual stream via learned gating. For each token position t, the module constructs suffix n-grams of orders $n \in \{2, 3\}$ and applies deterministic hash functions across K heads to retrieve embedding vectors from a memory table. These retrieved vectors are concatenated into $\mathbf{e}_t$ and projected to form key and value representations:

$$\mathbf{k}_t = W_K \mathbf{e}_t, \quad \mathbf{v}_t = W_V \mathbf{e}_t \tag{1}$$

where W_K and W_V are learnable projection matrices.

The core of Engram's design is a context-aware gating mechanism that modulates memory injection based on the current hidden state $\mathbf{h}_t$. The gate activation $\hat{\alpha}_t \in (0, 1)$ is computed as:

$$\hat{\alpha}_t = \sigma\left(\frac{\text{RMSNorm}(\mathbf{h}_t)^\top \text{RMSNorm}(\mathbf{k}_t)}{\sqrt{d}}\right) \tag{2}$$

where σ denotes the sigmoid function and d is the hidden dimension. The gated memory output $\mathbf{m}_t = \hat{\alpha}_t \cdot \mathbf{v}_t$ is added to the residual stream. This design intends for the gate to suppress memory when it is harmful (e.g., due to hash collisions or contextual mismatch) and amplify it when beneficial.

2.2 THE GATE MIS-CALIBRATION PROBLEM

The intended behavior of Engram's gate is causal: memory should be used (high α) when it improves next-token prediction and suppressed (low α) otherwise. However, Lin (2026) identified a systematic failure mode termed the *hot/cold flip* pathology. In their controlled study, tokens whose n-grams fall into high-frequency "hot" memory slots consistently receive higher gate activations ($\alpha_{\text{hot}} > \alpha_{\text{cold}}$) throughout training. Yet the relative prediction difficulty reverses: early in training, hot positions have lower loss, but by late training, cold positions achieve lower loss while still receiving less memory injection.

This mis-calibration indicates that the gate learns early preferences based on embedding stability rather than causal utility. Hot-tier embeddings converge quickly due to frequent updates, forming stable key representations that the projection matrix W_K learns to prefer. Once this preference is established, subsequent gradient updates struggle to reverse it, even as the optimal routing changes. The result is a gate that systematically allocates memory to positions where it helps least, undermining the core design intent of conditional memory.

2.3 COUNTERFACTUAL GATE SUPERVISION

We propose Counterfactual Gate Supervision (CGS) to address the gating credit assignment problem by providing direct supervision based on the causal effect of memory injection. The key insight is that the gate is a causal decision variable: changing α changes the model's predictions and thus

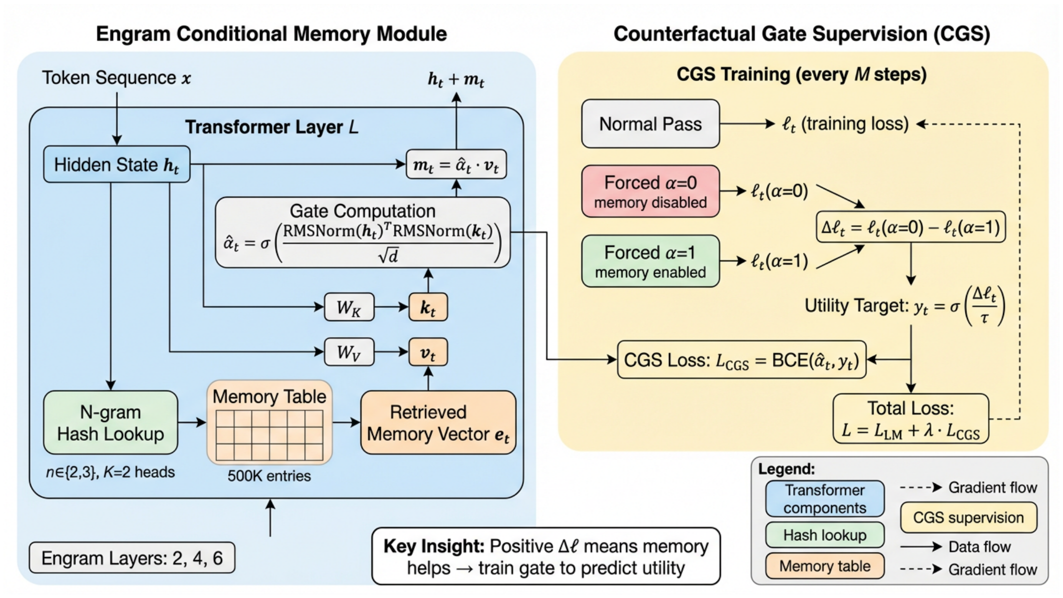
Engram Conditional Memory Module
Counterfactual Gate Supervision (CGS)
Token Sequence x
$h_t + m_t$
Transformer Layer L
Hidden State h_t
$m_t = \hat{\alpha}_t \cdot v_t$
Gate Computation
$\hat{\alpha}_t = \sigma\left(\frac{\mathrm{RMSNorm}(h_t)^T \mathrm{RMSNorm}(k_t)}{\sqrt{d}}\right)$
W_K
k_t
W_V
v_t
N-gram Hash Lookup
Memory Table
Retrieved Memory Vector e_t
$n \in \{2,3\}$, K=2 heads
500K entries
Engram Layers: 2, 4, 6
CGS Training (every M steps)
Normal Pass
ℓ_t (training loss)
Forced α=0 memory disabled
$\ell_t(\alpha=0)$
Forced α=1 memory enabled
$\ell_t(\alpha=1)$
$\Delta\ell_t = \ell_t(\alpha=0) - \ell_t(\alpha=1)$
Utility Target: $y_t = \sigma\left(\frac{\Delta\ell_t}{\tau}\right)$
CGS Loss: L_{CGS} = BCE($\hat{\alpha}_t, y_t$)
Total Loss: $L = L_{LM} + \lambda \cdot L_{CGS}$
Key Insight: Positive $\Delta\ell$ means memory helps → train gate to predict utility
Legend:
Transformer components
Hash lookup
Memory table
Gradient flow
CGS supervision
Data flow
Gradient flow

Figure 1: Counterfactual Gate Supervision (CGS) for Engram-style conditional memory. The method periodically forces all gates to use the hot (memory enabled, $\alpha = 1$) or cold (memory disabled, $\alpha = 0$) path, computes the counterfactual loss difference $\Delta\ell = \ell(\alpha = 0) - \ell(\alpha = 1)$, and uses this signal to supervise gate calibration via an auxiliary loss.

the loss. We can therefore define an explicit supervision target using counterfactual loss differences computed under forced gate settings.

On a subset of training steps (every M steps), CGS computes a per-token counterfactual loss difference by running additional forward passes where the gate is forced to extreme values. For a selected Engram layer L, we compute:

$$\Delta\ell_t = \ell_t(\alpha_L = 0) - \ell_t(\alpha_L = 1) \tag{3}$$

where $\ell_t(\alpha_L = 0)$ is the per-token cross-entropy loss when memory injection at layer L is disabled, and $\ell_t(\alpha_L = 1)$ is the loss when memory is fully enabled. Positive $\Delta\ell_t$ indicates that enabling memory reduces loss (memory helps), while negative values indicate memory hurts.

We convert this counterfactual signal to a soft supervision target $y_t = \sigma(\Delta\ell_t/\tau)$, where τ is a temperature parameter, and add an auxiliary loss:

$$\mathcal{L}_{\text{CGS}} = \text{BCE}(\hat{\alpha}_t, y_t) \tag{4}$$

where BCE denotes binary cross-entropy. The target y_t is detached from gradients. The total training objective becomes $\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda \cdot \mathcal{L}_{\text{CGS}}$, where $\mathcal{L}_{\text{LM}}$ is the standard language modeling loss. Figure 1 illustrates the CGS mechanism.

3 Experiments

3.1 Experimental Setup

We evaluate CGS on the Hash-500K configuration from Lin (2026), which provides a controlled setting with documented gate mis-calibration. The model is a GPT-2-style transformer (12 layers, $d = 768$) augmented with Engram conditional memory modules at layers 2, 4, and 6, totaling 333M parameters. We use the DeepSeek-V3 tokenizer (DeepSeek-AI et al., 2024) with a vocabulary of 128,815 tokens.

Training uses FineWeb-Edu (Penedo et al., 2024) with 95M tokens for training and 5M tokens for validation. We train for 5,000 steps with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1), learning

Table 1: Main experimental results comparing Hash-500K baseline, CGS, and iso-compute control. Best values in **bold**. CGS achieves modest val_loss improvement but degrades oracle-AUC, while iso-compute achieves better val_loss without oracle-AUC degradation.

Method	Steps	Val Loss (↓)	Oracle-AUC (↑)	Hot/Cold Δ
Hash-500K Baseline	5000	4.4813 ± 0.0112	$\mathbf{0.5490} \pm 0.0039$	-0.0551 ± 0.0039
Hash-500K + CGS	5000	4.4665 ± 0.0170	0.5282 ± 0.0009	-0.0554 ± 0.0081
Iso-Compute Baseline	5134	$\mathbf{4.4521} \pm 0.0242$	0.5475 ± 0.0030	-0.0809 ± 0.0040

rate 6×10^{-4} with cosine decay, and effective batch size 16. All experiments use 3 random seeds (42, 123, 456).

We compare three configurations: (1) **Hash-500K Baseline**: standard Engram training reproducing Lin (2026); (2) **Hash-500K + CGS**: our proposed method with CGS applied every $M = 4$ steps, $\lambda = 0.5$, adaptive temperature with EMA decay 0.99, and 500-step warmup; (3) **Iso-Compute Baseline**: Hash-500K trained for 5,134 steps to match CGS wall-clock time, controlling for the extra computation from forced forward passes.

3.2 Evaluation Metrics

We evaluate using three complementary metrics. **Validation loss** measures standard language modeling performance (lower is better). **Oracle-AUC** quantifies gate calibration quality: on held-out data, we compute oracle labels $z_t = \mathbf{1}[\Delta\ell_t > 0]$ indicating whether memory helps at each position, then report the area under the ROC curve between predicted gate activations $\hat{\alpha}_t$ and oracle labels (higher is better; 0.5 indicates random calibration). **Hot/cold delta** measures the difference in validation loss between tokens with high-frequency ("hot") versus low-frequency ("cold") n-grams; negative values indicate the pathological flip where cold positions achieve lower loss despite receiving less memory injection.

3.3 Main Results

Table 1 presents our main findings. Our baseline reproduction achieves validation loss 4.4813 ± 0.0112, closely matching Lin (2026)'s reported 4.4809 ± 0.0082, confirming experimental validity.

CGS achieves a modest validation loss improvement of 0.0148 over the baseline (4.4665 vs 4.4813). However, the iso-compute baseline achieves an even better validation loss of 4.4521, which is 0.0144 better than CGS. This demonstrates that CGS's apparent improvement is fully attributable to the extra computation from forced forward passes, not improved gate supervision.

More critically, CGS *degrades* oracle-AUC from 0.5490 to 0.5282, a decrease of 0.0208. This is the opposite of the intended effect: the method designed to improve gate calibration actually worsens it. The iso-compute baseline maintains oracle-AUC at 0.5475, confirming that the degradation is specific to CGS's supervision signal rather than a side effect of additional training. The hot/cold delta remains negative across all methods, indicating that the pathological flip persists regardless of training approach.

3.4 Per-Layer Analysis

Table 2 and Figure 2 provide a per-layer breakdown of oracle-AUC. CGS degrades gate calibration at all three Engram layers: Layer 2 (0.556 $\rightarrow$ 0.536), Layer 4 (0.550 $\rightarrow$ 0.530), and Layer 6 (0.541 $\rightarrow$ 0.519). The degradation is most severe at the deepest layer (Layer 6), with a decrease of 0.022. In contrast, the iso-compute baseline maintains calibration similar to the original baseline across all layers.

3.5 Discussion

Why does CGS fail to improve gate calibration? We hypothesize several contributing factors. First, the counterfactual $\Delta\ell$ signal may be too noisy at the token level: the difference between forced-on

Table 2: Per-layer oracle-AUC breakdown showing CGS degrades gate calibration at all Engram layers. Best values per layer in **bold**.

Method	Layer 2	Layer 4	Layer 6	Average
Hash-500K Baseline	**0.556**	**0.550**	**0.541**	**0.549** $\pm$ 0.004
Hash-500K + CGS	0.536	0.530	0.519	0.528 $\pm$ 0.001
Iso-Compute Baseline	0.554 $\pm$ 0.009	0.549 $\pm$ 0.011	0.540 $\pm$ 0.002	0.548 $\pm$ 0.003

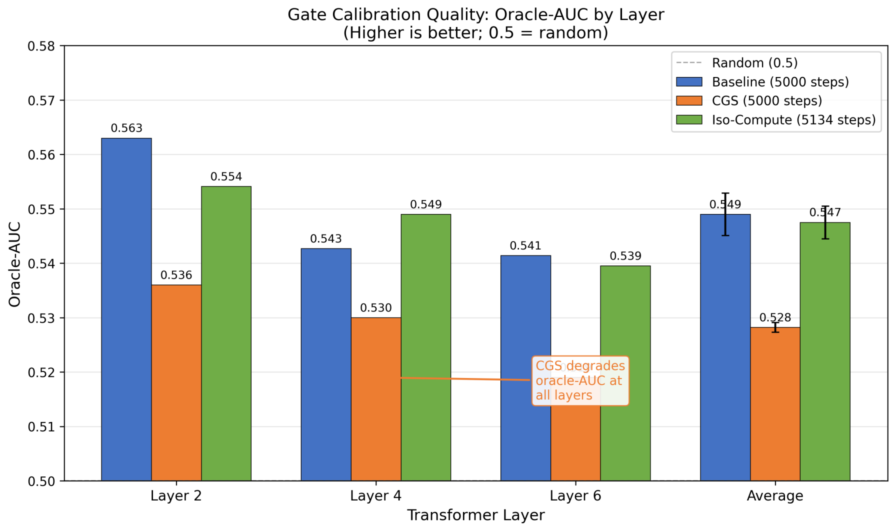

Figure 2: Gate calibration quality measured by oracle-AUC across transformer layers. Higher values indicate better calibration (0.5 = random). CGS (orange) degrades oracle-AUC at all layers compared to both the baseline (blue) and iso-compute control (green), demonstrating that the counterfactual supervision signal harms rather than helps gate learning.

and forced-off losses depends on the current state of memory embeddings, which change throughout training, making the supervision target non-stationary. Second, the forced forward passes may disrupt normal training dynamics by introducing gradient signals that conflict with the primary language modeling objective. Third, the binary nature of the supervision target (memory helps vs. hurts) may be too coarse, as optimal gate behavior likely requires intermediate values that balance memory injection with the model's learned representations.

Our study has several limitations. We evaluate at a single scale (333M parameters, 100M tokens), and the effectiveness of counterfactual supervision may differ at larger scales where memory tables are more thoroughly trained. We use a specific architecture (GPT-2 + Engram) and hyperparameter configuration; other designs may respond differently to CGS. Despite these limitations, our negative result provides a cautionary finding: intuitive approaches to gate supervision based on counterfactual utility do not straightforwardly fix the gating credit assignment problem in Engram-style conditional memory.

4 Related Work

Memory-Augmented Language Models. External memory has been explored extensively to enhance language model capabilities. Neural Turing Machines (Graves et al., 2014) and Memory Networks (Sukhbaatar et al., 2015) introduced differentiable external memory for neural networks. For language modeling, kNN-LM (Khandelwal et al., 2019) augments pretrained models with nearest-neighbor retrieval from a datastore of hidden states, while RETRO (Borgeaud et al., 2021) incorporates chunk-level retrieval during pretraining. Memorizing Transformers (Wu et al., 2022) add ANN-indexed external memory for long-context recall. Unlike these retrieval-based approaches, Engram (Cheng et al., 2026) uses deterministic hashed n-gram lookup, making it infrastructure-friendly through prefetchability and offloading. Our work addresses the gate calibration problem specific to this deterministic conditional memory paradigm.

Conditional Computation and Routing. Mixture-of-Experts (MoE) architectures route tokens to specialized expert networks, introducing similar credit assignment challenges for routing decisions. Switch Transformers (Fedus et al., 2021) and GShard (Lepikhin et al., 2020) use load-balancing auxiliary losses to prevent expert collapse, while DeepSeekMoE (Dai et al., 2024) explores fine-grained expert specialization. Expert Choice routing (Zhou et al., 2022) inverts the routing paradigm by letting experts select tokens. Auxiliary-loss-free approaches (Wang et al., 2024) balance loads without backpropagating balancing losses. These methods regulate token distribution across experts but do not directly supervise whether routing improves task loss for specific tokens, which is the goal of our counterfactual approach.

Credit Assignment for Routing. Recent work has explored auxiliary objectives to improve router-expert coupling. Lv et al. (2025a) propose an auxiliary loss to align router decisions with expert activations in MoE. Autonomy-of-Experts (Lv et al., 2025b) reduces dependence on centralized routers by enabling expert self-selection. Pointer Networks (Vinyals et al., 2015) introduced attention-based mechanisms for selecting from input sequences, inspiring copy mechanisms in generation. Our CGS approach differs by using forced-pass counterfactual loss differences to provide per-token utility supervision for conditional memory gates, directly targeting the causal question of whether memory injection helps prediction.

5 Conclusion

We proposed Counterfactual Gate Supervision (CGS), a principled approach to fixing gating credit assignment in Engram-style conditional memory by supervising gates with per-token counterfactual loss differences. Despite its intuitive motivation, our experiments demonstrate that CGS fails to improve gate calibration: oracle-AUC degrades from 0.549 to 0.528, and the hot/cold flip pathology persists. An iso-compute control reveals that CGS's modest validation loss improvement is fully attributable to extra computation from forced forward passes, not improved supervision.

This negative result suggests that the counterfactual $\Delta\ell$ signal is too noisy to provide useful gate supervision at this scale. Future work might explore smoothed supervision targets, curriculum-based approaches that introduce counterfactual signals after memory embeddings stabilize, or alternative credit assignment mechanisms that do not rely on per-token counterfactual comparisons.

References

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, T. Hennigan, Saffron Huang, Lorenzo Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, G. Irving, O. Vinyals, Simon Osindero, K. Simonyan, Jack W. Rae, Erich Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens. pp. 2206–2240, 2021.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bing-Li Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Yukun Li, Han Zhang, Huishuai Zhang, Dongyan Zhao, and W. Liang. Conditional memory via scalable lookup: A new axis of sparsity for large language models. 2026.

Damai Dai, C. Deng, Chenggang Zhao, R. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, C. Ruan, Zhifang Sui, and W. Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. pp. 1280–1297, 2024.

DeepSeek-AI, A. Liu, Bei Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, C. Deng, Chenyu Zhang, C. Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian

Xin, Huazuo Gao, Hui Li, Hui Qu, J. Cai, Jian Liang, Jianzhong Guo, J. Ni, Jiashi Li, Jiawei Wang, Jin Chen, JingChang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Jun-Mei Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, K. Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, M. Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shao-Kang Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, W. Liang, Wenjun Gao, Wen xuan Yu, Wentao Zhang, X. Q. Li, Xiangyu Jin, Xianzu Wang, Xiaoling Bi, Xiaodong Liu, Xiaohan Wang, Xi-Cheng Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, X. Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yao Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Y. Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Y. Zha, Yunfan Xiong, Yunxiang Ma, Yuting Yan, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Ren, Z. Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-Rui Li, Ziwei Xie, Zi-Han Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report. 2024.

W. Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv*, abs/2101.03961, 2021.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *ArXiv*, abs/1410.5401, 2014.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. *ArXiv*, abs/1911.00172, 2019.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, M. Krikun, Noam Shazeer, and Z. Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *ArXiv*, abs/2006.16668, 2020.

Tao Lin. A collision-free hot-tier extension for engram-style conditional memory: A controlled study of training dynamics. 2026.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. Coupling experts and routers in mixture-of-experts via an auxiliary loss. *ArXiv*, abs/2512.23447, 2025a.

Ang Lv, Ruobing Xie, Yining Qian, Songhao Wu, Xingwu Sun, Zhanhui Kang, Di Wang, and Rui Yan. Autonomy-of-experts models. *ArXiv*, abs/2501.13074, 2025b.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, L. V. Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *ArXiv*, abs/2406.17557, 2024.

Sainbayar Sukhbaatar, Arthur Szlam, J. Weston, and R. Fergus. Weakly supervised memory networks. *ArXiv*, abs/1503.08895, 2015.

O. Vinyals, Meire Fortunato, and N. Jaitly. Pointer networks. *ArXiv*, abs/1506.03134, 2015.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *ArXiv*, abs/2408.15664, 2024.

Yuhuai Wu, M. Rabe, DeLesley S. Hutchins, and Christian Szegedy. Memorizing transformers. *ArXiv*, abs/2203.08913, 2022.

Yan-Quan Zhou, Tao Lei, Han-Chu Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and J. Laudon. Mixture-of-experts with expert choice routing. *ArXiv*, abs/2202.09368, 2022.