SCALING LAWS FOR LLM MEMORY SPARSITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse memory architectures offer a route to expanding language-model capacity while accessing only a small fraction of parameters per token. Their finer access granularity promises greater sparsity than conventional mixture-of-experts (MoE), but how memory capacity should scale with active computation and training data remains insufficiently understood. We study how architecture and sparsity should vary with parameter and training-compute budgets. We introduce scaling laws for Product Key Memory (PKM) that capture how active parameters, memory capacity, and training tokens jointly determine loss, using 421 measured training endpoints from 44 configurations spanning dense Transformers, PKM, token-indexed memory (STEM), and MoE, with total parameter capacities reaching 21.9B and training budgets reaching 128B tokens.[1] The fitted laws reveal distinct regions of architectural advantage: PKM achieves the lowest predicted loss when capacity is abundant relative to compute, MoE is preferred in an intermediate regime, and dense Transformers become preferable when compute is abundant relative to capacity. Within PKM, increasing capacity at fixed compute favors greater sparsity, whereas increasing compute at fixed capacity favors more active parameters. Along a hardware-calibrated scaling path, PKM maintains lower predicted loss at higher optimal sparsity than MoE, permitting more training tokens and fewer operational FLOPs per token. A complementary roofline analysis estimates a $10.5\times$ PKM token-throughput advantage over MoE in the modeled mobile inference setting. Together, these findings characterize when sparse memory offers advantages in predicted loss and execution efficiency.

1 INTRODUCTION

Scaling laws have become central to the development of large language models (LLMs), turning smaller experiments into predictions that guide costly training runs (Kaplan et al., 2020; Hoffmann et al., 2022). By relating performance to capacity, data, and computation, they help identify promising model structures and training settings before committing resources to large-scale experiments (Ludziejewski et al., 2024; Bian et al., 2026). Among these design decisions, sparsity is a central architectural choice. Mixture-of-Experts (MoE) expands capacity while activating only selected experts per token, improving the trade-off between model quality and computation (Clark et al., 2022; Ludziejewski et al., 2024). It has become a standard architectural choice across leading open-weight and proprietary LLM families (Kimi Team, 2026; GLM-5 Team, 2026; Google DeepMind, 2026; Xu et al., 2026). Its success motivates exploring how far this separation of capacity from computation can extend.

An emerging direction is sparse parametric memory: large trainable tables from which each token accesses only a few entries. Addressing can use token identities or learned queries from hidden states (Sadhukhan et al., 2026; Sanseviero & Ballantyne, 2025; Lample et al., 2019). Such fine-grained retrieval offers a route to extreme sparsity and cheaper capacity expansion, with recent results showing competitive quality and lower memory-access costs than conventional MoE (Berges et al., 2024; Huang et al., 2024; 2026). This requires understanding how additional memory interacts with active computation and training data, and when its benefits survive hardware constraints.

[1] All code and model checkpoints will be released upon acceptance and deanonymization.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

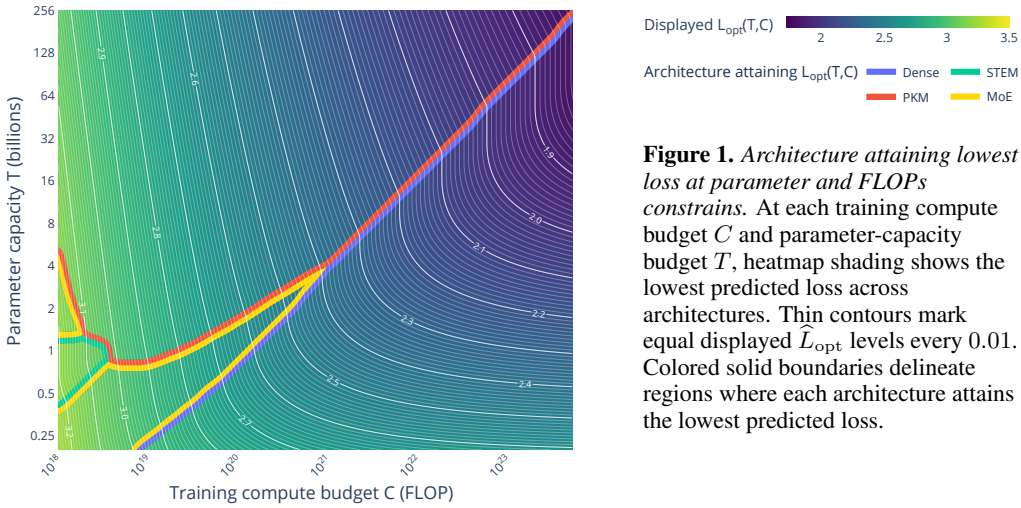


Figure 1. *Architecture attaining lowest loss at parameter and FLOPs constrains.* At each training compute budget C and parameter-capacity budget T, heatmap shading shows the lowest predicted loss across architectures. Thin contours mark equal displayed $\widehat{L}_{\text{opt}}$ levels every 0.01. Colored solid boundaries delineate regions where each architecture attains the lowest predicted loss.

We therefore study the scaling laws of sparse memory, focusing on Product Key Memory (PKM), whose learned retrieval supports varying bank size independently of vocabulary size and retrieval count. Using 421 measured training endpoints from 44 PKM, MoE, STEM, and dense Transformer configurations, we connect loss prediction to resource allocation and hardware efficiency. Our contributions are:

- **Scaling laws for sparse memory.** We formulate $L(N, D, M)$ to capture how memory capacity interacts with active parameters and training data, together with a compact two-variable law for dense and STEM models. The latter achieves lower held-out error than Chinchilla in all 32 reported comparisons and Farseer (Li et al., 2026) in 27. We further find that architecture-specific fits improve prediction: sharing coefficients between PKM and MoE worsens held-out MoE prediction in every matched test.
- **Resource-dependent architecture and sparsity selection.** We map architectures minimizing predicted loss across parameter and compute budgets (Figure 1): PKM is favored when parameter capacity is abundant relative to compute, MoE in an intermediate regime, and dense Transformers when compute is abundant relative to parameter capacity. Optimal PKM sparsity increases with the total parameter budget at fixed compute and decreases with compute at a fixed parameter budget. Along a hardware-calibrated scaling path, PKM maintains lower predicted loss at higher optimal sparsity than MoE, permitting more training tokens and fewer operational FLOPs per token.
- **Hardware efficiency of sparse memory.** We use a roofline model to show how retrieval granularity affects mobile inference when model capacity exceeds DRAM. PKM's fine-grained retrieval reduces flash traffic, yielding a modeled $10.5\times$ token-throughput advantage over MoE at a processing window of 512 tokens. In this setting, PKM sustains throughput as memory capacity grows, whereas MoE slows as more experts are added.

2 Related Work

Memory models. Parametric memory can use token-based addresses or learned content-based retrieval. Gemma 3n's Per-Layer Embeddings (PLE) (Sanseviero & Ballantyne, 2025) and STEM (Sadhukhan et al., 2026) use token-indexed embeddings; STEM replaces the FFN up-projection while retaining contextual gating. Engram (Cheng et al., 2026) extends deterministic lookup to n-grams. Product Key Memory (PKM) (Lample et al., 2019) learns queries from hidden states, allowing memory selection to vary with context. Memory layers at scale (Berges et al., 2024) and UltraMem/UltraMemV2 (Huang et al., 2024; 2026) develop this retrieval-based line through improvements to capacity, retrieval, and value processing. PEER (He, 2024) retrieves tiny experts, while hierarchical memories (Pouransari et al., 2026) separate long-tail knowledge from the active model. We focus on baseline PKM because its bank size can vary independently of vocabulary size and the number of retrieved values, providing a controlled setting for studying capacity scaling

with learned contextual addressing. STEM, dense Transformers, and MoE serve as comparative references.

Scaling laws. Power-law descriptions of language-model loss (Kaplan et al., 2020) underpin compute-optimal allocation of parameters and training tokens (Hoffmann et al., 2022). Farseer refines their joint loss surface by allowing token scaling to depend on model size (Li et al., 2026); related work models architectural choices for inference efficiency (Bian et al., 2026), downstream benchmark performance (Krajewski et al., 2026), and training hyperparameters such as learning rate and batch size (Li et al., 2025). Building on Chinchilla and Farseer, we introduce interactions among active parameters, memory capacity, and training tokens, and compare our related two-variable law against both baselines using held-out prediction on dense and STEM models.

MoE scaling laws. Routed-model laws separate parameter capacity from computation (Clark et al., 2022); later studies examine expert granularity and fine-grained architectures (Ludziejewski et al., 2024; Krajewski et al., 2025), optimal sparsity (Abnar et al., 2025), and joint dense–MoE scaling under memory and compute constraints (Ludziejewski et al., 2025). We extend this resource-allocation analysis to PKM, compare its optimal sparsity and predicted loss with MoE at matched parameter and compute budgets, and test whether a shared coefficient vector describes both architectures. Our analysis uses architecture-specific fits and complements loss-based comparisons with roofline models of communication and memory-access costs.

3 Experimental Setup

Our experiments quantify how Product Key Memory (PKM) (Lample et al., 2019) scales. We ask how increasing its sparsely accessed memory capacity changes predictive loss at a fixed active model size and training-token budget, and how this trade-off informs allocation under parameter and compute constraints. Scaling Transformers with Embedding Modules (STEM) (Sadhukhan et al., 2026) provides a second memory-model reference. Mixture-of-Experts (MoE) (Shazeer et al., 2017) provides comparative context from an architecture with sparsely activated expert parameters, while dense Transformers provide the baseline.

3.1 Model Families and Scaling Variables

We consider decoder-only Transformers with different feed-forward modules. The dense baseline applies its feed-forward parameters to every token. PKM uses product-key retrieval to access a subset of a learned value bank, whereas MoE routes tokens to a subset of feed-forward experts. STEM replaces the feed-forward up-projection with a token-indexed embedding lookup while retaining dense gating and down-projection. We vary PKM memory-bank size and MoE expansion across backbone sizes. For STEM, we vary backbone size under a fixed memory configuration rather than independently sweeping memory capacity.

Let L denote the measured final cross-entropy using a common evaluation data subset, N the active non-embedding parameter count, and D the training token budget. For PKM and MoE, we additionally use the capacity coordinate M, representing value-bank parameters for PKM and inactive expert parameters for MoE. We define the law-accounted total capacity and sparsity as $T = N + M$.

These coordinates separate active computation from additional stored capacity. Here, non-embedding accounting excludes vocabulary embeddings, but retains the internal memory modules being studied. In particular, the PKM coordinate T is an accounting proxy, not an exact measure of physical checkpoint or training-memory storage.

For the dense baseline, $M = 0$ and $T = N$. STEM is also fitted as $L(N, D)$, because its memory capacity is not varied independently of backbone size; consequently, its effect cannot be identified as a separate scaling-law input. This does not imply that STEM has no inactive parameters. PKM and MoE are instead fitted as $L(N, D, M)$.

3.2 Training Setup and Experimental Coverage

The training corpus is DCLM-Edu, and the models use the GPT-NeoX-20B tokenizer (Black et al., 2022). We distinguish a model configuration, which fixes the architecture and its capacity settings

(N, M), from an endpoint, which records its measured loss at a particular token budget (B). All endpoints belonging to one configuration form a model lineage.

Across PKM, MoE, STEM, and the dense Transformer baseline, we use a shared backbone and common training configuration, with architecture-specific memory and feed-forward modules. These settings and the architecture-specific hyperparameters are detailed in Appendix A. The experimental design targets approximately uniform coverage on logarithmic scales for model sizes, training-token budgets, and sparsity ratios where these quantities are varied.

The fitting dataset comprises 421 measured endpoints from 44 model configurations (192 PKM, 124 MoE, 57 dense, and 48 STEM endpoints). Active non-embedding parameters reach 2.0B, and training budgets reach 128B tokens. The largest PKM models reach 21.9B total parameters, with memory up to 139 times the active parameter count. MoE models reach 6.4B total parameters.

4 Scaling Laws for Memory Sparsity

Scaling laws are empirical functions that describe how model performance changes as computational resources are scaled. Dense-model laws commonly express loss in terms of parameter count N and training tokens D, because nearly all model parameters are active for every token. Sparse memory architectures break this correspondence: two models with the same active parameter count and training budget can have very different stored capacities. We therefore extend the usual $L(N, D)$ formulation with the independently varied inactive parameters coordinate M defined. We fit and validate the resulting $L(N, D, M)$ laws for PKM and MoE, then test whether a common scaling relationship can describe both sparse mechanisms.

4.1 Functional Form

The Chinchilla formulation (Hoffmann et al., 2022) decomposes loss into an irreducible term and separate model- and data-dependent terms,

$$\widehat{L}_{\mathrm{C}}(N, D) = E + AN^{-\alpha} + BD^{-\beta}. \tag{1}$$

Here, the amplitude and exponent of the data-dependent term are independent of model size. Farseer (Li et al., 2026) relaxes this assumption by allowing both to depend on N, expressing loss as $F_U(N) + B_{\mathrm{F}}(N)D^{-A_{\mathrm{F}}(N)}$.

Our formulation starts from the Chinchilla decomposition and progressively introduces the additional-capacity coordinate M according to practices established in literature ((Ludziejewski et al., 2025), (Li et al., 2026), (Ludziejewski et al., 2024), (Abnar et al., 2025), (Clark et al., 2022)). During this process, we also introduce an interaction between N and D, following the same general motivation as Farseer Li et al. (2026) while extending it to architectures with independently varied active and total capacity. The candidate formulas and fitting comparisons are described in Appendix B. For PKM and MoE, these extensions yield our central result: a memory-sparsity scaling-law family in which inactive capacity enters both the finite-model and data-dependent contributions,

$$\boxed{\begin{aligned}\widehat{L}(N, D, M) = \quad & \underbrace{AN^{-\alpha}M^{-\kappa}}_{U(N,M)\text{: finite-model term}} \\ & + \underbrace{BN^{-\lambda}M^{-\mu}D^{-(\beta+\zeta\log M+\xi\log N)}}_{V(N,D,M)\text{: data-dependent term}}.\end{aligned}} \tag{2}$$

We denote the token exponent by $q_D(N, M) = \beta + \zeta \log M + \xi \log N$. Unlike a dense $L(N, D)$ law, this form lets M influence the finite-model contribution, the amplitude of the data-dependent contribution, and its token-scaling exponent.

For the dense Transformer and STEM, memory is not independently varied in the fitting data. We therefore use the related six-coefficient family

$$\boxed{\widehat{L}(N, D) = AN^{-\alpha} + BN^{-\lambda}D^{-(\beta+\xi\log N)}.} \tag{3}$$

This family retains the model-dependent amplitude and token exponent but omits all independently fitted M terms; This is not obtained by substituting $M = 0$ into Equation 2. The learnable parameter vectors for the two formula families are

$$\boldsymbol{\theta}_{N,D,M} = (A, B, \alpha, \beta, \kappa, \lambda, \mu, \zeta, \xi), \qquad \boldsymbol{\theta}_{N,D} = (A, B, \alpha, \beta, \lambda, \xi).$$

The first vector is fitted separately for PKM and MoE, and the second is fitted separately for the dense Transformer and STEM. These architecture-specific fits do not share coefficients; a joint PKM–MoE law fitted to the pooled data is examined separately in Section 4.3.

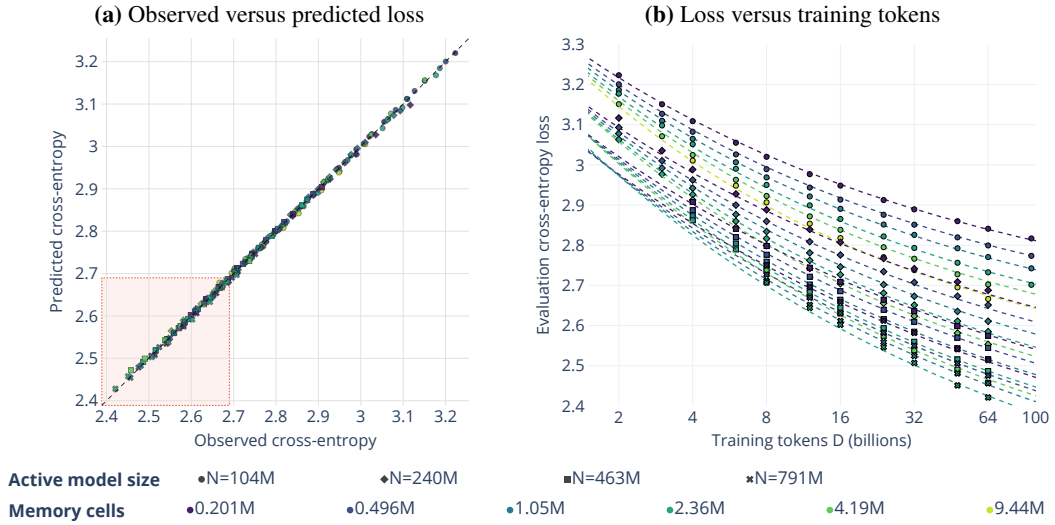


Figure 2. *PKM scaling-law diagnostics.* (a) Observed versus predicted cross-entropy under LLLO_{30}. The training subset has RMSE $4.805\text{e}{-3}$; the held-out subset has RMSE $5.860\text{e}{-3}$. The shaded square marks the validation envelope and the diagonal denotes exact prediction. (b) Measured endpoint losses (markers) and fitted curves (dashed lines). Marker shapes indicate active model size; colors indicate the number of cells in memory bank, not parameters.

4.2 Predictive Accuracy and Extrapolation

We compare 19 three-variable candidates on PKM, MoE, and the joint PKM–MoE data, and three two-variable candidates on the dense Transformer and STEM. The candidates progressively introduce memory dependence, model–data interactions, and alternative amplitude parameterizations. Appendix B gives the complete formulas and their evaluation results. Comparisons use the refitting and validity protocol in that appendix, rather than in-sample error alone.

Refitting every candidate on the complete data and each declared holdout produces 3,115 reported architecture–function–split fits. The evaluation suite tests prediction toward the lowest observed losses, withheld model configurations, size and memory-capacity families, future token budgets, and combinations of unseen capacities and training budgets, including the largest jointly supported size–capacity–token corner.

In the leave-lowest-losses-out test, denoted LLLO_p, we withhold the lowest-loss $p\%$ of endpoints, refit the law on the remaining higher-loss endpoints, and predict the withheld losses. We use $p \in \{10, 20, 30, 40, 50\}$. This tests prediction toward the best observed losses; it does not necessarily withhold entire model configurations. Figure 2a shows LLLO_{30} predictions of the PKM law (Equation 2), and Figure 2b its fitted loss curves against the training-token budget. The grouped and directional holdouts complement this test by withholding complete configurations or specified ranges of the scaling coordinates. The full-data and LLLO RMSE evaluations of the reported formulas are summarized in Table 1. Evaluation used in other scaling law papers: $\text{LLLO}_{20\%}$ RMSE = $19\text{e}{-3}$ for MoE and $\text{LLLO}_{20\%}$ RMSE = $15\text{e}{-3}$ for Dense ((Ludziejewski et al., 2024)), $\text{LLLO}_{10\%}$ RMSE = $3.9\text{e}{-3}$ for joint MoE and dense Transformer ((Ludziejewski et al., 2025)), full fit mean RMSE = $11\text{e}{-3}$ for dense Transformer downstream tasks evaluation ((Krajewski et al., 2026)), error ranging from $2.6\text{e}{-3}$ to $7.2\text{e}{-3}$ for dense Transformer beyond both unseen capacities and training budget ((Li et al., 2026)).

Table 1. *Full-data and LLLO RMSE of the reported scaling laws (*$\mathrm{e}{-3}$*); fitted coefficients are reported in Appendix Table 3.*

Architecture	Full-data	LLLO_{10}	LLLO_{20}	LLLO_{30}	LLLO_{40}	LLLO_{50}
PKM	4.58	4.32	5.01	5.86	7.39	9.27
MoE	5.80	5.36	6.08	8.49	9.11	9.79
Joint PKM–MoE	8.74	8.28	8.16	9.76	11.9	12.6
Dense Transformer	4.65	6.49	6.24	6.08	9.30	10.6
STEM	1.76	1.27	2.82	2.88	4.45	5.18

For the dense Transformer and STEM, our six-coefficient law (Equation 3) has lower held-out error than Chinchilla in all 32 reported architecture–metric–family comparisons, and lower error than Farseer in 27 of 32; the complete comparison is given in Appendix B, *Selection of the two-variable law*. These counts combine two architectures, MAE and RMSE, and eight held-out settings per architecture; they are not 32 independent experiments. Farseer achieves lower full-data error on both datasets, but uses nine coefficients rather than six. The results favor our two-variable law as a compact predictive description of these data, without implying it universally outperforms Farseer.

The selected law couples memory capacity and training data: at fixed active size, M changes both the finite-model component and the amplitude and exponent of the data-dependent component. Over the observed PKM domain, the fitted loss is locally less sensitive to M than to N or D. We therefore use the complete predicted loss surface for the allocation analysis in Section 5; Appendix B.1.1 derives the effective exponents and reports the detailed sensitivity analysis.

4.3 Joint PKM–MoE versus Architecture-Specific Scaling Laws

To test whether a common scaling surface describes PKM and MoE, we fit Equation 2 with a single coefficient vector on the pooled data (316 endpoints across 35 model configurations) and ask whether pooling improves MoE prediction, whose separately fitted law has higher LLLO_{30} error than PKM. Table 1 shows higher RMSE for the joint law than for the MoE-specific law on the full-data fit and all five LLLO evaluations, although their LLLO test sets are selected independently. We therefore perform a matched evaluation (Appendix B.4): for each LLLO_{10}–LLLO_{50} split, we refit the joint law on all 192 PKM endpoints plus the retained MoE endpoints and evaluate both laws on the same held-out MoE endpoints. All five refits converge and pass the endpoint-identity and train–test separation checks, and the joint law's RMSE is 1.35–2.05 times higher in every split. Pooling thus degrades MoE prediction under matched evaluation, providing direct evidence of different PKM and MoE scaling behavior; we therefore use architecture-specific laws both to compare the scaling behavior of PKM and MoE and for the analysis in Section 5.

5 Compute and Sparsity Optimality

The fitted scaling laws predict loss as a function of active parameters, training tokens, and inactive parameters. For each architecture and under fixed total-parameter and compute budgets, we identify the allocation between active and inactive parameters that minimizes predicted loss.

5.1 Sparsity at Fixed Parameter and Compute Budgets

At a fixed total capacity $T = N + M$, we express the sparsity between active parameters and memory through the active fraction $r = N/T$ and memory fraction $S = M/T = 1 - r$. We refer to S as sparsity under our parameter-accounting convention. For PKM, M counts the full value bank, so S is an accounting-based memory fraction rather than an exact fraction of physically inactive parameters. We approximate training compute by $C \approx 6ND$, following active-parameter FLOP accounting used in scaling studies (Kaplan et al., 2020), choosing r at fixed (T, C) determines all three inputs to the sparse-model scaling law:

$$N = rT, \qquad M = (1 - r)T, \qquad D = \frac{C}{6rT}. \tag{4}$$

Substitution into Equation 2 gives the one-dimensional objective

$$\widehat{L}_{\mathrm{alloc}}(r; T, C) := \widehat{L}\left(rT, \frac{C}{6rT}, (1 - r)T\right). \tag{5}$$

For the feasible active-fraction set $\mathcal{R}(T, C)$, we define

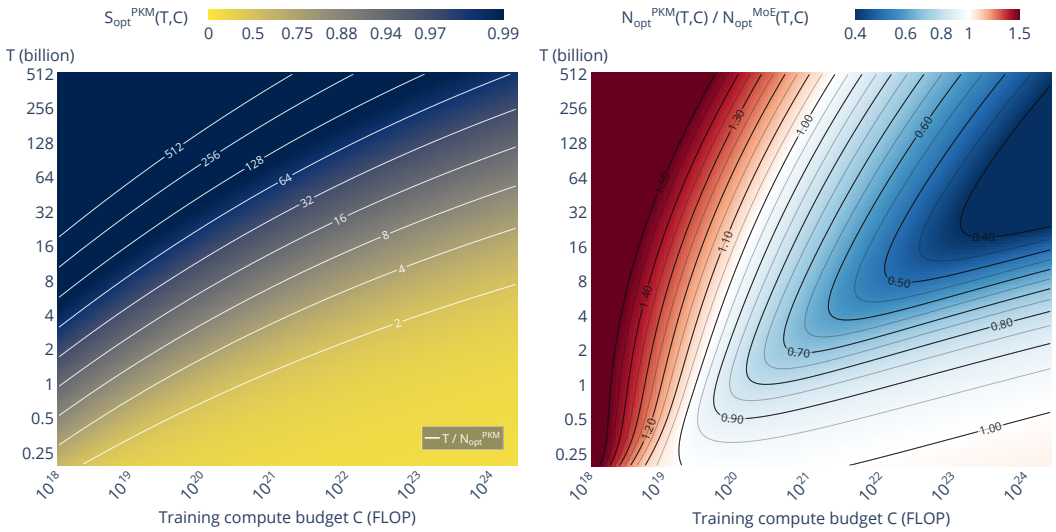
$$r_{\text{opt}}(T, C) \in \operatorname*{argmin}_{r \in \mathcal{R}(T,C)} \widehat{L}_{\text{alloc}}(r; T, C), \quad (6) \qquad \widehat{L}_{\text{opt}}(T, C) = \widehat{L}_{\text{alloc}}(r_{\text{opt}}(T, C); T, C)\,. \quad (7)$$

The optimal inactive fraction, active parameters, inactive parameters, and training tokens are

$$S_{\text{opt}}(T, C) = 1 - r_{\text{opt}}(T, C), \quad (8) \qquad N_{\text{opt}}(T, C) = r_{\text{opt}}(T, C)T, \quad (9)$$

$$M_{\text{opt}}(T, C) = [1 - r_{\text{opt}}(T, C)]\,T, \quad (10) \qquad D_{\text{opt}}(T, C) = \frac{C}{6r_{\text{opt}}(T, C)T}. \quad (11)$$

When an architecture superscript is shown below, it denotes the optimum obtained from that architecture's independently fitted scaling law. We perform this minimization numerically and separately for PKM and MoE, using their independently fitted coefficients. The expanded objective and numerical procedure are given in Appendix C.



(a) PKM optimal sparsity S **(b)** PKM / MoE optimal active parameters ratio

Figure 3. *Optimal sparsity in PKM and MoE.* Both panels show training compute C (FLOP) horizontally and parameter capacity T (billions) vertically. Both sparse architectures use the full available capacity in every displayed cell, so $N_{\text{opt}} + M_{\text{opt}} = T$. (a) Color shows the loss-minimizing PKM sparsity $S_{\text{opt}}^{\text{PKM}}(T, C)$; White contours show $T/N_{\text{opt}}^{\text{PKM}}$. The corresponding MoE optimal-sparsity map is shown in Appendix D, Figure 20. (b) Each architecture is optimized independently at matched budgets. Blue indicates fewer active parameters for PKM, red indicates fewer for MoE, and white denotes parity.

5.2 Optimal Sparsity in PKM and MoE

Figure 3a shows the predicted optimal PKM sparsity $S_{\text{opt}}^{\text{PKM}}(T, C)$ over parameter and compute budgets; the contours report the corresponding expansion ratio $T/N_{\text{opt}}^{\text{PKM}}$. At a fixed parameter budget, increasing compute favors a larger active fraction, and therefore lower sparsity and a smaller expansion ratio. Conversely, increasing parameter capacity at fixed compute favors allocating a larger fraction of parameters to inactive capacity, producing higher sparsity and a larger expansion ratio. The optimal sparsity therefore depends jointly on parameter and compute budgets, rather than on total model size alone.

Figure 3b compares the independently optimized sparsities through $N_{\text{opt}}^{\text{PKM}}(T, C)/N_{\text{opt}}^{\text{MoE}}(T, C)$ at matched parameter and compute budgets. Values below one indicate that PKM uses fewer active parameters than MoE, and therefore has higher sparsity and a larger expansion ratio; values above one indicate the converse. Because per-token inference compute scales with the active parameter count, a higher-sparsity allocation also lowers the cost of serving the model after training. PKM favors fewer active parameters when compute is high relative to parameter capacity, whereas MoE favors fewer

in the lower-compute regime. This ratio characterizes differences in the preferred sparsities, but does not compare the minimum losses achieved by the two architectures. At the highest compute-to-parameter ratios, the ratio approaches one because both architectures favor predominantly active parameters and approach the dense limit, $N_{\text{opt}} \approx T$.

5.3 Architecture Comparison across Resource Budgets

Figure 1 compares the architecture-specific lowest predicted losses $\widehat{L}^{(a)}_{\text{opt}}(T, C)$ of the dense Transformer, STEM, PKM, and MoE under matched parameter and compute budgets. At each budget, the heatmap shows $\min_a \widehat{L}^{(a)}_{\text{opt}}(T, C)$, while the colored boundaries identify the architecture attaining this minimum. This separates the question of how each architecture allocates its capacity from the question of which achieves the lowest predicted loss.

The comparison reveals distinct resource regimes. The dense Transformer is favored when compute is abundant relative to parameter capacity. As parameter capacity increases relative to compute, the preferred architecture shifts toward sparse alternatives: MoE occupies an intermediate regime, whereas PKM is favored across a broader capacity-rich, compute-constrained region. Thus, the architecture ranking depends on the specific combination of compute and parameter-capacity budgets, rather than defining a single ordering across all scales.

The sparsity analysis uses predicted loss together with the approximation $C \approx 6ND$, which counts active FLOPs but does not capture the reduction in hardware efficiency that accompany parameters sparsity increase. Changes to the memory module, expert design, or training configuration could also shift both the preferred sparsity and the architecture ranking. We therefore treat these results as loss-based comparisons of the studied architecture variants, distinct from the hardware considerations discussed in Section 5.5.

5.4 Optimal Sparsity along Coupled Resource Scaling

We next consider a coupled scaling scenario in which compute and parameter capacity increase together. Figure 4 evaluates a scaling path parameterized by n, with both the number of NVIDIA H100 80 GB GPUs and training duration proportional to n. We calibrate both budgets to the Qwen3-235B MoE configuration benchmarked with Megatron Core on 256 H100 GPUs in BF16 precision (NVIDIA, 2026; details in Appendix E.8). This configuration corresponds to approximately 0.92 billion model parameters per allocated GPU and a measured throughput of 320 TFLOP/s per GPU, or 32.3% Model FLOP Utilization (MFU). With $16n$ GPUs trained for n days, the resulting budgets are

$$C(n) = 4.424 \times 10^{20} n^2 \text{ FLOP}, \qquad T(n) = 14.69n \times 10^9 \text{ parameters}. \tag{12}$$

The compute budget covers both training and operation of the model. In addition to training on D tokens, each model processes $D_{\text{op}}(n) = n \times 100\text{B}$ tokens during its operational lifetime, at a cost of $2N$ FLOP per token. The budget is therefore split as

$$C(n) = 6ND + 2ND_{\text{op}}(n), \tag{13}$$

Figure 4a shows decreasing predicted loss for all four architectures, with PKM lowest across the entire displayed range, followed by MoE; the dense Transformer and STEM curves cross near $n = 16$. Along this path, the loss-optimal dense Transformer uses only about $1/16$ of $T(n)$, placing the path in the capacity-rich regime of Section 5.3, where sparse architectures are favored.

Figure 4b shows that the optimal ratio $T(n)/N_{\text{opt}}(T(n), C(n))$ increases with n and is consistently higher for PKM (35 to 160) than for MoE (33 to 84). Fewer active parameters also lower the operational cost $2ND_{\text{op}}$, which explains part of the sparse advantage over the dense Transformer: without this cost (Appendix D, Figure 21), PKM remains lowest, but its advantage over the dense Transformer at $n = 128$ shrinks from 0.062 to 0.016 and MoE falls behind the dense Transformer for $n \geq 16$, whereas PKM's advantage over MoE is largely unchanged.

The curves are extrapolations of the fitted laws under the assumed budget conversion; the hardware interpretation and its limitations are discussed in Section 5.5.

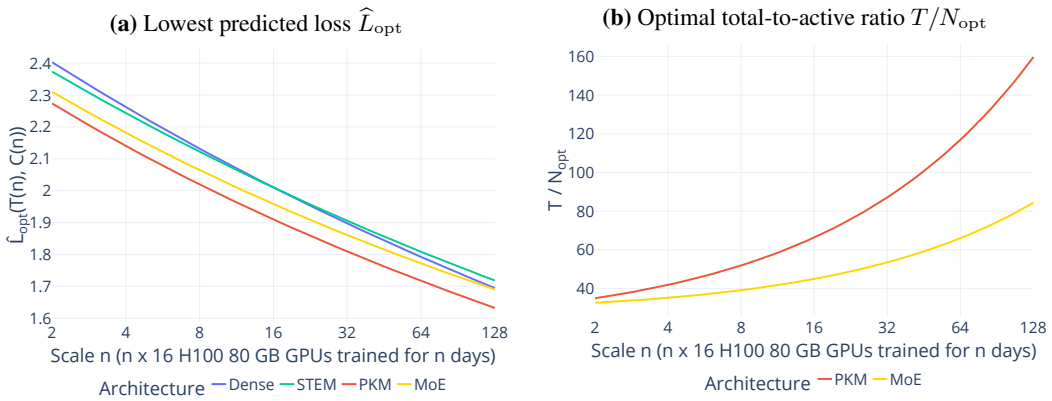


Figure 4. *Optimal loss and sparse allocation along coupled GPU-and-time scaling.* Both panels follow the budget path $C(n) = 4.424 \times 10^{20} n^2$ FLOP and $T(n) = 14.69n$ billion parameters, corresponding to $16n$ H100 80 GB GPUs trained for n days under the Megatron-Core-calibrated budget conversion. The compute budget includes the operational cost $2ND_{\text{op}}(n)$ with $D_{\text{op}}(n) = n \times 100$B tokens (Equation 13). Using the notation of Section 5.1, (a) each architecture's curve shows its lowest possible predicted loss $\widehat{L}_{\text{opt}}(T(n), C(n))$. (b) The PKM and MoE curves report $T(n)/N_{\text{opt}}(T(n), C(n))$ for their independently optimized sparsities.

5.5 Cost of Sparsity on Resource-Constrained Devices

We model inference token throughput of MoE and PKM on a mobile device with limited DRAM and flash storage using a roofline model as described in Appendix E.

On mobile devices, DRAM capacity caps N: the active-parameter count sets the static memory footprint for inference. Scaling model capacity beyond this budget pushes the additional parameters into flash (4 GB/s, roughly 17× slower than DRAM). Both PKM and MoE are flash memory bound, but MoE requires orders of magnitude more flash traffic. For the matched 3.93B-active-parameter setting in Figure 5, this is about a 260× gap per distinct load (78.6M versus 0.3M parameters; derivation in Appendix E). PKM is 10.5x faster at typical W=512 and still 2.6x faster at W=2048. Additionally, the PKM token throughput is independent of total memories, while MoE slows as the number of experts grows.

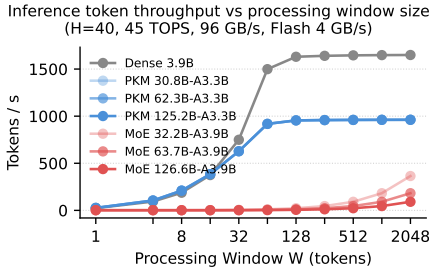


Figure 5. *Mobile inference: tokens/s versus processing window W.* PKM (blue) and MoE (red) at $T/N \in \{4, 8, 16\}$; dense (grey) for reference.

6 Conclusion and Discussion

We developed scaling laws coupling active parameters, memory capacity, and training data, and used them to derive resource-dependent allocation guidance. The fitted comparisons favor PKM when capacity is abundant relative to compute, while MoE and dense models become preferable in other regimes. Together with the roofline analysis, these results motivate evaluating sparse architectures through both loss scaling and memory-access costs.

These conclusions concern the architecture variants and training setup studied here; predictions beyond the measured range require validation at larger scales. Finer expert granularity (Ludziejewski et al., 2024), memory-layer expansion (Huang et al., 2024), or n-gram addressing (Cheng et al., 2026) could alter both optimal sparsity and architecture rankings. Extending the analysis to these designs and other training corpora is a natural next step.

Our compute approximation, $C \approx 6ND$, omits memory access, routing, communication, hardware utilization, and implementation efficiency. Equal FLOPs therefore need not imply equal training time, especially at high sparsity. The complementary roofline estimates depend on hardware and caching assumptions; validating these predictions with end-to-end measurements is necessary to establish when the predicted loss and throughput advantages translate into practical gains.

AI USE STATEMENT

Writing. We used generative AI tools to help review and correct grammar in the text. All AI-assisted text was reread and revised by the authors.

Retrieval and discovery. We used AI tools to survey the literature for additional relevant papers beyond the authors' initial background, and to analyze papers in bulk during the research to retrieve specific information, such as reported model configurations, fitting results, and hardware benchmark values. All AI-retrieved information was checked and confirmed by the authors in the original sources.

Research execution. We used AI tools to write plotting code and tables based on experimental results. The authors reviewed all generated code, verified that every plot and table uses the correct experimental data, and are responsible for all scientific claims in the paper. The research questions, experimental design, and model training were carried out by the authors.

Not applicable to this work: generating synthetic datasets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting with writing proofs.

REFERENCES

Samira Abnar, Harshay Shah, Dan Busbridge, Alaaeldin Mohamed Elnouby Ali, Josh Susskind, and Vimal Thilak. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv preprint arXiv:2501.12370*, 2025.

Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen-tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale. *arXiv preprint arXiv:2412.09764*, 2024.

Song Bian, Tao Yu, Shivaram Venkataraman, and Youngsuk Park. Scaling laws meet model architecture: Toward inference-efficient llms. In *International Conference on Learning Representations*, volume 2026, pp. 136664–136695, 2026.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé (eds.), *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, virtual+Dublin, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.9. URL https://aclanthology.org/2022.bigscience-1.9/.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Han Zhang, et al. Conditional memory via scalable lookup: A new axis of sparsity for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4968–4990, 2026.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pp. 4057–4086. PMLR, 2022.

GLM-5 Team. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Google DeepMind. Gemini 3.1 Pro model card. Google DeepMind, February 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Xu Owen He. Mixture of a million experts. *arXiv [cs.LG]*, July 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Zihao Huang, Qiyang Min, Hongzhi Huang, Defa Zhu, Yutao Zeng, Ran Guo, and Xun Zhou. Ultra-sparse memory network. *arXiv [cs.LG]*, November 2024.

Zihao Huang, Yu Bao, Qiyang Min, Siyan Chen, Ran Guo, Hongzhi Huang, Defa Zhu, Banggu Wu, Yutao Zeng, Siyuan Qiao, et al. Ultramemv2: Memory networks scaling to 120b parameters with superior long-context learning. In *International Conference on Learning Representations*, volume 2026, pp. 118101–118124, 2026.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kimi Team. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Jakub Krajewski, Marcin Chochowski, and Daniel Korzekwa. Scaling fine-grained MoE beyond 50B parameters: Empirical evaluation and practical insights. *arXiv [cs.LG]*, June 2025.

Jakub Krajewski, Amitis Shidani, Dan Busbridge, Sam Wiseman, and Jason Ramapuram. Revisiting the scaling properties of downstream metrics in large language model training. In *International Conference on Learning Representations*, volume 2026, pp. 123493–123533, 2026.

Guillaume Lample, Alexandre Sablayrolles, Marc' Aurelio Ranzato, Ludovic Denoyer, and Herve Jegou. Large memory layers with product keys. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/9d8df73a3cfbf3c5b47bc9b50f214aff-Paper.pdf.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part I, step law – optimal hyperparameter scaling law in large language model pretraining. *arXiv [cs.LG]*, March 2025.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Zhenyu Ding, Haoying Wang, Zili Wang, Shijie Xuyang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, et al. Predictable scale (part ii)—farseer: A refined scaling law in llms. *Advances in Neural Information Processing Systems*, 38:120612–120654, 2026.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling laws for fine-grained mixture of experts. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33270–33288. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/ludziejewski24a.html.

Jan Ludziejewski, Maciej Pióro, Jakub Krajewski, Maciej Stefaniak, Michał Krutul, Jan Małaśnicki, Marek Cygan, Piotr Sankowski, Kamil Adamczewski, Piotr Miłoś, and Sebastian Jaszczur. Joint moe scaling laws: mixture of experts can be memory efficient. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

NVIDIA. Scalable training of mixture-of-experts models with Megatron Core. Technical report, NVIDIA, 2026. URL https://arxiv.org/abs/2603.07685. arXiv:2603.07685.

Hadi Pouransari, David Grangier, C Thomas, Michael Kirchhof, and Oncel Tuzel. Pretraining with hierarchical memories: separating long-tail and common knowledge. In *International Conference on Learning Representations*, volume 2026, pp. 100596–100624, 2026.

Ranajoy Sadhukhan, Sheng Cao, Harry Dong, Changsheng Zhao, Attiano Purpura-Pontoniere, Yuandong Tian, Zechun Liu, and Beidi Chen. STEM: Scaling transformers with embedding modules. *arXiv [cs.LG]*, January 2026.

Omar Sanseviero and Ian Ballantyne. Introducing Gemma 3n: The developer guide. Google Developers Blog, June 2025. URL https://developers.googleblog.com/en/introducing-gemma-3n-developer-guide/.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. URL https://arxiv.org/abs/1701.06538.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

A Models and Training Hyperparameters

This appendix specifies the common backbone, architecture-specific modules, and training configuration used for the dense Transformer, product-key memory (PKM), Scaling Transformers with Embedding Modules (STEM), and mixture-of-experts (MoE) experiments.

A.1 SHARED H-FAMILY BACKBONE

All four families use the same decoder-only Qwen3 backbone, parameterized by an integer H. Both the number of decoder blocks and the number of query attention heads equal H. The hidden width is $d = 64H$, each attention head has width 64, grouped-query attention uses $H/2$ key/value heads, and the ordinary dense feed-forward width is $d_{\text{ff}} = 4d = 256H$. For example, H12 has 12 blocks, $d = 768$, and $d_{\text{ff}} = 3072$, whereas H24 has 24 blocks, $d = 1536$, and $d_{\text{ff}} = 6144$. The run names are H12, H16, H20, H24, and H32. For a sparse model, this name identifies the shared backbone.

The GPT-NeoX vocabulary is padded from 50,257 to 50,432 model entries. Input embeddings and the language-model head are not tied. Every block uses full causal self-attention with rotary position embeddings ($\theta = 10{,}000$), no sliding-window attention, and no attention dropout. Qwen3 RMSNorm ($\epsilon = 10^{-6}$) is applied before attention and before the feed-forward module, with a final RMSNorm after the decoder stack; queries and keys additionally receive per-head RMSNorm. Attention uses FlashAttention 2. The ordinary feed-forward modules are bias-free SwiGLUs,

$$\text{FFN}(x) = W_{\text{down}} \left[\text{SiLU}(W_{\text{gate}}x) \odot W_{\text{up}}x\right].$$

A.2 ARCHITECTURE-SPECIFIC MODULES

Dense Transformer. The dense model leaves the shared backbone unchanged: every block contains the standard width-$4d$ SwiGLU above.

Product-key memory. PKM replaces the dense feed-forward module in zero-based blocks $4, 8, \ldots, H-4$. This gives 2, 3, 4, 5, and 7 memory layers for H12, H16, H20, H24, and H32, respectively; all other blocks retain the standard width-$4d$ SwiGLU. In a memory block, the learned lookup replaces the dense up-projection:

$$\text{PKM}(x) = W_{\text{down}} \left[\text{SiLU}(W_{\text{gate}}x) \odot m(x)\right],$$

where the gate and retrieved memory vector both have width d. Each memory block owns an independent value bank.

The depth-two router uses two product-key factors of width b. A factor pair indexes one physical bank containing b^2 rows, each of width d. The capacity sweep uses $b \in \{448, 704, 1024, 1536, 2048, 3072\}$, or 0.200704M, 0.495616M, 1.048576M, 2.359296M, 4.194304M, and 9.437184M records per memory layer. If L_{mem} is the number of memory layers, their value parameters total $N_{\text{mem}} = L_{\text{mem}} b^2 d$.

Routing uses four memory heads. For each of the two factors, every head projects the block input to width $d/2 = 32H$, applies BatchNorm to the projected queries, and scores a shared table of b RMS-normalized keys with scaled dot products. The router retains the best 32 keys from each factor, forms their 32×32 Cartesian combinations, and keeps the best 32 complete addresses per head. A token therefore reads at most $4 \times 32 = 128$ value rows, and the per-head softmax-weighted values are summed into $m(x)$.

Scaling Transformers with Embedding Modules. STEM uses the same memory-modulated feed-forward construction but replaces learned routing by the deterministic lookup $m(x) = \mathcal{M}[\texttt{token_id}]$.

Each selected block owns a $V \times d_{\text{ff}}$ table with $V = 50{,}432$, and one full-width row is read for each token. The context-dependent gate $\text{SiLU}(W_{\text{gate}}x)$ modulates this context-independent token vector before the down-projection. STEM layers occupy zero-based indices $2, 4, \ldots, H-2$, every second block except the first and last, for a total of $H/2 - 1$ token-memory layers. Retrieved vectors receive no additional normalization.

Mixture of experts. MoE replaces the feed-forward module in every decoder block with token-choice MoE. Each expert is a complete bias-free SwiGLU of width $d_{\text{ff}} = 4d$, so one active expert has the same feed-forward width as the dense Transformer. We sweep $E \in \{2, 4, 8, 16\}$ experts per layer and always use top-1 routing; expert granularity is one, each layer holds E experts, and there is no shared expert.

A.3 Data, optimization, and training schedule

Models are trained from scratch on packed sequences of 2,048 tokens from the DCLM-Edu corpus, using next-token cross-entropy and bfloat16 arithmetic. The tokenizer uses the GPT-NeoX vocabulary described above (Black et al., 2022). Final cross-entropy is evaluated on the same fixed 50M-token held-out window for every model family.

Optimization uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1 for decay-eligible backbone parameters, and gradient-norm clipping at 1.0. PKM router parameters and PKM/STEM value tables are excluded from weight decay. The learning-rate schedule is warmup–stable–decay. Warmup duration, in billions of tokens, equals the dense-equivalent model size in billions: 0.183, 0.355, 0.620, 1.004, and 2.219B tokens for H12, H16, H20, H24, and H32. The stable phase is followed by a linear decay to zero over the final 20% of each run. Multiple training horizons are obtained from a shared stable-phase parent.

A.4 Learning-rate selection

For every model configuration, we tuned the peak learning rate on a base-2 logarithmic grid, $\eta = 2^{x_{1r}}$, and selected the value with the lowest loss at a training budget of 16B tokens. The selected learning rate was then used to train that configuration at all other token budgets. Training uses a continuously increasing batch size (Appendix A.5), which compensates for the increasing token budget and allows all training endpoints of a model configuration to share the same constant learning rate.

A.5 Batch-size schedule

The batch size increases continuously during training according to the Step Law optimal batch-size relation (Li et al., 2025),

$$B(D) = 0.58\, D^{0.571},$$

where B is the batch size in tokens and D is the current number of training tokens. Because Step Law finds the optimal batch size to depend primarily on D and to be largely invariant to model size, the same schedule is used for every model configuration.

A.6 Measured model configurations

Table 2 enumerates all 44 model lineages included in the scaling-law fits and all 421 measured endpoints associated with them. The table uses the non-embedding accounting conventions from Section 3.1.

Table 2: All model configurations and measured budgets in the fitting dataset. H defines the shared backbone; b is the PKM product-key factor width and E is the number of MoE experts per layer. `memc` gives millions of lookup records per memory layer and `memv` their vector width. N is the active non-embedding parameter count; M is the PKM value bank or inactive MoE expert count, with both columns given in billions of parameters and displayed to six decimal places. Each entry in the final column is a matched D/C pair, with D in billions of tokens and $C = 6ND$ in exaFLOPs. C is computed from the exact integer N and displayed to three decimal places. A dash denotes not applicable; STEM has physical token-memory parameters, but M is dashed because memory capacity was not an independently varied fitting coordinate.

Arch.	H	b	E	`memc`	`memv`	N	M	Measured D/C (B/EF)
PKM	12	448	–	0.200704	768	0.100009	0.308281	2/1.200, 3/1.800, 4/2.400, 6/3.600, 8/4.800, 12/7.201, 16/9.601, 24/14.401, 32/19.202, 48/28.803, 64/38.403, 96/57.605
PKM	12	704	–	0.495616	768	0.100402	0.761266	2/1.205, 3/1.807, 4/2.410, 6/3.614, 8/4.819, 12/7.229, 16/9.639, 24/14.458, 32/19.277, 48/28.916, 64/38.554, 96/57.832
PKM	12	1024	–	1.048576	768	0.100894	1.610613	2/1.211, 3/1.816, 4/2.421, 6/3.632, 8/4.843, 12/7.264, 16/9.686, 24/14.529, 32/19.372, 48/29.057, 64/38.743, 96/58.115
PKM	12	1536	–	2.359296	768	0.101680	3.623879	2/1.220, 3/1.830, 4/2.440, 6/3.660, 8/4.881, 12/7.321, 16/9.761, 24/14.642, 32/19.523, 48/29.284, 64/39.045, 96/58.568

Continued on next page

Table 2 (continued)

Arch.	H	b	E	memc	memv	N	M	Measured D/C (B/EF)
PKM	12	2048	–	4.194304	768	0.102467	6.442451	2/1.230, 3/1.844, 4/2.459, 6/3.689, 8/4.918, 12/7.378, 16/9.837, 24/14.755, 32/19.674, 48/29.510, 64/39.347
PKM	12	3072	–	9.437184	768	0.104039	14.495515	4/2.497, 6/3.745, 8/4.994, 12/7.491, 16/9.988, 24/14.982, 32/19.976, 48/29.963, 64/39.951
PKM	16	448	–	0.200704	1024	0.234615	0.616563	2/2.815, 3/4.223, 4/5.631, 6/8.446, 8/11.262, 12/16.892, 16/22.523, 24/33.785, 32/45.046, 48/67.569, 64/90.092
PKM	16	704	–	0.495616	1024	0.235402	1.522532	2/2.825, 3/4.237, 4/5.650, 6/8.474, 8/11.299, 12/16.949, 16/22.599, 24/33.898, 32/45.197, 48/67.796, 64/90.394
PKM	16	1024	–	1.048576	1024	0.236385	3.221225	2/2.837, 3/4.255, 4/5.673, 6/8.510, 8/11.346, 12/17.020, 16/22.693, 24/34.039, 32/45.386, 48/68.079, 64/90.772
PKM	16	1536	–	2.359296	1024	0.237958	7.247757	2/2.855, 3/4.283, 4/5.711, 6/8.566, 8/11.422, 12/17.133, 16/22.844, 24/34.266, 32/45.688, 48/68.532, 64/91.376
PKM	16	2048	–	4.194304	1024	0.239530	12.884902	4/5.749, 6/8.623, 8/11.497, 12/17.246, 16/22.995, 24/34.492, 32/45.990, 48/68.985, 64/91.980
PKM	20	448	–	0.200704	1280	0.455246	1.027604	4/10.926, 6/16.389, 8/21.852, 12/32.778, 16/43.704, 24/65.555, 32/87.407, 48/131.111, 64/174.814
PKM	20	704	–	0.495616	1280	0.456557	2.537554	4/10.957, 6/16.436, 8/21.915, 12/32.872, 16/43.829, 24/65.744, 32/87.659, 48/131.488, 64/175.318
PKM	20	1024	–	1.048576	1280	0.458195	5.368709	4/10.997, 6/16.495, 8/21.993, 12/32.990, 16/43.987, 24/65.980, 32/87.973, 48/131.960, 64/175.947
PKM	20	1536	–	2.359296	1280	0.460817	12.079596	4/11.060, 6/16.589, 8/22.119, 12/33.179, 16/44.238, 24/66.358, 32/88.477, 48/132.715, 64/176.954
PKM	20	2048	–	4.194304	1280	0.463438	21.474836	8/22.245, 12/33.368, 16/44.490, 24/66.735, 32/88.980, 48/133.470, 64/177.960
PKM	24	448	–	0.200704	1536	0.783135	1.541407	8/37.590, 12/56.386, 16/75.181, 24/112.771, 32/150.362, 48/225.543, 64/300.724
PKM	24	704	–	0.495616	1536	0.785101	3.806331	8/37.685, 12/56.527, 16/75.370, 24/113.055, 32/150.739, 48/226.109, 64/301.479
PKM	24	1024	–	1.048576	1536	0.787559	8.053064	8/37.803, 12/56.704, 16/75.606, 24/113.408, 32/151.211, 48/226.817, 64/302.423
PKM	24	1536	–	2.359296	1536	0.791491	18.119393	8/37.992, 12/56.987, 16/75.983, 24/113.975, 32/151.966, 48/227.949, 64/303.932
MoE	12	–	2	–	–	0.106207	0.084935	2/1.274, 3/1.912, 4/2.549, 6/3.823, 8/5.098, 12/7.647, 16/10.196, 24/15.294, 32/20.392, 48/30.588, 64/40.784
MoE	12	–	4	–	–	0.106226	0.254804	2/1.275, 3/1.912, 4/2.549, 6/3.824, 8/5.099, 12/7.648, 16/10.198, 24/15.297, 32/20.395, 48/30.593, 64/40.791, 96/61.186
MoE	12	–	8	–	–	0.106263	0.594543	2/1.275, 3/1.913, 4/2.550, 6/3.825, 8/5.101, 12/7.651, 16/10.201, 24/15.302, 32/20.402, 48/30.604, 64/40.805, 96/61.207
MoE	12	–	16	–	–	0.106337	1.274020	2/1.276, 3/1.914, 4/2.552, 6/3.828, 8/5.104, 12/7.656, 16/10.208, 24/15.312, 32/20.417, 48/30.625, 64/40.833, 96/61.250
MoE	16	–	2	–	–	0.251727	0.201327	2/3.021, 3/4.531, 4/6.041, 6/9.062, 8/12.083, 12/18.124, 16/24.166, 24/36.249, 32/48.332
MoE	16	–	4	–	–	0.251760	0.603980	2/3.021, 3/4.532, 4/6.042, 6/9.063, 8/12.084, 12/18.127, 16/24.169, 24/36.253, 32/48.338, 48/72.507, 64/96.676
MoE	16	–	8	–	–	0.251825	1.409286	2/3.022, 3/4.533, 4/6.044, 6/9.066, 8/12.088, 12/18.131, 16/24.175, 24/36.263, 32/48.350, 48/72.526, 64/96.701
MoE	16	–	16	–	–	0.251956	3.019899	16/24.188, 24/36.282, 32/48.376, 48/72.563, 64/96.751
MoE	20	–	2	–	–	0.491626	0.393216	4/11.799, 6/17.699, 8/23.598, 12/35.397, 16/47.196, 24/70.794, 32/94.392, 48/141.588
MoE	20	–	4	–	–	0.491677	1.179648	4/11.800, 6/17.700, 8/23.601, 12/35.401, 16/47.201, 24/70.802, 32/94.402, 48/141.603
MoE	20	–	8	–	–	0.491780	2.752512	4/11.803, 6/17.704, 8/23.605, 12/35.408, 16/47.211, 24/70.816, 32/94.422, 48/141.633
MoE	20	–	16	–	–	0.491985	5.898240	16/47.231, 24/70.846, 32/94.461, 48/141.692, 64/188.922
MoE	24	–	2	–	–	0.849499	0.679477	8/40.776
MoE	24	–	4	–	–	0.849572	2.038432	8/40.779, 12/61.169, 16/81.559, 24/122.338, 32/163.118

Continued on next page

Table 2 (continued)

Arch.	H	b	E	memc	memv	N	M	Measured D/C (B/EF)
MoE	24	–	8	–	–	0.849720	4.756341	8/40.787, 12/61.180, 16/81.573, 24/122.360, 32/163.146, 48/244.719
Dense	12	–	–	–	–	0.106189	0	1/0.637, 2/1.274, 3/1.911, 4/2.549, 6/3.823, 8/5.097, 12/7.646, 16/10.194, 24/15.291, 32/20.388, 48/30.582, 64/40.777, 96/61.165
Dense	16	–	–	–	–	0.251694	0	1/1.510, 2/3.020, 3/4.530, 4/6.041, 6/9.061, 8/12.081, 12/18.122, 16/24.163, 24/36.244, 32/48.325, 48/72.488, 64/96.651, 96/144.976
Dense	20	–	–	–	–	0.491575	0	2/5.899, 3/8.848, 4/11.798, 6/17.697, 8/23.596, 12/35.393, 16/47.191, 24/70.787, 32/94.382, 48/141.574, 64/188.765, 96/283.147
Dense	24	–	–	–	–	0.849425	0	3/15.290, 4/20.386, 6/30.579, 8/40.772, 12/61.159, 16/81.545, 24/122.317, 32/163.090, 48/244.634, 64/326.179
Dense	32	–	–	–	–	2.013403	0	4/48.322, 6/72.483, 8/96.643, 12/144.965, 16/193.287, 24/289.930, 32/386.573, 48/579.860, 64/773.147
STEM	12	–	–	0.050432	3072	0.094408	–	1/0.566, 2/1.133, 3/1.699, 4/2.266, 6/3.399, 8/4.532, 12/6.797, 16/9.063, 24/13.595, 32/18.126, 48/27.189, 64/36.253, 96/54.379, 128/72.505
STEM	16	–	–	0.050432	4096	0.222363	–	1/1.334, 2/2.668, 3/4.003, 4/5.337, 6/8.005, 8/10.673, 12/16.010, 16/21.347, 24/32.020, 32/42.694, 48/64.040, 64/85.387, 96/128.081
STEM	20	–	–	0.050432	5120	0.432639	–	2/5.192, 3/7.787, 4/10.383, 6/15.575, 8/20.767, 12/31.150, 16/41.533, 24/62.300, 32/83.067, 48/124.600, 64/166.133
STEM	24	–	–	0.050432	6144	0.745683	–	3/13.422, 4/17.896, 6/26.845, 8/35.793, 12/53.689, 16/71.586, 24/107.378, 32/143.171, 48/214.757, 64/286.342

B FITTING SCALING LAWS FUNCTIONS

B.1 OPTIMIZATION OF FORMULA COEFFICIENTS

For all four architectures, coefficients are fitted independently by nonlinear least squares. Throughout the fitting procedure, N, D, T, and M are fixed-unit raw counts, and $\log$ denotes the natural logarithm. For observation i, with predictors $\mathbf{x}_i = (N_i, D_i)$ or (N_i, D_i, M_i), the optimized residual is

$$r_i(\boldsymbol{\theta}) = \sqrt{w_i}\left[\log \widehat{L}(\mathbf{x}_i; \boldsymbol{\theta}) - \log L_i\right].$$

The weight w_i is proportional to the inverse number of observations in the corresponding architecture lineage and is normalized to have mean one. Thus, every lineage contributes equal total weight even when token-budget coverage differs across lineages. Full-data fits use the soft-L_1 loss with $f_{\text{scale}} = 10^{-3}$; validation fits use the Huber loss with the same scale. Optimization is performed in log-loss space.

The scale variables N, D, and M are observed inputs and are not optimized. For the PKM and MoE three-variable fits, the positive amplitudes are represented as $A = \exp(a)$ and $B = \exp(b)$, where a and b are unconstrained optimizer coordinates. The positive exponents are represented as $\alpha = \text{softplus}(a_\alpha) + \epsilon$, $\beta = \text{softplus}(a_\beta) + \epsilon$, and $\kappa = \text{softplus}(a_\kappa) + \epsilon$, with $\text{softplus}(x) = \log(1 + \exp x)$ and $\epsilon = 10^{-8}$. The interaction coefficients λ, μ, ζ, and ξ are optimized directly and may have either sign. The Dense Transformer and STEM fits instead use exponential transforms for A, B, α, and β, while retaining signed λ and ξ.

The Dense Transformer and STEM fits use the same robust least-squares objective, with the auxiliary coordinates

$$a = \log A, \qquad b = \log B, \qquad a_\alpha = \log \alpha, \qquad a_\beta = \log \beta,$$

with bounds

$$a, b \in [-60, 100], \qquad a_\alpha, a_\beta \in [-6, 3],$$

$$\lambda \in [-5, 5], \qquad \xi \in [-0.5, 0.5].$$

B.1.1 Capacity–Token Coupling and Local Responses

The parameterization introduces coupling between the amplitude and exponent coefficients. In particular, the data-dependent term can be written equivalently as

$$\frac{BN^{-\lambda}M^{-\mu}}{D^{\beta+\zeta\log M+\xi\log N}} = BD^{-\beta}N^{-(\lambda+\xi\log D)}M^{-(\mu+\zeta\log D)}.$$

Therefore, at a given token budget, the effective exponent of N is $\lambda + \xi\log D$, while the effective exponent of M is $\mu + \zeta\log D$. Over a finite range of D, an increase in λ can be partly offset by a decrease in ξ, and the same compensation can occur between μ and ζ, without substantially changing the predictions. Consequently, individual fitted coefficients should not be interpreted as independent causal effects. We instead evaluate the combined amplitude, $q_D(N, M)$, effective exponents, local scaling responses, and held-out predictions. The fitted full-data coefficients of the reported laws are shown in Table 3. PKM and MoE use the selected P6F law (Equation 2; Appendix B.3); the joint PKM–MoE fit uses its P6F working reference; and the Dense Transformer and STEM use vP6F (Equation 3). The table includes every learned amplitude, exponent, and interaction parameter. A dash indicates that the parameter is absent from the two-variable law.

Table 3. *Full-data fitted coefficients of the reported scaling laws.*

Architecture	Formula	A	B	α	β	κ	λ	μ	ζ	ξ
PKM	P6F	1.96e1	2.02e4	9.29e−2	6.36e−1	1.60e−2	5.41e−1	−2.63e−1	1.07e−2	−3.19e−2
MoE	P6F	1.42e1	7.30e3	7.47e−2	5.51e−1	1.73e−2	2.94e−1	−9.86e−2	2.34e−3	−1.77e−2
Joint PKM and MoE	P6F	1.79e1	6.12e4	9.29e−2	6.72e−1	1.20e−2	5.80e−1	−2.50e−1	1.08e−2	−3.36e−2
Dense Transformer	vP6F	1.32e1	6.35e2	8.49e−2	4.71e−1	–	1.11e−2	–	–	−8.00e−3
STEM	vP6F	1.16e1	1.08e3	8.03e−2	4.80e−1	–	5.85e−2	–	–	−9.46e−3

All numerical ranges below evaluate the single PKM coefficient vector in Table 3 over the observed configurations; they are not uncertainty intervals.

The finite-model component is

$$U(N, M) = AN^{-\alpha}M^{-\kappa}.$$

Because $\alpha > 0$ and $\kappa > 0$, U decreases monotonically with either N or M when the other coordinate is fixed. Across the 20 observed (N, M) configurations, U ranges from 1.994 to 2.580. Its minimum occurs at $N = 791.5$ million and $M = 18.119$ billion, whereas its maximum occurs at $N = 100.0$ million and $M = 0.308$ billion. Doubling either capacity coordinate gives

$$\frac{U(2N, M)}{U(N, M)} = 2^{-\alpha} = 0.9376, \qquad \frac{U(N, 2M)}{U(N, M)} = 2^{-\kappa} = 0.9889.$$

Thus, within this fitted component, doubling N reduces U by 6.24%, while doubling M reduces it by 1.11%. These percentages concern U alone, not total predicted loss.

For the data-dependent component, define

$$V_0(N, M) = BN^{-\lambda}M^{-\mu}, \qquad V(N, D, M) = V_0(N, M)D^{-q_D(N,M)}.$$

Across the same 20 configurations, $V_0(N, M)$ ranges from 82.1 to 441.4, a 5.4-fold variation in the prefactor of V, while $q_D(N, M)$ ranges from 0.211 at $N = 783.1$ million and $M = 1.541$ billion to 0.299 at $N = 104.0$ million and $M = 14.496$ billion. Increasing N lowers both V_0 and q_D, whereas increasing M raises both. Their effects on V oppose each other: V_0 is the multiplicative prefactor of D^{-q_D}, while a larger q_D makes D^{-q_D} smaller. At fixed D, the net dependence of V on N and M is therefore governed by the effective exponents $\lambda + \xi\log D$ and $\mu + \zeta\log D$, respectively.

At fixed supported N and M, doubling the training budget gives

$$\frac{V(N, 2D, M)}{V(N, D, M)} = 2^{-q_D(N,M)}.$$

The proportional reduction is therefore $R_D(N, M) = 1 - 2^{-q_D(N,M)}$, with $\partial R_D/\partial q_D = (\log 2)2^{-q_D} > 0$. Evaluating the endpoints of the fitted q_D range gives reductions of 13.6% and 18.7%.

To measure how doubling N or M changes this percentage benefit of doubling D, we compare the corresponding reductions R_D. Doubling either capacity coordinate changes q_D according to

$$q_D(2N, M) = q_D(N, M) + \xi\log 2, \qquad q_D(N, 2M) = q_D(N, M) + \zeta\log 2.$$

The relative changes in the data-doubling benefit are therefore

$$\frac{R_D(2N, M)}{R_D(N, M)} = \frac{1 - 2^{-[q_D(N,M)+\xi \log 2]}}{1 - 2^{-q_D(N,M)}},$$

$$\frac{R_D(N, 2M)}{R_D(N, M)} = \frac{1 - 2^{-[q_D(N,M)+\zeta \log 2]}}{1 - 2^{-q_D(N,M)}}.$$

With the fitted coefficients, the first ratio ranges from 0.902 to 0.933: doubling N makes the reduction in V obtained from doubling D approximately 6.7–9.8% smaller. The second ratio ranges from 1.022 to 1.033: doubling M makes that reduction approximately 2.2–3.3% larger. Each is a range because the same fixed change in q_D is applied to different initial $q_D(N, M)$ values across the observed configurations. These ratios compare percentage reductions in V and do not describe changes in total predicted loss. Their directions are consistent across all 47 validation fits: every fitted ξ is negative and every fitted ζ is positive.

Finally, the response of total predicted loss $\widehat{L} = U + V$ depends on the relative sizes and responses of both components. Its local log-log responses are

$$\mathcal{E}_N = -\frac{\partial \log \widehat{L}}{\partial \log N} = \frac{\alpha U + (\lambda + \xi \log D)V}{U + V},$$

$$\mathcal{E}_M = -\frac{\partial \log \widehat{L}}{\partial \log M} = \frac{\kappa U + (\mu + \zeta \log D)V}{U + V}, \qquad \mathcal{E}_D = -\frac{\partial \log \widehat{L}}{\partial \log D} = \frac{q_D V}{U + V}.$$

In the fitted decomposition, U constitutes 73.7–91.7% of predicted loss over the observed data. Evaluating the single full-data fit of the selected PKM scaling law at all 192 observed (N, D, M) endpoints, which span $N = 100.0$–791.5 million active non-embedding parameters, $M = 0.308$–21.475 billion inactive parameters, and $D = 2$–96 billion training tokens, total predicted loss decreases locally as any of N, M, or D increases. These are the marginal ranges of the observed coordinates; the 192 endpoints do not form a complete Cartesian grid over them. The following ranges are the minimum and maximum responses across those endpoints, not uncertainty intervals or ranges across validation fits. Holding the other coordinates fixed, a small 1% increase reduces predicted loss by approximately 0.020–0.063% for N, 0.003–0.015% for M, and 0.022–0.069% for D. Thus, the fitted loss is less locally sensitive to additional memory capacity than to active parameters or training tokens. These quantities are local sensitivities of the fitted function, not finite-doubling effects or causal estimates.

B.2 Evaluation of Fitted Formulas

To assess both goodness of fit and predictive generalization, we evaluate each fitted scaling law on the complete dataset and under a suite of held-out validation tests. Each endpoint is one measured final cross-entropy value L_i, associated with scale coordinates (N_i, D_i) or (N_i, D_i, M_i).

For held-out evaluation, each validation fold k partitions the observed endpoints into a training set $\mathcal{T}_k$ and a disjoint test set $\mathcal{S}_k$. The formula coefficients are refitted using only $\mathcal{T}_k$, after which prediction error is evaluated only on $\mathcal{S}_k$. For any evaluation set $\mathcal{S}$, let $e_i = \widehat{L}_i - L_i$ and let $\widetilde{w}_i(\mathcal{S})$ denote the normalized reporting weight of endpoint i, such that $\sum_{i \in \mathcal{S}} \widetilde{w}_i(\mathcal{S}) = 1$. We report

$$\mathrm{MAE}(\mathcal{S}) = \sum_{i \in \mathcal{S}} \widetilde{w}_i(\mathcal{S})|e_i|, \qquad \mathrm{RMSE}(\mathcal{S}) = \sqrt{\sum_{i \in \mathcal{S}} \widetilde{w}_i(\mathcal{S}) e_i^2}.$$

Lower values are better. MAE measures the typical absolute prediction error, whereas RMSE gives greater weight to large errors. Both are evaluated in the original cross-entropy scale. For PKM and MoE, let $\mathcal{G}(\mathcal{S})$ be the set of model lineages represented in $\mathcal{S}$, let $g(i)$ be the lineage of endpoint i, and let $n_g(\mathcal{S}) = |\{i \in \mathcal{S} : g(i) = g\}|$. The reporting weights are

$$\widetilde{w}_i(\mathcal{S}) = \frac{1}{|\mathcal{G}(\mathcal{S})|\, n_{g(i)}(\mathcal{S})}.$$

Each represented sparse-model lineage therefore contributes equal total mass within the evaluation set, irrespective of how many token-budget endpoints it contains. For the Dense Transformer and

STEM tables, $\widetilde{w}_i(\mathcal{S}) = 1/|\mathcal{S}|$, so the reported metrics reduce to ordinary endpoint-wise MAE and RMSE.

For a validation family containing K folds, we report the arithmetic means of the fold-level errors,

$$\mathrm{MAE}_{\mathrm{family}} = \frac{1}{K}\sum_{k=1}^{K}\mathrm{MAE}(\mathcal{S}_k), \qquad \mathrm{RMSE}_{\mathrm{family}} = \frac{1}{K}\sum_{k=1}^{K}\mathrm{RMSE}(\mathcal{S}_k),$$

only when every declared fold yields a valid fit. Failed folds are reported separately rather than omitted from the average.

Fit validity is determined by fixed numerical and structural checks, not by the magnitude of MAE or RMSE. Every invalid P3-or-later result reported in the PKM and MoE tables is explained by one or both of two checks: (i) the optimizer's first-order optimality measure for the retained best-objective fit is not finite or exceeds 10^{-3}; and (ii) increasing either N or D causes predicted loss to increase beyond numerical tolerance somewhere within the observed coordinate range.

For every table in this section, a validation fold contributes to a family metric only when its refit passes the validity checks above, its test-set predictions are finite, and both its MAE and RMSE are finite. Consequently, one failed fold suppresses the aggregate for that family. A large prediction error does not by itself make a fit invalid. Likewise, a large condition number, strong parameter correlation, bootstrap instability, or failure to improve a parent formula's held-out error is diagnostic evidence but does not alone suppress a table entry.

For the validation definitions below, capacity refers to the number of inactive parameters M. A capacity family groups corresponding sparse-capacity configurations across active-model sizes, although their exact values of M may differ. A model lineage contains all token-budget endpoints associated with one fixed observed (N, M) configuration. We report the following full-data and held-out evaluations:

Full. "Full" denotes the in-sample result obtained by fitting and evaluating on the complete dataset. Every other reported evaluation is a genuinely held-out result from a refitted validation fold or the mean across all folds in that family.

Leave lowest losses out (LLLO$_p$). For a held-out percentage p, all n endpoints are ordered from lowest to highest observed cross-entropy. If two endpoints have exactly the same loss, a fixed endpoint identifier is used only to make their ordering reproducible. The lowest-loss $\lceil (p/100)n \rceil$ endpoints form the test set, and the complementary higher-loss endpoints form the training set; each partition is therefore disjoint and exhaustive. We evaluate $p \in \{10, 20, 30, 40, 50\}$ and report the five LLLO$_p$ tests separately. Increasing p measures progressively longer extrapolation from worse observed losses toward the best observed endpoints. Because LLLO$_p$ is an endpoint-level split, its training and test sets may contain endpoints from the same model lineage.

Leave one model out (LOMO). Each fold removes one complete lineage, refits on all other lineages, and predicts every endpoint of the excluded model configuration.

Leave one size out (LOSO). Each fold removes one complete active-model- size family, represented by its measured non-embedding active-parameter counts N, refits on the remaining size families, and predicts the excluded family.

Leave one capacity out (LOCO). Each fold removes one complete capacity family for inactive parameters across all available active-model sizes and token budgets. A capacity family can correspond to different absolute values of M at different active-model sizes; all reported capacities use the measured values of M.

Capacity-forward validation (CF). The observed capacity families for inactive parameters are ordered from smallest to largest. For each target after the first two levels, the formula is fitted using only lower-capacity families and evaluated at the target family. This tests forward extrapolation in M.

Future-token validation (FT). We first identify token budgets observed in at least 80% of the model lineages and select the two largest eligible budgets. The formula is fitted on all rows below the smaller selected budget and evaluated jointly on the two selected tail budgets.

Unseen-capacity future-token validation (UCF). For each observed capacity family for inactive parameters, the training set excludes that family and contains only rows below the smaller of the same two selected tail budgets. The test set contains the excluded capacity family for inactive parameters at those two budgets. UCF therefore tests simultaneous extrapolation in M and training tokens D.

Largest-corner validation (LC). The test configuration uses the largest observed active-model-size family and the largest capacity in inactive parameters available at that size having observations at both selected tail budgets. Training is restricted to rows belonging to lower active-model-size families, lower capacity families for inactive parameters, and lower token budgets. This evaluates extrapolation to the largest jointly supported (N, M, D) corner.

Formula selection considers these held-out MAE and RMSE results together with the full-data fit; it is not based on the full-data error alone.

B.3 Fitted Formulas

We evaluated the following candidate functional forms. Reused coefficient symbols denote analogous roles across candidates, not shared numerical values: every formula is refitted independently. The labels are experimental names used to track the progression of fitted hypotheses.

Three-variable laws $L(N, D, M)$. For compactness, define

$$U_M(N, M) = \frac{AM^{-\kappa}}{N^{\alpha}}, \qquad q_M(M) = \beta + \zeta \log M,$$

$$q_{NM}(N, M) = \beta + \zeta \log M + \xi \log N, \qquad q_N(M) = \alpha + \gamma \log M.$$

The fitted candidates are

$$\begin{aligned}
\text{P0:} \quad & \widehat{L}_0(N, D, M) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}, \\
\text{P1A:} \quad & \widehat{L}_{1A}(N, D, M) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} + \frac{G}{M^{\eta}}, \\
\text{P1B:} \quad & \widehat{L}_{1B}(N, D, M) = E + U_M(N, M) + \frac{B}{D^{\beta}}, \\
\text{P1C:} \quad & \widehat{L}_{1C}(N, D, M) = E + \frac{A}{N^{\alpha}} + \frac{BM^{-\omega}}{D^{\beta}}, \\
\text{P2B:} \quad & \widehat{L}_{2B}(N, D, M) = E + U_M(N, M) + \frac{BM^{-\omega}}{D^{\beta}}, \\
\text{P3A:} \quad & \widehat{L}_{3A}(N, D, M) = E + \frac{AM^{-\kappa}}{N^{q_N(M)}} + \frac{B}{D^{\beta}}, \\
\text{P3B:} \quad & \widehat{L}_{3B}(N, D, M) = E + U_M(N, M) + \frac{B}{D^{q_M(M)}}, \\
\text{P3C:} \quad & \widehat{L}_{3C}(N, D, M) = E + \frac{AM^{-\kappa}}{N^{q_N(M)}} + \frac{B}{D^{q_M(M)}}, \\
\text{P4:} \quad & \widehat{L}_4(N, D, M) = E + \left[U_M(N, M) + \frac{B}{D^{q_M(M)}}\right]^{\rho}, \\
\text{P5:} \quad & \widehat{L}_5(N, D, M) = E + U_M(N, M) + \frac{BN^{-\lambda}}{D^{q_M(M)}}.
\end{aligned}$$

The P0 baseline is included in this group because it was fitted to the same three-coordinate dataset, although its prediction is independent of M.

The candidates with a joint model–capacity-dependent token exponent are

$$\text{P6B:}\quad \widehat{L}_{6B}(N,D,M) = E + U_M(N,M) + \frac{BN^{-\lambda}}{D^{q_{NM}(N,M)}},$$
$$\text{P6C:}\quad \widehat{L}_{6C}(N,D,M) = E + U_M(N,M) + \frac{BM^{-\mu}}{D^{q_{NM}(N,M)}},$$
$$\text{P6D:}\quad \widehat{L}_{6D}(N,D,M) = E + U_M(N,M) + \frac{BN^{-\lambda}M^{-\mu}}{D^{q_{NM}(N,M)}},$$
$$\text{P6F:}\quad \widehat{L}_{6F}(N,D,M) = U_M(N,M) + \frac{BN^{-\lambda}M^{-\mu}}{D^{q_{NM}(N,M)}},$$
$$\text{P6G:}\quad \widehat{L}_{6G}(N,D,M) = U_M(N,M) + \frac{B(N+M)^{-\mu}}{D^{q_{NM}(N,M)}},$$
$$\text{P6I:}\quad \widehat{L}_{6I}(N,D,M) = U_M(N,M) + \frac{B(NM)^{-\mu}}{D^{q_{NM}(N,M)}},$$
$$\text{P6J:}\quad \widehat{L}_{6J}(N,D,M) = U_M(N,M) + \frac{B\left(N/(N+M)\right)^{-\mu}}{D^{q_{NM}(N,M)}},$$
$$\text{P6K:}\quad \widehat{L}_{6K}(N,D,M) = U_M(N,M) + \frac{B\left(N/M\right)^{-\mu}}{D^{q_{NM}(N,M)}}.$$

Thus, P6F removes P6D's separately fitted constant floor, while P6G, P6I, P6J, and P6K replace its two-factor data-term amplitude with a single capacity coordinate.

Finally, define the positive learned functions

$$B_7(N) = \exp(b_0 + b_N \log N)\,, \qquad q_7(N,M) = \exp(a_0 + a_N \log N + a_M \log M)\,.$$

The Farseer-style three-variable candidate is

$$\text{P7:}\qquad \widehat{L}_7(N,D,M) = E + U_M(N,M) + \frac{B_7(N)}{D^{q_7(N,M)}}.$$

Two-variable laws $L(N,D)$. Three mathematical forms were evaluated for the Dense Transformer and STEM, with all coefficients fitted independently for each architecture. We retain the established Chinchilla name for the two-variable P0 baseline and use the prefix vP6F only to distinguish that form from the similarly named three-variable candidate. For the normalized coordinates

$$n = \frac{N}{N_{\text{ref}}}, \qquad d = \frac{D}{D_{\text{ref}}},$$

the Chinchilla baseline is

$$\text{Chinchilla:}\qquad \widehat{L}_{\text{C}}(N,D) = E + An^{-\alpha} + Bd^{-\beta}.$$

The memory-free joint-interaction law is evaluated on raw N and D:

$$q_D(N) = \beta + \xi \log N, \qquad \text{vP6F:}\qquad \widehat{L}_{\text{vP6F}}(N,D) = \frac{A}{N^{\alpha}} + \frac{BN^{-\lambda}}{D^{q_D(N)}}.$$

For Farseer, define

$$F_U(N) = \exp(a_3 n^{\gamma} + b_3)\,, \qquad B_{\text{F}}(N) = \exp\big(a_2 n^{\beta} + b_2\big)\,,$$
$$A_{\text{F}}(N) = \exp(a_1 n^{\alpha} + b_1)\,.$$

Its fitted form is

$$\text{Farseer:}\qquad \widehat{L}_{\text{F}}(N,D) = F_U(N) + B_{\text{F}}(N) D^{-A_{\text{F}}(N)}.$$

B.4 Fitted-Formula Evaluation Results

MAE and RMSE are reported in separate tables in the original cross-entropy scale using a common 10^{-3} exponent and a three-digit mantissa; lower values are better. The RMSE tables remain in this appendix, while the corresponding MAE tables are collected in Appendix D. The lowest displayed

value in each evaluation column is bolded, including all formulas tied at the displayed precision. Under the validity rules specified above, an invalid full fit or held-out family is shown by its parenthesized validity-condition number rather than an error value. A held-out family is suppressed when at least one declared fold is invalid; valid-subset fallbacks are not substituted. The datasets differ across architectures, so values should be compared between formulas within an architecture, not directly between architectures. When a table cell contains a parenthesized Roman numeral instead of an error value, the corresponding full fit or at least one declared fold in that family failed the identically numbered validity condition above, so the family mean is not reported. Multiple numerals indicate failures of more than one kind within that family.

PKM. The PKM fits use 192 endpoints. The complete validation suite contains 47 folds, including the five LLLO fractions. The corresponding full-data and LLLO RMSE trends are visualized in Figure 6, and the selected P6F law's LLLO_{30} endpoint predictions are shown in Figure 11.

Table 4. *PKM RMSE (all values in 10^{-3} cross-entropy units).*

Formula	Full-data and LLLO						Grouped and directional holdouts						
	Full	LLLO_{10}	LLLO_{20}	LLLO_{30}	LLLO_{40}	LLLO_{50}	LOMO	LOSO	LOCO	CF	FT	UCF	LC
P0	36.7	45.1	54.3	62.6	67.9	116	42.9	41.8	46.3	52.5	48.0	88.6	70.8
P1A	12.4	17.7	27.5	33.2	33.5	37.5	11.6	16.5	12.0	12.0	21.4	18.7	25.8
P1B	13.1	25.4	30.0	32.3	31.0	33.2	13.5	18.8	13.7	13.8	19.8	19.1	26.3
P1C	17.1	16.7	19.9	30.6	33.6	44.9	16.4	20.0	17.0	15.9	32.6	26.8	21.4
P2B	13.1	25.4	30.0	32.3	31.0	33.2	12.5	18.8	13.7	13.8	32.6	19.1	26.3
P3A	12.8	24.6	29.7	32.4	31.2	33.4	12.1	17.9	13.1	13.2	19.7	18.5	(i)
P3B	10.2	23.8	26.5	28.2	26.7	26.2	12.0	20.0	12.5	13.8	14.5	14.6	5.23
P3C	9.79	22.9	25.5	27.2	26.2	25.9	11.2	(i)	11.5	(i)	14.3	13.6	7.43
P4	7.52	11.4	11.5	11.9	11.1	11.2	8.91	13.6	10.1	11.2	12.1	15.2	5.69
P5	7.80	11.8	12.2	12.8	11.9	11.6	8.57	(i)	10.1	11.5	12.2	15.1	13.8
P6B	6.52	9.85	9.04	9.21	9.87	10.1	6.75	10.8	8.25	9.60	10.9	13.0	7.54
P6C	5.22	6.36	8.42	(i)	10.9	9.76	(i)	11.6	6.47	7.22	9.10	10.1	5.98
P6D	**4.58**	**4.32**	**5.01**	**5.86**	11.0	**9.27**	**4.82**	**8.11**	**6.08**	**6.66**	7.63	(i)	3.11
P6F	**4.58**	**4.32**	**5.01**	**5.86**	**7.39**	**9.27**	5.13	8.67	**6.08**	**6.66**	7.63	**8.49**	3.11
P6G	7.04	10.6	10.1	11.3	(ii)	12.0	(i, ii)	(ii)	(i, ii)	11.0	10.3	(i, ii)	**2.84**
P6I	7.68	(i, ii)	10.3	(i, ii)	24.5	12.1	(i, ii)	17.4	(i, ii)	(ii)	11.0	(i, ii)	4.07
P6J	(ii)	(ii)	(ii)	(ii)	(ii)	11.4	(ii)	(ii)	(ii)	(ii)	(ii)	(ii)	3.62
P6K	5.67	6.13	6.53	8.24	9.66	11.1	7.41	10.6	7.98	7.34	**6.79**	9.63	(i, ii)
P7	7.48	10.9	10.9	11.5	10.9	11.4	7.93	11.6	9.65	11.1	11.5	14.3	13.6

MoE. The MoE fits use 124 endpoints across 15 model lineages and 36 validation folds, including the five LLLO fractions. P6F gives valid full-data and LLLO fits, with LLLO RMSE rising steadily from $5.36\mathrm{e}{-3}$ to $9.79\mathrm{e}{-3}$ as the held-out fraction grows. These errors indicate a good fit, comparable to or lower than the LLLO errors reported in other scaling-law papers (Section 4.2). The LOMO, LOCO, CF, and UCF families contain invalid folds (Table 5); these holdouts remove entire lineages or capacity families, which is demanding for the smaller MoE dataset (124 endpoints and four capacity families, versus 192 and six for PKM). RMSE for the full-data fit and all five LLLO fractions is shown in Figure 7, and LLLO_{30} observed-versus-predicted losses in Figure 12.

Table 5. *MoE RMSE (all values in 10^{-3} cross-entropy units).*

Formula	Full-data and LLLO						Grouped and directional holdouts						
	Full	LLLO_{10}	LLLO_{20}	LLLO_{30}	LLLO_{40}	LLLO_{50}	LOMO	LOSO	LOCO	CF	FT	UCF	LC
P0	23.5	22.3	22.1	28.2	47.2	52.0	25.3	22.8	29.1	32.6	45.4	40.8	45.0
P1A	10.3	13.7	15.9	22.9	33.5	37.8	12.1	12.7	11.7	12.3	16.7	17.5	26.8
P1B	10.4	14.1	15.2	25.8	29.9	35.8	12.8	12.6	11.7	12.2	16.5	16.4	20.5
P1C	13.3	11.8	15.8	20.9	40.4	43.2	12.8	15.5	14.9	14.5	22.7	22.3	22.0
P2B	10.4	11.8	15.2	25.8	29.9	35.8	10.2	12.6	12.7	12.2	22.7	16.4	22.0
P3A	10.3	14.4	15.8	25.3	29.9	35.9	9.98	12.6	11.5	12.0	16.5	16.5	25.6
P3B	7.64	6.70	8.88	17.5	17.1	23.4	9.46	12.2	10.3	13.6	7.98	9.91	14.8
P3C	7.28	6.79	8.42	15.8	16.0	22.6	8.48	(i)	10.2	13.4	7.93	**9.62**	15.9
P4	6.09	5.99	6.48	9.98	9.50	10.7	8.30	8.85	**8.70**	11.8	8.39	12.2	13.7
P5	6.26	6.16	6.94	11.0	10.2	11.8	**6.90**	**7.15**	8.91	12.1	8.41	12.4	14.6
P6B	6.10	6.04	6.18	9.09	9.46	10.4	(i)	8.65	9.18	**11.5**	8.13	11.8	13.1
P6C	5.79	**5.29**	6.51	9.48	10.0	11.2	7.57	8.39	9.70	12.0	7.89	10.8	14.5
P6D	**5.78**	5.36	6.17	8.49	**9.11**	10.4	(i)	(i)	(i)	12.1	7.75	11.1	14.3
P6F	5.80	5.36	6.08	8.49	**9.11**	**9.79**	(i)	8.56	(i)	(i)	7.67	(i, ii)	14.3
P6G	6.27	6.52	6.16	8.71	9.39	10.1	7.43	(i, ii)	(ii)	(i, ii)	7.62	(ii)	(i, ii)
P6I	6.27	6.36	6.17	9.47	18.5	(i, ii)	(i, ii)	(i, ii)	(i, ii)	12.6	7.72	15.5	(i, ii)
P6J	6.25	6.68	6.22	8.76	9.39	10.1	7.35	9.27	(ii)	18.4	7.60	(ii)	**2.61**
P6K	6.01	6.10	**5.74**	**8.48**	9.42	10.1	8.54	8.59	9.78	12.3	**7.37**	10.6	8.92
P7	6.22	6.31	6.45	9.96	9.78	10.9	(i)	8.22	8.91	11.7	8.32	11.8	14.9

Dense Transformer. The dense-Transformer fits use 57 endpoints and 12 validation folds. Because each two-variable model-size setting defines one lineage, LOMO is also the available held-out-size

test. A separate LOSO family and the capacity-based LOCO, CF, and UCF tests are not defined for this dataset; LC denotes the largest (N, D) corner.

Table 6. *Dense Transformer RMSE (all values in 10^{-3} cross-entropy units).*

Formula	Full	LLLO_{10}	LLLO_{20}	LLLO_{30}	LLLO_{40}	LLLO_{50}	LOMO	FT	LC
Chinchilla	18.8	42.2	49.4	51.7	61.4	61.1	25.6	25.8	27.3
vP6F	4.65	**6.49**	**6.24**	**6.08**	**9.30**	**10.6**	**11.4**	**6.17**	2.30
Farseer	**4.58**	6.98	7.06	6.22	9.72	38.7	47.5	7.13	**1.69**

STEM. The STEM fits use 48 endpoints and 11 validation folds. As for the Dense Transformer, LOMO is the available held-out-size test, capacity-based tests are not defined, and LC denotes the largest (N, D) corner.

Table 7. *STEM RMSE (all values in 10^{-3} cross-entropy units).*

Formula	Full	LLLO_{10}	LLLO_{20}	LLLO_{30}	LLLO_{40}	LLLO_{50}	LOMO	FT	LC
Chinchilla	13.2	26.2	29.3	34.2	35.1	36.4	16.8	17.1	25.9
vP6F	1.76	**1.27**	**2.82**	**2.88**	4.45	**5.18**	**4.32**	**2.31**	**1.15**
Farseer	**1.61**	5.12	4.44	7.86	**3.65**	16.2	15.0	2.73	4.27

Selection of the two-variable law. Farseer fits nine coefficients, three more than vP6F. It achieves lower full-data error for both architectures: for the Dense Transformer, MAE/RMSE decrease from $3.32/4.65$ to $3.25/4.58$; for STEM, they decrease from $1.43/1.76$ to $1.15/1.61$, with all values in 10^{-3} units. However, vP6F has lower held-out error in 27 of 32 architecture–metric–family comparisons reported in the Dense Transformer and STEM RMSE tables above and their MAE counterparts in Appendix D. The LLLO trends are shown in Figures 9 and 10.

Farseer's lower full-data error but poorer held-out extrapolation, together with its three additional fitted coefficients, is consistent with overfitting the observed endpoints, although it does not uniquely prove overfitting. We therefore retain the more parsimonious and better-generalizing vP6F. Its structure also matches the selected three-variable P6F law after removing the M-dependent factors, enabling direct comparison between the two- and three-variable fits.

Joint Sparse LLM Scaling-Law Formula. We additionally fit the three-variable candidate laws jointly to the PKM and MoE observations. The joint dataset contains 316 endpoints—192 PKM and 124 MoE—across 35 architecture-qualified model lineages. Its validation suite contains 74 folds, including the five LLLO fractions. The reported values include the full-data fits and every validation fold but no bootstrap uncertainty estimates. P6F remains the working reference for this joint fit; 72 of its 74 validation folds pass the applicable validity checks. The corresponding full-data and LLLO RMSE trends are visualized in Figure 8, and the joint PKM–MoE P6F law's LLLO_{30} endpoint predictions are shown in Figure 13.

Table 8. *Joint PKM–MoE RMSE (all values in 10^{-3} cross-entropy units).*

Formula	Full-data and LLLO						Grouped and directional holdouts						
	Full	LLLO_{10}	LLLO_{20}	LLLO_{30}	LLLO_{40}	LLLO_{50}	LOMO	LOSO	LOCO	CF	FT	UCF	LC
P0	37.6	48.8	46.8	52.7	63.6	78.4	34.0	37.1	40.2	46.5	46.8	49.7	55.7
P1A	13.0	22.7	23.0	26.7	45.1	33.5	12.6	15.2	13.8	14.1	19.5	20.2	42.2
P1B	13.8	50.0	27.6	28.5	32.1	34.1	13.2	18.9	16.6	14.9	21.0	19.0	49.3
P1C	17.1	26.3	26.9	33.7	37.7	44.5	17.1	18.4	19.1	18.6	31.7	29.9	48.0
P2B	13.8	26.3	26.9	28.5	32.1	34.1	13.1	18.8	16.3	14.9	31.7	19.0	49.3
P3A	13.6	27.4	(i)	28.4	32.4	(i)	(i)	(i)	(i)	14.6	20.8	19.3	49.6
P3B	12.4	24.9	26.8	26.9	30.5	32.5	12.8	20.1	13.5	15.0	19.2	16.6	37.5
P3C	12.2	25.9	25.8	26.2	30.1	32.4	(i)	(i)	13.3	14.9	19.1	17.0	38.9
P4	10.2	20.8	16.7	14.4	15.1	15.2	10.9	16.6	12.7	13.5	16.1	17.3	37.0
P5	10.4	19.2	17.3	14.9	15.5	15.3	10.9	13.6	12.7	13.7	16.4	16.6	37.2
P6B	9.65	22.8	12.2	12.6	13.6	13.9	9.85	11.8	(i)	12.6	14.0	15.7	35.0
P6C	9.17	11.8	12.0	11.4	13.1	13.3	(i)	(i)	(i)	(i)	11.6	(i)	33.6
P6D	**8.74**	**8.28**	**8.16**	(i)	13.6	**12.6**	(i)	**10.6**	**9.48**	**10.9**	**10.8**	(i)	37.7
P6F	**8.74**	**8.28**	**8.16**	**9.76**	**11.9**	**12.6**	**8.42**	**10.6**	**9.48**	(i)	**10.8**	12.7	(i)
P6G	9.71	13.9	12.0	13.1	14.8	(i, ii)	10.5	15.2	13.7	(i, ii)	13.3	15.8	**8.16**
P6I	10.1	14.3	12.5	15.2	(i, ii)	29.2	11.8	17.4	(i, ii)	16.1	13.4	(i)	24.6
P6J	9.83	14.0	14.1	20.0	15.3	14.8	10.9	16.8	(ii)	(ii)	13.3	(ii)	21.1
P6K	9.38	9.62	10.5	13.9	15.0	15.2	9.89	12.6	10.5	11.4	11.2	**12.4**	14.7
P7	10.2	18.2	15.8	13.8	14.8	14.8	10.6	13.0	12.5	13.4	15.5	16.3	36.5

Matched MoE LLLO holdout. The LLLO columns in Table 8 rank endpoints within the pooled PKM–MoE dataset, whereas the MoE LLLO columns in Table 5 rank endpoints within the MoE

dataset alone. To isolate the effect of sharing coefficients, we therefore construct a matched evaluation from the exact MoE LLLO test sets. For each fraction, the separately fitted MoE law is trained on the retained MoE endpoints, while the joint law is refitted on those same retained MoE endpoints plus all 192 PKM endpoints. Both laws are then evaluated on the identical held-out MoE endpoints; Table 9 reports the results. All five refits converge; the held-out endpoints match the MoE folds exactly, every held-out endpoint is absent from joint training, and every PKM endpoint remains in joint training.

Table 9. *Matched MoE LLLO evaluation of the separately fitted MoE law and joint PKM–MoE law. Errors are reported in* $\mathrm{e}{-3}$ *cross-entropy units; the ratio is joint RMSE divided by MoE RMSE.*

Split	MoE train	MoE test	MoE MAE	Joint MAE	MoE RMSE	Joint RMSE	RMSE ratio
LLLO_{10}	111	13	4.28	9.07	5.36	11.01	2.05
LLLO_{20}	99	25	4.50	7.61	6.08	10.01	1.65
LLLO_{30}	86	38	5.99	9.29	8.49	12.27	1.44
LLLO_{40}	74	50	6.53	10.23	9.11	13.01	1.43
LLLO_{50}	62	62	6.83	10.61	9.79	13.19	1.35

The separately fitted MoE law has lower MAE and RMSE in every matched evaluation. Therefore, for these observed architectures and extrapolation directions, adding PKM data while forcing one shared coefficient vector consistently degrades prediction of the held-out MoE endpoints. This matched comparison directly supports architecture-specific coefficients for the allocation analysis and provides evidence that PKM and MoE follow different scaling behavior over the measured regime.

The diagnostics below are grouped by evaluation type. Each group contains the PKM, MoE, joint PKM–MoE, Dense Transformer, and STEM charts at their original full-page width.

B.4.1 Full-data and LLLO RMSE

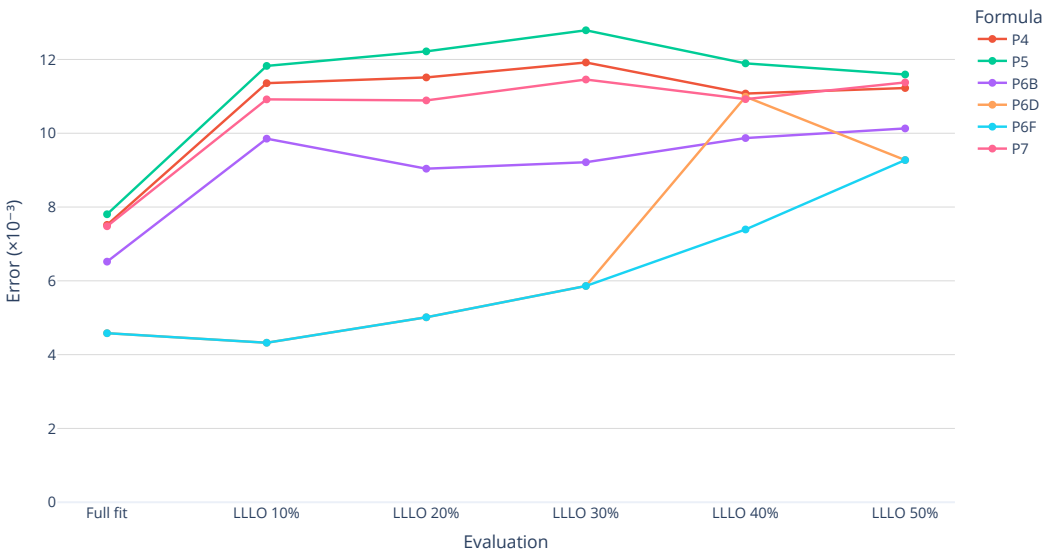


Figure 6. *PKM full-data and LLLO evaluation.* RMSE is shown in 10^{-3} cross-entropy units; lower values are better. Lines connect the separately refitted evaluation settings only as visual guides.

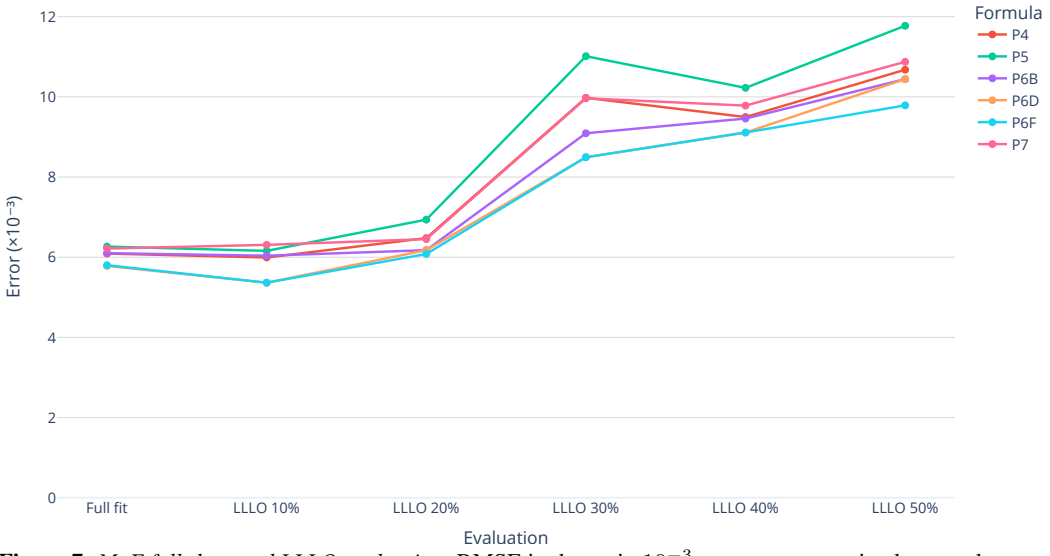


Figure 7. *MoE full-data and LLLO evaluation.* RMSE is shown in 10^{-3} cross-entropy units; lower values are better. Lines connect the separately refitted evaluation settings only as visual guides. A missing point denotes an invalid family metric.

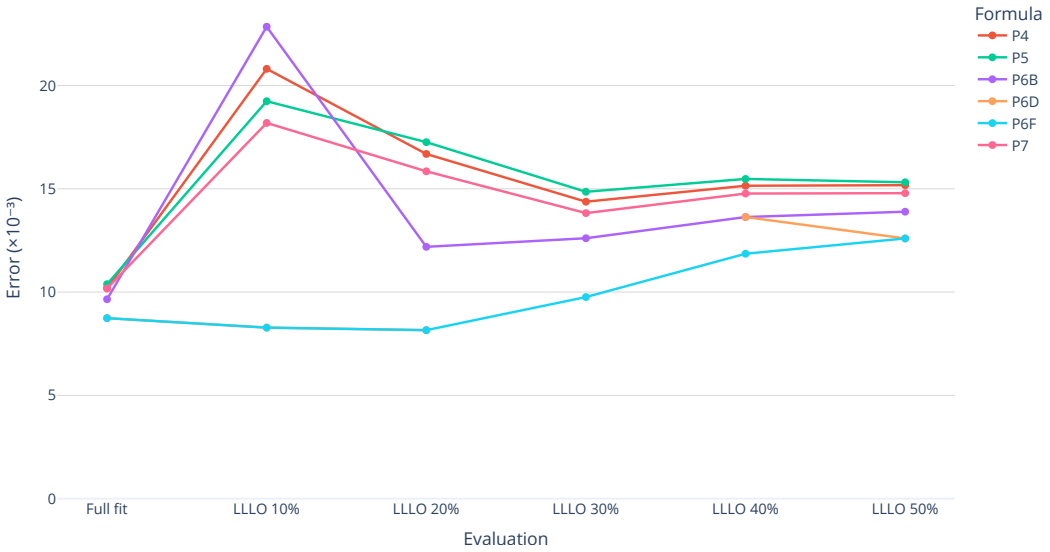


Figure 8. *Joint PKM–MoE full-data and LLLO evaluation.* The joint three-variable laws are evaluated on the pooled PKM and MoE endpoints. RMSE is shown in 10^{-3} cross-entropy units; lower values are better, and missing points denote invalid family metrics.

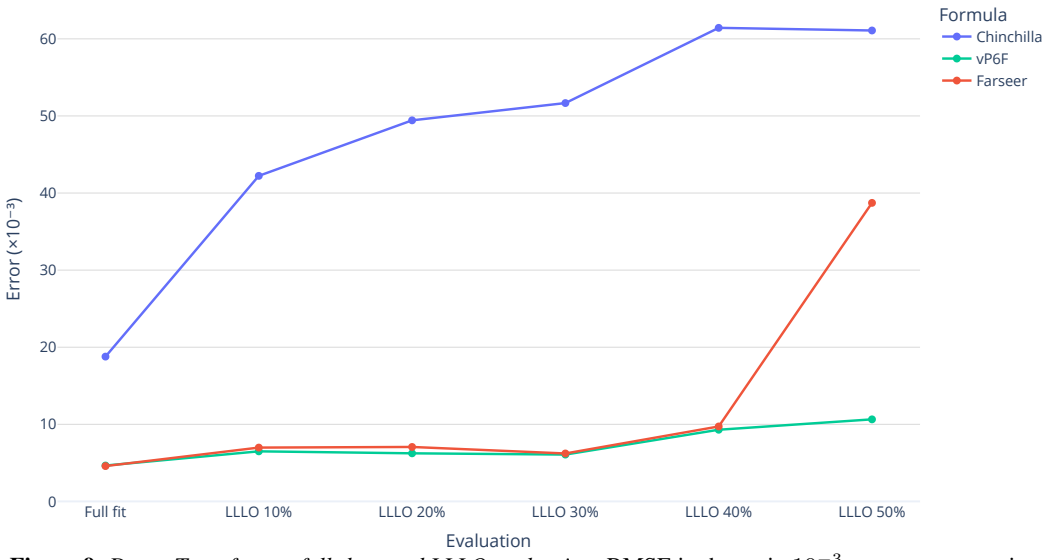


Figure 9. *Dense Transformer full-data and LLLO evaluation.* RMSE is shown in 10^{-3} cross-entropy units; lower values are better. Lines connect the separately refitted evaluation settings only as visual guides.

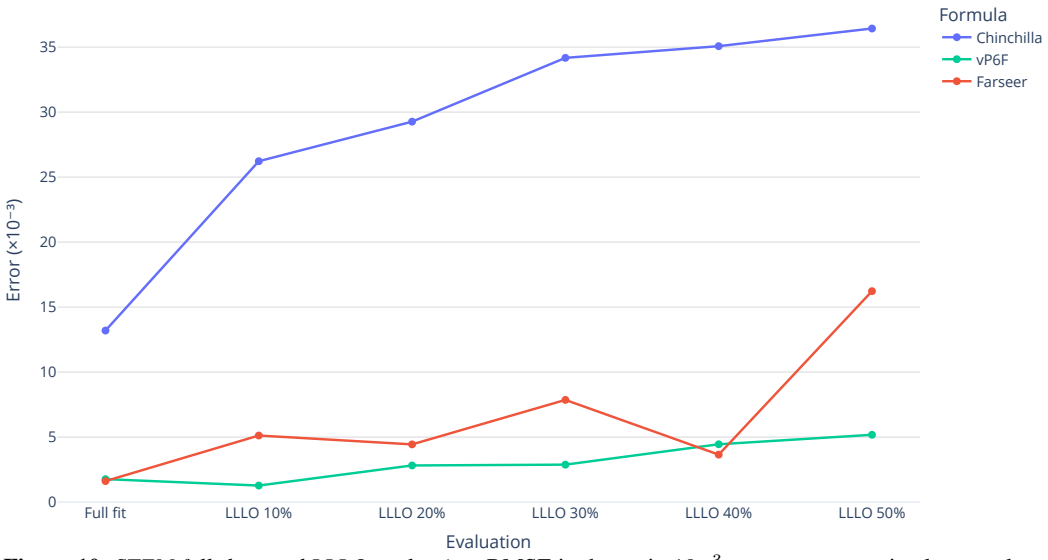


Figure 10. *STEM full-data and LLLO evaluation.* RMSE is shown in 10^{-3} cross-entropy units; lower values are better. Lines connect the separately refitted evaluation settings only as visual guides.

B.4.2 LLLO_{30} OBSERVED VERSUS PREDICTED CROSS-ENTROPY

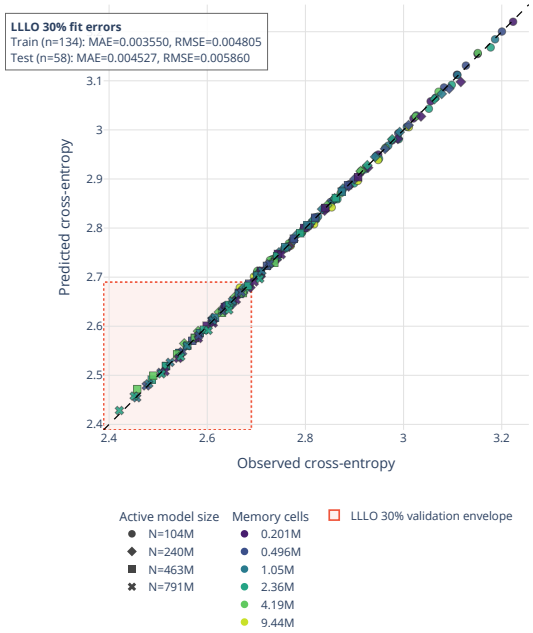


Figure 11. *PKM P6F LLLO_{30} observed versus predicted cross-entropy.* P6F is fitted on the highest-loss 70% of endpoints and evaluated on the held-out lowest-loss 30%. The shaded square marks the validation envelope and the dashed diagonal denotes exact prediction. Marker shape identifies N, and color identifies memory-table size.

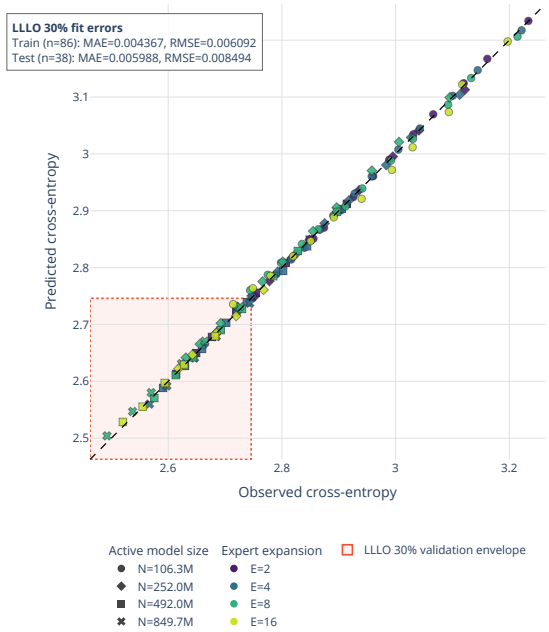


Figure 12. *MoE P6F LLLO_{30} observed versus predicted cross-entropy.* P6F is fitted on the highest-loss 70% of endpoints and evaluated on the held-out lowest-loss 30%. The shaded square marks the validation envelope and the dashed diagonal denotes exact prediction. Marker shape identifies N, and color identifies expert expansion.

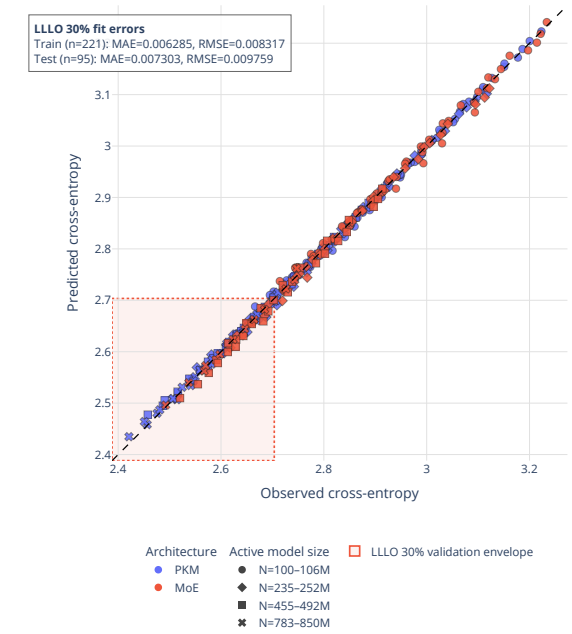


Figure 13. *Joint PKM–MoE P6F $LLLO_{30}$ observed versus predicted cross-entropy.* The joint PKM–MoE P6F law is fitted on the highest-loss 70% of endpoints and evaluated on the held-out lowest-loss 30%. The shaded square marks the validation envelope and the dashed diagonal denotes exact prediction. Marker shape identifies the shared active-model-size range, and color identifies architecture.

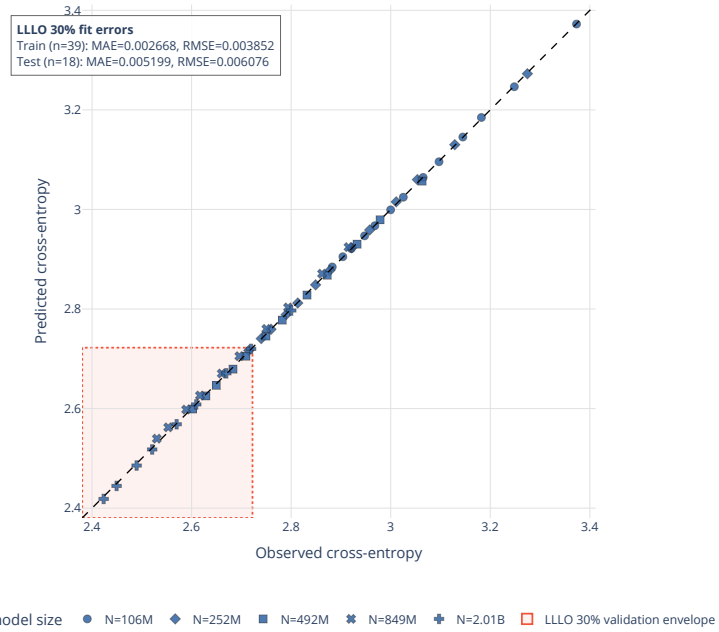


Figure 14. *Dense Transformer vP6F $LLLO_{30}$ observed versus predicted cross-entropy.* vP6F is fitted on the highest-loss 70% of endpoints and evaluated on the held-out lowest-loss 30%. The shaded square marks the validation envelope and the dashed diagonal denotes exact prediction. Marker shape identifies N.

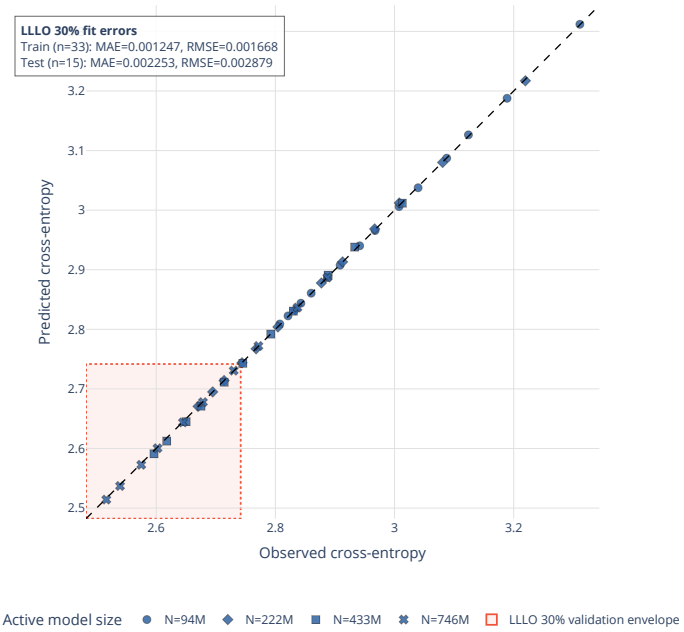


Figure 15. *STEM vP6F $LLLO_{30}$ observed versus predicted cross-entropy.* vP6F is fitted on the highest-loss 70% of endpoints and evaluated on the held-out lowest-loss 30%. The shaded square marks the validation envelope and the dashed diagonal denotes exact prediction. Marker shape identifies N.

B.4.3 Measured endpoints and fitted P6F and vP6F curves over tokens

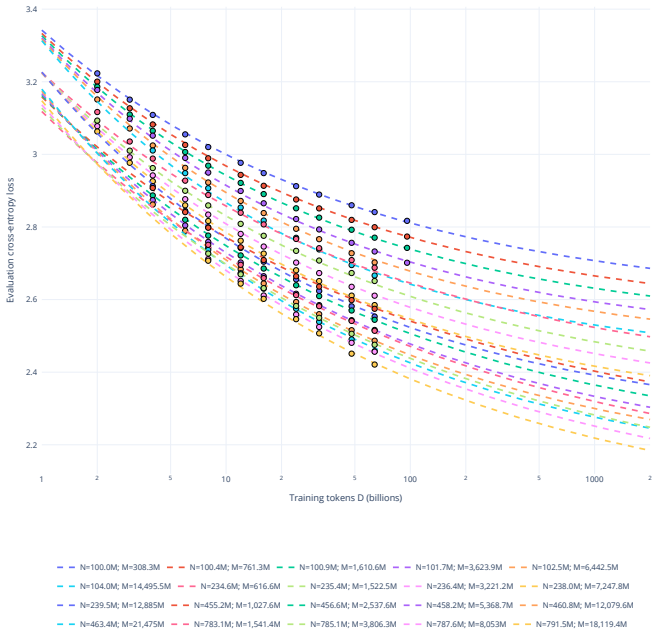


Figure 16. *PKM measured endpoints and fitted P6F curves over tokens.* Markers show measured trained-model endpoints; dashed lines show fitted and extrapolated P6F curves.

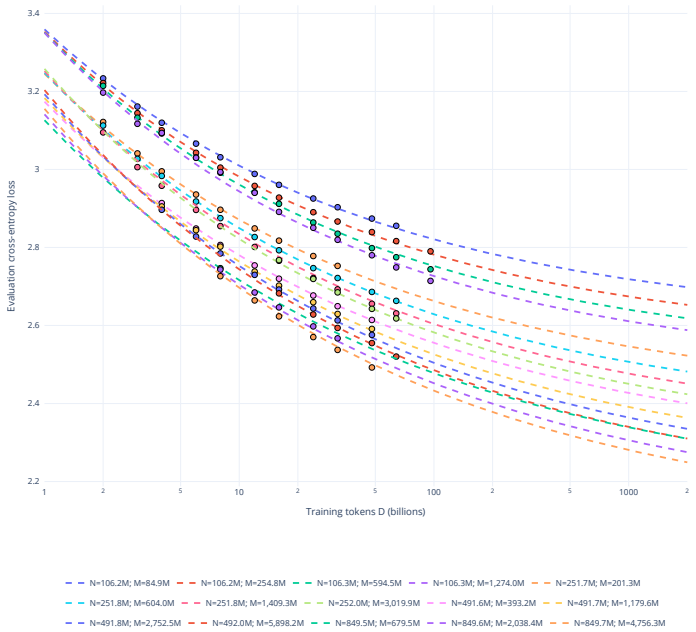


Figure 17. *MoE measured endpoints and fitted P6F curves over tokens.* Markers show measured trained-model endpoints; dashed lines show fitted and extrapolated P6F curves.

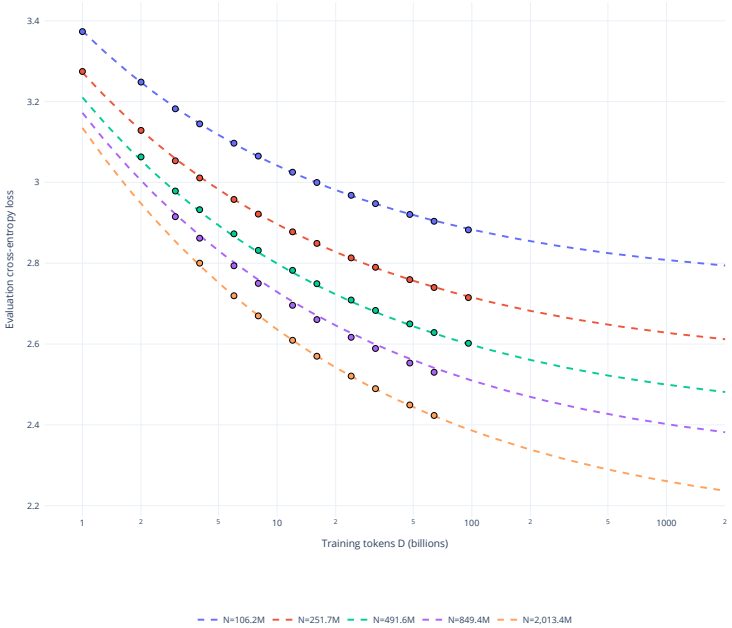


Figure 18. *Dense Transformer measured endpoints and fitted vP6F curves over tokens.* Markers show measured trained-model endpoints; dashed lines show fitted and extrapolated vP6F curves.

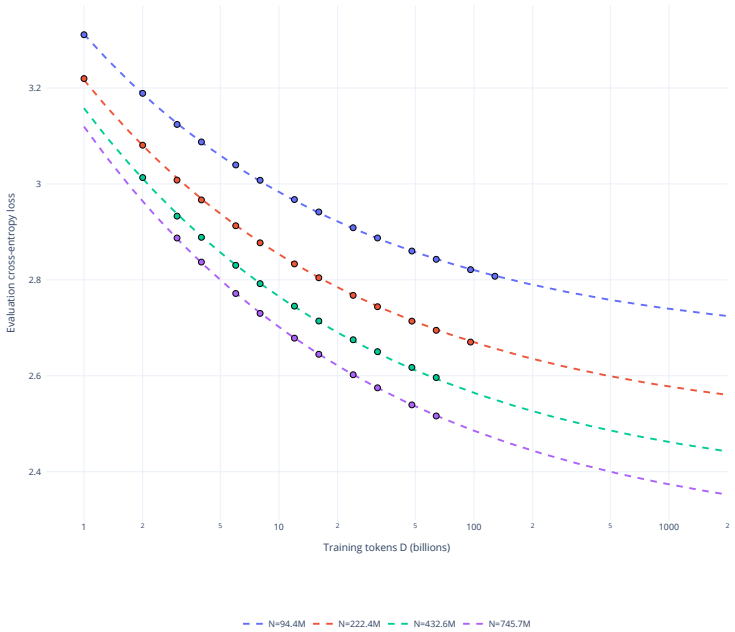


Figure 19. *STEM measured endpoints and fitted vP6F curves over tokens.* Markers show measured trained-model endpoints; dashed lines show fitted and extrapolated vP6F curves.

C Lowest-Loss Allocation at Fixed Compute and Total Parameters

Our fitted three-variable scaling law predicts final cross-entropy from the number of active non-embedding parameters N, the training-token budget D, and the number of inactive parameters M:

$$q_D(N, M) = \beta + \zeta \log M + \xi \log N, \tag{14}$$

$$\widehat{L}(N, D, M) = AM^{-\kappa}N^{-\alpha} + BN^{-\lambda}M^{-\mu}D^{-q_D(N,M)}. \tag{15}$$

Here, log is the natural logarithm and all parameter and token coordinates use the same raw-count units as the fitted law. The coefficient vector

$$\boldsymbol{\theta} = (A, B, \alpha, \beta, \kappa, \lambda, \mu, \zeta, \xi) \tag{16}$$

contains all fitted parameters of the law. The following calculations transform this function and introduce no additional fitted coefficients.

We seek the lowest predicted loss for a fixed total non-embedding parameter count T and a fixed training-compute budget C. The complete parameter budget is allocated between active and inactive parameters, so

$$T = N + M. \tag{17}$$

Using the active fraction $r = N/T$ and inactive fraction $S = M/T = 1 - r$ gives

$$N = rT, \qquad M = (1 - r)T, \qquad 0 < r < 1. \tag{18}$$

We approximate total training compute by

$$C = 6ND. \tag{19}$$

Consequently, fixing (T, C) and choosing r uniquely determines all three scaling-law coordinates:

$$N = rT, \qquad M = (1 - r)T, \qquad D = \frac{C}{6rT}. \tag{20}$$

The factor $6ND$ is an active-parameter FLOP approximation and excludes operation costs not represented by this compute model.

Substitution of equation 20 into equation 14 yields the allocation-dependent token exponent

$$q_D(r, T) = \beta + \zeta \log\big((1 - r)T\big) + \xi \log(rT). \tag{21}$$

Substitution into the complete loss law gives a one-dimensional objective for each fixed (T, C) pair:

$$\begin{aligned}\widehat{L}_{\mathrm{alloc}}(r; T, C) &:= \widehat{L}\left(rT, \frac{C}{6rT}, (1-r)T\right) \\ &= AT^{-(\alpha+\kappa)} r^{-\alpha}(1-r)^{-\kappa} \\ &\quad + BT^{-(\lambda+\mu)} r^{-\lambda}(1-r)^{-\mu} \left(\frac{C}{6rT}\right)^{-q_D(r,T)}.\end{aligned} \tag{22}$$

Thus, T and C are held fixed and the optimization changes only how T is partitioned between N and M. Every feasible allocation uses the complete total parameter count through the equality $N + M = T$.

Let $\mathcal{R}(T, C)$ denote the nonempty compact active-fraction set used for a reported calculation. It is chosen as a closed subset of the physically feasible region with positive coordinates and a positive effective token exponent:

$$\begin{aligned}\emptyset \neq \mathcal{R}(T, C) \subseteq \Big\{ & r \in (0,1) : N = rT > 0,\ M = (1-r)T > 0, \\ & D = \frac{C}{6rT} > 0,\ q_D(r,T) > 0 \Big\},\end{aligned} \tag{23}$$

$\mathcal{R}(T, C)$ is compact.

Any prescribed closed analysis-domain bounds on N, M, and D are incorporated into $\mathcal{R}(T, C)$. These restrictions change the feasible set but not the transformed objective. Because $\widehat{L}_{\mathrm{alloc}}(r; T, C)$ is continuous on the nonempty compact set $\mathcal{R}(T, C)$, its minimum is attained.

The globally optimal allocation is

$$\mathcal{R}^*(T, C) = \operatorname*{argmin}_{r \in \mathcal{R}(T,C)} \widehat{L}_{\mathrm{alloc}}(r; T, C). \tag{24}$$

If the minimizer is unique, we denote it by $r_{\mathrm{opt}}(T, C)$. If multiple allocations attain the same minimum, any $r_{\mathrm{opt}}(T, C) \in \mathcal{R}^*(T, C)$ gives the same optimal loss. The lowest predicted loss is therefore

$$\begin{aligned}\widehat{L}_{\mathrm{opt}}(T, C) &= \widehat{L}_{\mathrm{alloc}}(r_{\mathrm{opt}}(T, C); T, C) \\ &= \min_{r \in \mathcal{R}(T,C)} \widehat{L}_{\mathrm{alloc}}(r; T, C).\end{aligned} \tag{25}$$

The associated optimal inactive fraction, active parameters, inactive parameters, and training tokens are

$$S_{\mathrm{opt}}(T, C) = 1 - r_{\mathrm{opt}}(T, C), \tag{26}$$

$$N_{\mathrm{opt}}(T, C) = r_{\mathrm{opt}}(T, C)T, \tag{27}$$

$$M_{\mathrm{opt}}(T, C) = [1 - r_{\mathrm{opt}}(T, C)]\, T, \tag{28}$$

$$D_{\mathrm{opt}}(T, C) = \frac{C}{6r_{\mathrm{opt}}(T, C)T}. \tag{29}$$

Here, $S_{\mathrm{opt}} = M_{\mathrm{opt}}/T$ is the inactive-parameter fraction, reported as model sparsity.

The optimum generally has no closed-form power-law solution. To see why, write equation 22 as $\widehat{L}_{\mathrm{alloc}}(r; T, C) = U(T, r) + V(T, C, r)$, where U and V are respectively its first and second terms, and define

$$D(r) = \frac{C}{6rT}, \qquad q_D'(r, T) = -\frac{\zeta}{1-r} + \frac{\xi}{r}. \tag{30}$$

The logarithmic derivatives of U and V are

$$\frac{1}{U}\frac{\partial U}{\partial r} = -\frac{\alpha}{r} + \frac{\kappa}{1-r}, \tag{31}$$

$$\frac{1}{V}\frac{\partial V}{\partial r} = \frac{q_D(r,T) - \lambda}{r} + \frac{\mu}{1-r} - q_D'(r,T)\log D(r). \tag{32}$$

An interior stationary allocation must consequently satisfy

$$0 = U(T,r)\left(-\frac{\alpha}{r} + \frac{\kappa}{1-r}\right) + V(T,C,r)\left[\frac{q_D(r,T)-\lambda}{r} + \frac{\mu}{1-r} - q_D'(r,T)\log D(r)\right]. \tag{33}$$

Because $q_D(r,T)$ and $D(r)$ both depend on r, this equation contains coupled logarithmic and power-law terms. It may also admit multiple local minima, so solving one stationary equation from one initial value is insufficient to establish the global minimum.

We therefore solve equation 25 numerically. For a feasible interval $[r_{\min}, r_{\max}]$, the allocation is represented in logit coordinates,

$$u = \log\frac{r}{1-r}, \qquad r = \sigma(u) = \frac{1}{1+e^{-u}}. \tag{34}$$

A uniform grid in u provides denser coverage near both allocation boundaries than a uniform grid in r:

$$u_j = u_{\min} + \frac{j}{J-1}(u_{\max} - u_{\min}), \qquad r_j = \sigma(u_j), \qquad j = 0, \ldots, J-1. \tag{35}$$

The objective is evaluated at every feasible r_j. Every sampled point whose loss is no larger than its feasible neighbors is retained as a candidate local minimum; a feasible boundary is retained when it is a one-sided sampled minimum. Each interior candidate is refined independently by bounded scalar minimization over its neighboring grid interval. Finally, the candidate with the smallest refined loss is selected:

$$\widehat{r}_{\mathrm{opt}}(T,C) \in \operatorname*{argmin}_{r \in \mathcal{Q}(T,C)} \widehat{L}_{\mathrm{alloc}}(r;T,C), \tag{36}$$

where $\mathcal{Q}(T,C)$ contains the refined local minima and feasible boundary candidates. This multi-candidate procedure does not assume that $\widehat{L}_{\mathrm{alloc}}(r;T,C)$ is unimodal. A solution on a feasibility boundary is recorded as a boundary optimum because its allocation may be controlled by the imposed analysis domain rather than by an interior stationary point. The resulting $\widehat{r}_{\mathrm{opt}}(T,C)$ is the numerical approximation to $r_{\mathrm{opt}}(T,C)$ used when evaluating the reported quantities in Equations 25–29.

Equations 25–29 define the reported lowest-loss surface and its associated optimal allocation for the fitted scaling law. PKM and MoE use the same transformation and optimization procedure, but their independently fitted coefficient vectors produce separate functions $\widehat{L}_{\mathrm{opt}}(T,C)$.

D Additional Results

This appendix provides the MoE optimal-sparsity view corresponding to the PKM result in Figure 3a, a fixed-budget comparison of the sparsity selected between Dense and MoE, the coupled resource-scaling path of Section 5.4 without the operational cost, and the mean absolute error (MAE) tables corresponding to the root-mean-square error results in Appendix B.

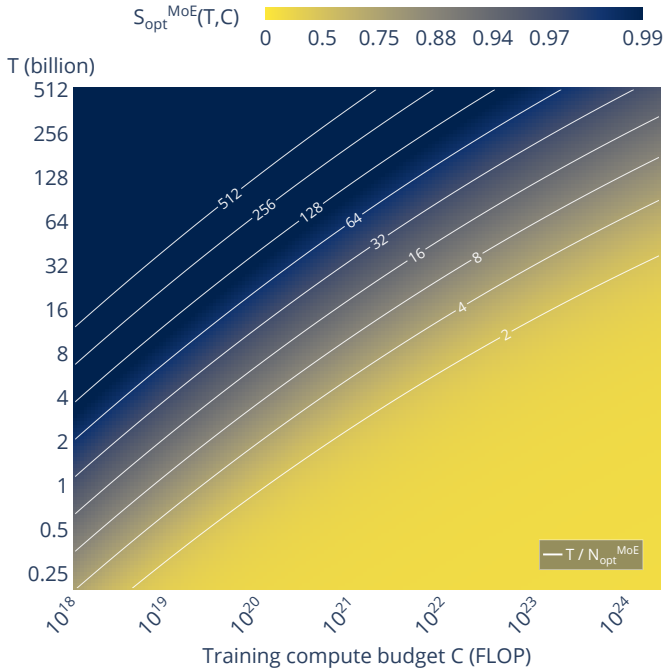


Figure 20. *MoE optimal sparsity S.* The optimal inactive fraction $S^{\mathrm{MoE}}_{\mathrm{opt}}(T, C) = M^{\mathrm{MoE}}_{\mathrm{opt}}(T, C)/T$ is shown over training-compute and total-parameter budgets. The axes, color mapping, and $T/N^{\mathrm{MoE}}_{\mathrm{opt}}$ contour levels match Figure 3a.

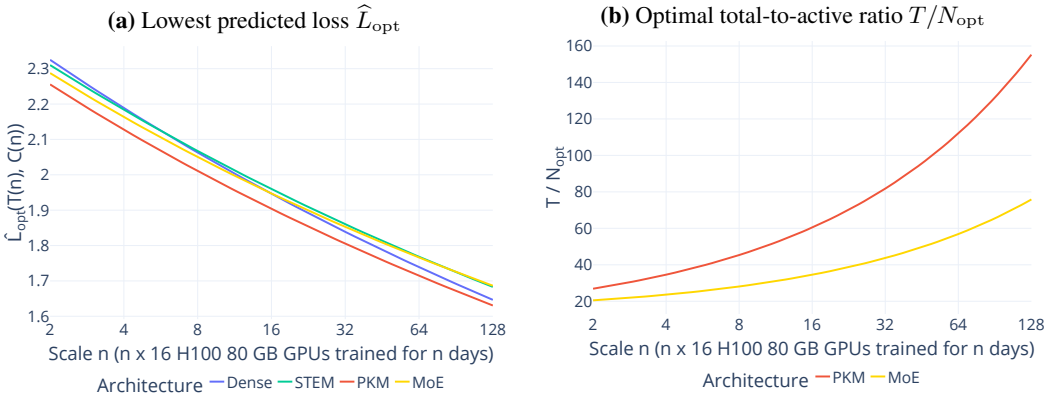


Figure 21. *Coupled GPU-and-time scaling without the operational cost.* Same budget path as Figure 4, $C(n) = 4.424 \times 10^{20} n^2$ FLOP and $T(n) = 14.69n$ billion parameters, but with $D_{\mathrm{op}} = 0$, so the entire compute budget is spent on training, $C(n) = 6ND$. (a) Lowest predicted loss $\widehat{L}_{\mathrm{opt}}(T(n), C(n))$ of each architecture. (b) $T(n)/N_{\mathrm{opt}}(T(n), C(n))$ for the independently optimized PKM and MoE allocations.

Table 10. *PKM MAE (all values in 10^{-3} cross-entropy units).*

Formula	Full-data and LLLO						Grouped and directional holdouts						
	Full	$LLLO_{10}$	$LLLO_{20}$	$LLLO_{30}$	$LLLO_{40}$	$LLLO_{50}$	LOMO	LOSO	LOCO	CF	FT	UCF	LC
P0	29.6	37.4	45.7	52.2	57.5	104	39.6	35.0	43.5	51.0	39.9	84.7	70.7
P1A	8.69	15.4	24.9	28.9	28.4	31.1	9.53	13.0	8.69	9.43	16.3	15.4	25.6
P1B	9.67	21.7	26.3	27.7	25.5	26.6	11.6	15.6	10.7	11.3	16.2	16.7	26.1
P1C	11.5	13.8	15.4	25.8	28.2	36.9	12.9	15.5	12.1	12.8	24.6	22.6	21.3
P2B	9.67	21.7	26.3	27.7	25.5	26.6	10.8	15.6	10.7	11.3	24.6	16.7	26.1
P3A	9.41	20.9	26.3	27.8	25.8	26.8	10.3	14.6	10.1	10.7	15.9	16.0	(i)
P3B	8.18	20.8	23.5	24.1	22.1	21.2	10.6	17.4	10.6	12.2	12.4	13.7	5.19
P3C	7.80	20.2	22.7	23.3	21.6	20.9	9.81	(i)	9.71	(i)	12.1	12.6	7.40
P4	5.74	8.59	9.01	9.52	9.10	9.17	8.21	12.1	8.44	9.25	9.70	14.2	5.44
P5	5.80	9.03	9.59	10.2	9.62	9.08	7.75	(i)	8.40	9.41	9.50	14.1	13.7
P6B	4.73	7.65	6.96	7.23	8.01	8.17	5.98	9.18	6.77	7.99	8.74	12.2	7.42
P6C	3.92	4.89	6.65	(i)	8.84	7.68	(i)	9.86	5.30	6.05	7.24	9.12	5.65
P6D	**3.48**	**3.27**	**3.65**	**4.53**	8.94	**7.53**	**4.30**	**6.74**	**5.08**	**5.67**	5.91	(i)	2.85
P6F	**3.48**	**3.27**	**3.65**	**4.53**	**5.76**	**7.53**	4.58	7.10	**5.08**	**5.67**	5.91	**7.77**	2.85
P6G	5.47	8.95	8.23	9.38	(ii)	9.76	(i, ii)	(ii)	(i, ii)	8.60	8.37	(i, ii)	**2.51**
P6I	5.84	(i, ii)	8.17	(i, ii)	20.6	9.79	(i, ii)	15.7	(i, ii)	(ii)	8.89	(i, ii)	3.82
P6J	(ii)	(ii)	(ii)	(ii)	(ii)	9.29	(ii)	(ii)	(ii)	(ii)	(ii)	(ii)	3.39
P6K	4.15	5.14	5.31	6.88	8.22	9.15	6.30	8.85	6.42	6.16	**5.66**	8.40	(i, ii)
P7	5.54	8.31	8.47	9.08	8.90	8.94	7.15	9.87	7.95	9.04	9.14	13.3	13.4

Table 11. *MoE MAE (all values in 10^{-3} cross-entropy units).*

Formula	Full-data and LLLO						Grouped and directional holdouts						
	Full	$LLLO_{10}$	$LLLO_{20}$	$LLLO_{30}$	$LLLO_{40}$	$LLLO_{50}$	LOMO	LOSO	LOCO	CF	FT	UCF	LC
P0	16.9	18.5	19.0	24.1	40.3	43.0	22.7	18.7	25.3	31.5	34.8	38.2	44.8
P1A	7.25	10.5	13.1	19.6	28.3	31.8	9.78	9.33	9.14	9.94	11.4	13.9	26.5
P1B	7.32	10.7	12.4	23.0	25.6	30.1	11.2	9.92	8.67	9.44	11.9	13.4	20.1
P1C	8.86	9.22	13.2	17.6	35.1	36.0	10.2	11.0	11.2	12.3	15.8	19.3	21.8
P2B	7.32	9.22	12.4	23.0	25.6	30.1	8.55	9.92	9.30	9.44	15.8	13.4	21.8
P3A	7.21	11.0	12.6	22.3	25.5	30.0	8.25	9.86	8.56	9.22	11.8	13.4	25.3
P3B	5.81	5.30	7.30	15.5	14.3	19.5	8.35	10.5	8.53	11.2	6.29	8.05	14.7
P3C	5.57	5.37	6.81	13.9	13.3	18.9	7.62	(i)	8.42	11.0	**6.10**	**7.89**	15.9
P4	4.50	5.07	4.86	7.13	6.87	7.47	7.35	7.59	**6.90**	9.47	6.62	10.9	13.6
P5	4.65	5.10	5.08	8.26	7.46	8.45	**6.17**	**5.78**	7.14	9.81	6.82	11.2	14.4
P6B	4.43	4.83	4.58	6.39	6.60	7.19	(i)	7.39	7.37	**9.00**	6.51	10.5	13.0
P6C	4.29	**4.01**	4.58	6.89	7.31	8.10	6.57	7.03	7.81	10.0	6.26	9.11	14.4
P6D	4.28	4.28	4.58	**5.99**	**6.53**	7.19	(i)	(i)	(i)	10.3	6.16	9.25	14.2
P6F	**4.27**	4.28	**4.50**	**5.99**	**6.53**	**6.83**	(i)	6.98	(i)	(i)	6.18	(i, ii)	14.2
P6G	4.68	5.38	5.29	6.33	7.14	7.68	6.65	(i, ii)	(ii)	(i, ii)	6.59	(ii)	(i, ii)
P6I	4.67	5.34	5.30	6.94	15.7	(i, ii)	(i, ii)	(i, ii)	(i, ii)	10.1	6.64	14.0	(i, ii)
P6J	4.66	5.50	5.38	6.37	7.11	7.65	6.63	7.80	(ii)	14.7	6.60	(ii)	**1.98**
P6K	4.41	5.06	4.93	6.61	7.31	7.77	7.20	7.18	7.93	10.4	6.29	8.83	8.82
P7	4.54	4.93	4.76	7.18	7.04	7.66	(i)	6.77	7.09	9.43	6.66	10.6	14.7

Table 12. *Dense Transformer MAE (all values in 10^{-3} cross-entropy units).*

Formula	Full	$LLLO_{10}$	$LLLO_{20}$	$LLLO_{30}$	$LLLO_{40}$	$LLLO_{50}$	LOMO	FT	LC
Chinchilla	13.8	41.5	46.0	46.2	53.5	51.2	20.9	22.0	27.2
vP6F	3.32	**5.34**	**5.24**	**5.20**	**7.55**	**8.38**	**11.1**	**4.52**	2.30
Farseer	**3.25**	5.50	5.97	5.46	8.37	29.1	46.0	5.21	**1.66**

Table 13. *STEM MAE (all values in 10^{-3} cross-entropy units).*

Formula	Full	$LLLO_{10}$	$LLLO_{20}$	$LLLO_{30}$	$LLLO_{40}$	$LLLO_{50}$	LOMO	FT	LC
Chinchilla	9.16	25.1	26.5	30.2	30.1	30.3	14.2	14.2	25.8
vP6F	1.43	**0.85**	**2.33**	**2.25**	3.74	**4.43**	**4.11**	2.19	**1.11**
Farseer	**1.15**	4.42	3.41	6.10	**2.88**	11.9	12.0	**2.16**	4.18

Table 14. *Joint PKM–MoE MAE (all values in 10^{-3} cross-entropy units).*

Formula	Full-data and LLLO						Grouped and directional holdouts						
	Full	LLLO_{10}	LLLO_{20}	LLLO_{30}	LLLO_{40}	LLLO_{50}	LOMO	LOSO	LOCO	CF	FT	UCF	LC
P0	29.5	39.7	36.5	40.8	52.8	65.7	32.1	30.2	37.7	44.4	38.1	46.8	55.7
P1A	9.88	20.3	19.9	22.4	35.3	27.9	10.7	12.1	11.0	11.4	15.3	17.4	42.1
P1B	10.3	40.6	22.6	22.9	26.2	26.8	11.4	15.6	13.3	11.6	16.1	16.5	49.2
P1C	12.9	21.2	20.7	26.9	30.9	35.9	14.2	14.1	15.3	15.5	25.0	25.7	47.9
P2B	10.3	21.2	20.7	22.9	26.2	26.8	11.2	15.8	12.9	11.6	25.0	16.5	49.2
P3A	10.1	23.1	(i)	22.8	26.5	(i)	(i)	(i)	(i)	11.4	16.1	16.8	49.5
P3B	9.80	20.0	22.0	21.5	24.8	25.5	11.2	17.3	10.9	12.8	14.6	14.3	37.5
P3C	9.58	21.7	21.2	21.0	24.5	25.5	(i)	(i)	10.7	12.7	14.7	14.8	38.8
P4	8.11	16.0	12.4	11.1	12.1	11.9	9.82	14.6	10.8	11.4	12.5	15.9	36.9
P5	8.17	15.2	12.8	11.3	12.2	11.7	9.84	11.6	10.9	11.5	12.5	15.5	37.1
P6B	7.47	17.6	9.52	10.0	10.7	10.5	8.79	9.22	(i)	10.5	11.3	14.4	34.9
P6C	6.89	8.66	9.27	8.80	10.1	9.96	(i)	(i)	(i)	(i)	9.38	(i)	33.5
P6D	**6.47**	**6.46**	**6.51**	(i)	10.7	**9.70**	(i)	**7.98**	**8.01**	**9.20**	**8.61**	(i)	37.5
P6F	**6.47**	**6.46**	**6.51**	**7.30**	**8.91**	**9.70**	**7.51**	**7.98**	**8.01**	(i)	**8.61**	11.9	(i)
P6G	7.62	11.1	10.1	11.1	12.5	(i, ii)	9.38	12.4	12.0	(i, ii)	10.7	14.9	**7.96**
P6I	8.33	11.5	10.6	12.8	(i, ii)	20.8	10.8	15.3	(i, ii)	13.8	11.3	(i)	24.3
P6J	7.77	11.2	11.4	15.4	13.0	12.1	9.45	14.2	(ii)	(ii)	10.8	(ii)	21.0
P6K	7.23	7.77	7.87	11.6	12.6	12.5	8.71	10.9	9.09	9.71	8.96	**11.5**	14.5
P7	8.01	14.3	11.8	10.8	11.9	11.4	9.54	10.8	10.7	11.2	12.0	15.2	36.4

E Roofline Analysis Formulas: Derivations and Parameters

This appendix derives the quantities F, b_{HBM}, b_{DRAM}, and b_{flash} for each architecture and regime, and documents the hardware parameters used in Section 5.5. Table 15 defines all symbols used.

E.1 Step-Time Roofline Model

Training at Scale. The available training resource is one GB200 NVL72 node, so EP, TP, and PP communication all use $\mathcal{B}_{\text{NVLink}}$ and placements requiring more than 72 GPUs are invalid.

$$t_{\text{step}} = \max\big(t_{\text{compute}},\ t_{\text{HBM}}\big) + t_{\text{A2A}}, \qquad t_{\text{compute}} = \frac{F}{\alpha_F F_{\text{peak}}}, \quad t_{\text{HBM}} = \frac{b_{\text{HBM}}}{\alpha_{\text{HBM}} \mathcal{B}_{\text{HBM}}}, \quad t_{\text{A2A}} = \frac{b_{\text{A2A}}}{\alpha_{\text{NVLink}} \mathcal{B}_{\text{NVLink}}}, \tag{37}$$

where t_{compute} is time to perform all FLOPs at sustained throughput $\alpha_F F_{\text{peak}}$, t_{HBM} is time to stream weights from HBM, and t_{A2A} is GPU-to-GPU communication time, including EP all-to-all, TP All-Reduce, and PP boundary transfers, which does not overlap with compute. For a per-GPU training step containing τ tokens, the token throughput is

$$\Theta_{\text{GPU}} = \frac{\tau}{t_{\text{step}}} \qquad \text{tokens/s/GPU}. \tag{38}$$

Inference at the Edge.

$$t_{\text{step}} = \max\big(t_{\text{compute}},\ t_{\text{DRAM}},\ t_{\text{flash}}\big), \qquad t_{\text{DRAM}} = \frac{b_{\text{DRAM}}}{\alpha_{\text{DRAM}} \mathcal{B}_{\text{DRAM}}}, \quad t_{\text{flash}} = \frac{b_{\text{flash}}}{\alpha_{\text{flash}} \mathcal{B}_{\text{flash}}}, \tag{39}$$

where t_{DRAM} streams static parameters and KV cache, while t_{flash} pages dynamically selected parameters. The efficiency factors convert peak hardware rates into sustained effective rates. There is no communication term because inference runs on a single device.

E.2 FLOPs per Step (F)

E.2.1 FLOPs per Layer

Attention. With grouped-query attention, let n_{kv} be the number of key/value heads and d_{head} the per-head dimension. FLOPs per token, per layer:

$$F_{\text{attn}} = 2\big(\underbrace{2d_{\text{model}}^2 + 2d_{\text{model}} n_{\text{kv}} d_{\text{head}}}_{\text{Q, K, V, O projections}} + \underbrace{2d_{\text{model}} \cdot C}_{\text{QK}^{\text{T}} + \text{AV}}\big). \tag{40}$$

This reduces to the usual $4d_{\text{model}}^2$ projection term when $n_{\text{kv}} d_{\text{head}} = d_{\text{model}}$.

FFN Dense. Three projections (gate, up, down) of size $d_{\text{model}} \times d_{\text{ff}}$:

$$F_{\text{FFN}}^{\text{Dense}} = 2 \times 3\, d_{\text{model}}\, d_{\text{ff}} = 6\, d_{\text{model}}\, d_{\text{ff}}. \tag{41}$$

Table 15: Notation used in this appendix.

Symbol	Definition
Architecture	
d_{model}	Model width
d_{ff}	FFN intermediate size
$d_{\text{ff}}^{\text{exp}}$	Expert intermediate size $= d_{\text{ff}}/k$
d_v	PKM value-vector dimension
L	Number of layers
k	Top-k activated (MoE)
E	Number of experts (MoE)
P_e	Params per expert $= 3d_{\text{model}}\, d_{\text{ff}}^{\text{exp}}$
n_{heads}	PKM router heads
d_q	Query dim per router head $=$ rank $\cdot\, d_{\text{head}}$
d_{head}	Per-axis key/query dim (PKM router)
rank	Product-key factorisation rank
$n_{\text{per_axis}}$	Keys per axis (PKM router)
k_{total}	Total top-k retrievals $= n_{\text{heads}} \times k$
Δ	PKM layer interval: 1 every Δ layers
Sequence lengths and counts	
W	Processing window (tokens per forward pass)
C	Context length (KV cache size)
τ	Tokens per GPU per training step (batch $\times$ seq. length)
D	Total training token count (dataset)
Quantization	
q_{act}	Bytes per activation element (training, bf16)
q_w	Bytes per stored weight; precision depends on regime (e.g. int4 on mobile)
q_{kv}	Bytes per KV-cache element (mobile, int8)
Hardware	
F_{peak}	Peak compute throughput
α_F	Compute-throughput efficiency factor
G_{HBM}	HBM capacity per GPU
$\mathcal{B}_{\text{HBM}}$	Peak HBM bandwidth
α_{HBM}	HBM bandwidth efficiency factor
G_{DRAM}	Mobile DRAM capacity
α_G^{DRAM}	Usable mobile-DRAM capacity fraction
$\mathcal{B}_{\text{DRAM}}$	Peak DRAM bandwidth
α_{DRAM}	DRAM bandwidth efficiency factor
$\mathcal{B}_{\text{NVLink}}$	Peak NVLink bandwidth
α_{NVLink}	NVLink bandwidth efficiency factor
Θ_{GPU}	Training throughput in tokens/s/GPU $= \tau/t_{\text{step}}$
$\mathcal{B}_{\text{flash}}$	Peak flash bandwidth
α_{flash}	Flash bandwidth efficiency factor

FFN MoE. Top-k experts, each with intermediate size $d_{\text{ff}}^{\text{exp}} = d_{\text{ff}}/k$ (ensuring matched active compute):

$$F_{\text{FFN}}^{\text{MoE}} = 2 \cdot k \cdot 3\, d_{\text{model}}\, d_{\text{ff}}^{\text{exp}} = 2 \cdot k \cdot 3\, d_{\text{model}}\, \frac{d_{\text{ff}}}{k} = 6\, d_{\text{model}}\, d_{\text{ff}} = F_{\text{FFN}}^{\text{Dense}}. \tag{42}$$

FFN PKM. PKM has *no up projection*: a **projection** (gate + down, size $d_{\text{model}} \times d_v$) and a **router** (query projection + product-key lookup over n_{heads} heads):

$$F_{\text{FFN}}^{\text{PKM}} = \underbrace{4 d_{\text{model}} d_v}_{\text{gate+down}} + \underbrace{2\, n_{\text{heads}} (d_{\text{model}}\, d_q + \text{rank} \cdot n_{\text{per_axis}}\, d_{\text{head}})}_{\text{router}}. \tag{43}$$

The retrieved value vectors ($k_{\text{total}} = n_{\text{heads}} \times k$ values of dimension d_v) are loaded from flash but contribute negligible multiply-adds.

E.2.2 FLOPs per GPU

We derive the per-GPU FLOPs from the per-layer FLOPs above.

FLOPs per GPU per layer. Column-wise Tensor Parallelism (TP) shards PKM value storage. Expert Parallelism (EP) shards expert ownership across GPUs. Given τ defined as the per-GPU token count, balanced routing leaves each GPU processing approximately τ expert-token assignments. With a batch of τ tokens, the average FLOPs per GPU per layer are:

$$f_{\text{GPU}}^{\text{MoE}} = \tau\big(F_{\text{attn}} + F_{\text{FFN}}^{\text{MoE}}\big), \tag{44}$$

$$f_{\text{GPU}}^{\text{PKM}} = \tau\left(F_{\text{attn}} + \frac{F_{\text{FFN}}^{\text{PKM}}}{\Delta} + \left(1 - \frac{1}{\Delta}\right) F_{\text{FFN}}^{\text{Dense}}\right). \tag{45}$$

FLOPs per GPU per step. Pipeline Parallelism (PP) assigns L/PP consecutive layers to each GPU group. Training requires a forward pass plus backward activation and weight gradients, which we approximate as $m_{\text{train}} = 3$ times the forward FLOPs. This factor is the usual matmul accounting: each weight-using linear layer does one GEMM in the forward pass and two GEMMs in backward. The total per-step FLOPs per GPU are therefore:

$$F_{\text{GPU}}^{\text{MoE}} = m_{\text{train}} \frac{L}{\text{PP}} \cdot \tau\big(F_{\text{attn}} + F_{\text{FFN}}^{\text{MoE}}\big), \tag{46}$$

$$F_{\text{GPU}}^{\text{PKM}} = m_{\text{train}} \frac{L}{\text{PP}} \cdot \tau\left(F_{\text{attn}} + \frac{F_{\text{FFN}}^{\text{PKM}}}{\Delta} + \left(1 - \frac{1}{\Delta}\right) F_{\text{FFN}}^{\text{Dense}}\right). \tag{47}$$

E.3 EP/TP/PP Computation and Placement

All parallelism choices are constrained by the single NVL72 resource envelope. For MoE, each pipeline stage uses an EP group, so the simulated placement must satisfy

$$\text{PP} \cdot \text{EP} \le 72. \tag{48}$$

To fit all E experts within GPU memory, EP must also satisfy

$$\text{EP} \;\ge\; \frac{E \cdot P_e \cdot q_w \cdot L_{\text{PP}}}{G_{\text{HBM}}}, \tag{49}$$

where $L_{\text{PP}} = L/p$ and the numerator is the total expert weight bytes per pipeline stage. Since $P_e = 3 d_{\text{model}} d_{\text{ff}}/k \propto d_{\text{model}}^2$ and $L_{\text{PP}} \propto d_{\text{model}}$ in a balanced-depth configuration,

$$\text{EP} \;\propto\; \frac{E \cdot d_{\text{model}}^3}{G_{\text{HBM}}}. \tag{50}$$

For PKM, each pipeline stage uses a TP group, giving the analogous constraint

$$\text{PP} \cdot \text{TP} \le 72. \tag{51}$$

Operating points that violate these bounds fall outside the single-NVL72 resource envelope and are excluded from the roofline curves.

Rather than jointly sweeping EP/TP and PP, the simulator uses the monotonic structure of the roofline. Candidate pipeline degrees must divide the layer count and leave at least one PKM interval per pipeline stage,

$$\mathcal{P}(L) = \{p : p \mid L,\ p \le 72,\ L/p \ge \Delta\}. \tag{52}$$

This avoids imbalanced stages in which some stages contain a PKM layer and others do not. For each $p \in \mathcal{P}(L)$, the simulator first sets the sharding degree to the smallest value that satisfies the memory constraint, then selects the pipeline degree with maximum tokens/s/GPU. For MoE,

$$\text{EP}_{\min}(p) = \min_{c} \left\{ c : \left\lceil \frac{E}{c} \right\rceil P_e q_w \frac{L}{p} \le B_{\text{avail}}^{\text{MoE}}(p) \right\}, \tag{53}$$

$$p_{\text{MoE}}^{\star} = \underset{p \in \mathcal{P}(L):\, p\,\text{EP}_{\min}(p) \le 72}{\arg\max} \Theta_{\text{GPU}}^{\text{MoE}}\big(p, \text{EP}_{\min}(p)\big). \tag{54}$$

For PKM,

$$L_{\text{PKM}}^{\max}(p) = \max_{s \in \{1,\dots,p\}} \#\{\text{PKM layers assigned to stage } s\}, \tag{55}$$

$$\text{TP}_{\min}(p) = \left\lceil \frac{B_{\text{bank}}\, L_{\text{PKM}}^{\max}(p)}{B_{\text{avail}}^{\text{PKM}}(p)} \right\rceil, \tag{56}$$

$$p_{\text{PKM}}^{\star} = \underset{p \in \mathcal{P}(L):\, p\,\text{TP}_{\min}(p) \le 72}{\arg\max} \Theta_{\text{GPU}}^{\text{PKM}}\big(p, \text{TP}_{\min}(p)\big). \tag{57}$$

Here $B_{\text{avail}}(p)$ is the per-GPU HBM left after static parameters, activations, and a reserve margin, and B_{bank} is the per-layer PKM value-bank footprint before TP sharding. We use the maximum number of PKM banks resident on any pipeline stage because HBM capacity is a per-GPU constraint. Thus EP/TP are never increased beyond the minimum needed to fit memory; PP is the only enumerated degree of freedom.

We include pipeline bubbles using $M = 16$ pipeline microbatches. The pipeline efficiency for candidate pipeline degree p is

$$\eta_{\text{PP}}(p) = \frac{M}{M + p - 1}, \qquad M = 16, \tag{58}$$

and the reported throughput applies this multiplicative factor to the ideal steady-state tokens/s/GPU.

EP reduces resident expert weights per GPU, while attention and the active expert computation for the local per-GPU token batch remain unsharded in the per-GPU step model. In PKM, TP is used to fit the value bank in memory and to synchronise retrieved representations, but it does not divide the PKM layer FLOPs. Attention, PKM compute, and the dense FFN layers between PKM layers are therefore replicated within each pipeline stage. PP reduces the number of resident layers from L to L/PP and adds the pipeline activation/gradient traffic included in b_{PP}.

E.4 Memory and Communication Bytes per Step

E.4.1 b_{HBM} - Training at Scale

In a single forward pass every active weight is streamed from HBM once. Training additionally requires reading weights in the backward pass and reading/writing gradients and optimizer states. We approximate these streams with the same effective bytes-per-element q_w:

$$b_{\text{HBM}} \approx 4\,\frac{N}{\text{PP}}\,q_w, \tag{59}$$

where N is the full-model active-parameter count used throughout the paper and q_w is the effective byte width for training weight/gradient-state traffic. Because PP assigns each GPU group roughly $1/\text{PP}$ of the layers, the per-GPU HBM traffic entering the step-time model scales as N/PP. Activation tensors are included in the HBM capacity budget but not in this bandwidth term; this keeps b_{HBM} focused on repeated weight/gradient-state streams. For matched active N, this baseline HBM term is comparable across Dense, MoE, and PKM.

E.4.2 b_{A2A} - Training at Scale

MoE performs all-to-all collectives to dispatch tokens across EP GPUs and combine expert outputs, in both forward and backward passes. The per-layer expert-parallel traffic is

$$b_{\text{EP}}^{\text{MoE}} = 4\,k\,\frac{\text{EP} - 1}{\text{EP}}\,\tau\,d_{\text{model}}\,q_{\text{act}}. \tag{60}$$

PKM synchronises the retrieved value representation via an All-Reduce over the TP group, in both forward and backward passes. The per-PKM-layer tensor-parallel traffic is

$$b_{\text{TP}}^{\text{PKM}} = 2\,\frac{\text{TP} - 1}{\text{TP}}\,\tau\,d_{\text{model}}\,q_{\text{act}}. \tag{61}$$

PP adds forward activation transfers and backward gradient transfers across pipeline boundaries. We use the maximum per-stage traffic across pipeline stages, counting one forward/backward pair for each adjacent pipeline neighbor,

$$b_{\text{PP}} = 2\,\min(\text{PP} - 1, 2)\,\tau\,d_{\text{model}}\,q_{\text{act}}. \tag{62}$$

The per-GPU step-time model therefore uses

$$b_{\text{A2A}}^{\text{MoE}} = \frac{L}{\text{PP}}\,b_{\text{EP}}^{\text{MoE}} + b_{\text{PP}}, \tag{63}$$

$$b_{\text{A2A}}^{\text{PKM}} = \frac{L}{\text{PP}\Delta}\,b_{\text{TP}}^{\text{PKM}} + b_{\text{PP}}. \tag{64}$$

Ignoring the common PP boundary term, their ratio is

$$\frac{b_{\text{A2A}}^{\text{PKM}}}{b_{\text{A2A}}^{\text{MoE}}} \approx \frac{1}{2k\Delta} \frac{(\text{TP}-1)/\text{TP}}{(\text{EP}-1)/\text{EP}}, \tag{65}$$

so PKM has substantially lower communication whenever TP and EP are comparable; for k=1 and Δ=4, the non-PP term is about 8× smaller at large EP/TP.

Throughput degradation with EP and TP. Since t_{A2A} adds directly to step time, tokens/s/GPU degrades for all $\text{EP} > 1$: as EP grows, per-GPU expert storage falls while t_{A2A} grows toward $4k\tau d_{\text{model}}\, q_{\text{act}}/(\alpha_{\text{NVLink}}\mathcal{B}_{\text{NVLink}})$ per layer. Since EP is forced upward by the memory constraint (Eq. 49), large models incur compounding communication overhead and training throughput collapses.

For PKM, growing the value bank increases capacity without increasing $b_{\text{TP}}^{\text{PKM}}$ at a fixed TP degree: the All-Reduce payload depends on the active representation size, not on the number of stored values. If the bank no longer fits in HBM, the memory constraint can still force a larger TP degree, which increases the factor $(\text{TP}-1)/\text{TP}$. At TP=1, this factor is zero and no TP communication occurs.

E.4.3 b_{DRAM} - Inference at the Edge

Attention parameters and KV cache are held in DRAM. Both MoE and PKM also keep their router weights in DRAM. In a PKM model, only L/Δ layers are PKM; the remaining $L(1-1/\Delta)$ regular layers keep their dense FFN in DRAM as well:

$$b_{\text{DRAM}}^{\text{Dense}} = L\big(\underbrace{4d_{\text{model}}^2\, q_w}_{\text{attn weights}} + \underbrace{2d_{\text{model}} \cdot C\, q_{\text{kv}}}_{\text{KV cache}} + \underbrace{3\, d_{\text{model}}\, d_{\text{ff}}\, q_w}_{\text{FFN weights}}\big), \tag{66}$$

$$b_{\text{DRAM}}^{\text{MoE}} = L\big(\underbrace{4d_{\text{model}}^2\, q_w}_{\text{attn weights}} + \underbrace{2d_{\text{model}} \cdot C\, q_{\text{kv}}}_{\text{KV cache}} + \underbrace{d_{\text{model}} \cdot E\, q_w}_{\text{router}}\big), \tag{67}$$

$$\begin{aligned} b_{\text{DRAM}}^{\text{PKM}} = {} & L\big(\underbrace{4d_{\text{model}}^2\, q_w}_{\text{attn weights}} + \underbrace{2d_{\text{model}} \cdot C\, q_{\text{kv}}}_{\text{KV cache}}\big) + \tfrac{L}{\Delta}\big(\underbrace{2d_{\text{model}} d_v\, q_w}_{\text{gate+down}} + \underbrace{N_{\text{router}}\, q_w}_{\text{router}}\big) \\ & + L\big(1-\tfrac{1}{\Delta}\big)\underbrace{3\, d_{\text{model}}\, d_{\text{ff}}\, q_w}_{\text{regular FFN}}, \end{aligned} \tag{68}$$

where C is context length. Because the $2d_{\text{model}} C\, q_{\text{kv}}$ KV-cache term grows with context length while all weight terms are fixed, b_{DRAM} is dominated by the KV cache at long context for all three architectures.

E.4.4 b_{FLASH} - Inference at the Edge

Static parameters (same weights used regardless of the input token) are held in DRAM. *Dynamic* parameters (selected per token by the routing or retrieval mechanism) are paged from flash.

For **Dense**, all FFN parameters are static: $b_{\text{flash}}^{\text{Dense}} = 0$.

For **MoE**, the activated expert matrices are dynamic; $\min(E, W)$ distinct expert matrices are loaded per window of W tokens:

$$b_{\text{flash}}^{\text{MoE}} = L\, q_w\, \min(E,\, W) \cdot P_e, \qquad P_e = 3\, d_{\text{model}}\, d_{\text{ff}}^{\text{exp}} = \frac{3\, d_{\text{model}}\, d_{\text{ff}}}{k}. \tag{69}$$

For **PKM**, the retrieved value vectors are dynamic; only the L/Δ PKM layers contribute flash traffic:

$$b_{\text{flash}}^{\text{PKM}} = \frac{L}{\Delta}\, q_w\, W\, k_{\text{total}}\, d_v, \qquad k_{\text{total}} = n_{\text{heads}} \times k. \tag{70}$$

With $P_e \approx 78.6$M and $k_{\text{total}}\, d_v \approx 0.3$M, the per-token flash gap is $78.6/0.3 \approx 260\times$, independent of T/N.

Arithmetic intensity. Both MoE and PKM are equally flash-bandwidth-bound per loaded byte: $\text{AI}_{\text{flash}} = 2/q_w$ FLOPs/byte. PKM's throughput advantage comes entirely from loading $260\times$ fewer bytes, not from higher arithmetic intensity.

E.5 Hardware Parameters

Table 16: Cluster hardware parameters used in Section 5.5.

Parameter	Value	Efficiency factor
Peak compute (F_{peak})	4500 TFLOP/s	$\alpha_F = 0.35$
HBM capacity (G_{HBM})	192 GB/GPU	—
HBM bandwidth ($\mathcal{B}_{\text{HBM}}$)	8 TB/s	$\alpha_{\text{HBM}} = 0.80$
NVLink bandwidth ($\mathcal{B}_{\text{NVLink}}$)	900 GB/s	$\alpha_{\text{NVLink}} = 0.70$
GPUs per NVLink domain	72	—

Table 17: Mobile hardware parameters used in Section 5.5.

Parameter	Value	Efficiency factor
Peak compute (F_{peak})	45 TOPS	$\alpha_F = 0.35$
DRAM capacity (G_{DRAM})	12 GB	$\alpha_G^{\text{DRAM}} = 0.35$
DRAM bandwidth ($\mathcal{B}_{\text{DRAM}}$)	96 GB/s	$\alpha_{\text{DRAM}} = 0.70$
Flash bandwidth ($\mathcal{B}_{\text{flash}}$)	4 GB/s	$\alpha_{\text{flash}} = 0.70$

E.6 Model Parameters for the Mobile Inference Simulation

The mobile inference figure uses a fixed active-parameter count of 3.93B: d_{model}=2560, d_{ff}=10240, L=40 layers, PKM layer interval Δ=4 (10 PKM layers, 30 regular dense layers). Attention weights are in DRAM; FFN parameters in flash for PKM and MoE. Three expansion ratios $T/N \in \{4, 8, 16\}$ are evaluated.

Expert count from T/N. Let $\alpha \approx 0.2$ be the attention fraction of active parameters. With top-k=1:

$$E = \frac{T/N - \alpha}{1 - \alpha}, \tag{71}$$

giving $E \in \{5, 10, 20\}$ for $T/N \in \{4, 8, 16\}$.

KV-cache crossover. KV-cache traffic equals FFN flash traffic when $2d_{\text{model}}\, C\, L\, q_{\text{kv}} = 3\, d_{\text{model}}\, d_{\text{ff}}\, L\, q_w$, giving $C = \frac{3\, d_{\text{ff}}\, q_w}{2\, q_{\text{kv}}}$. With q_w=0.5 (int4) and q_{kv}=1 (int8): $C = \frac{3\times 10240\times 0.5}{2\times 1} \approx$ 7.7k tokens.

E.7 Simulation Assumptions

The mobile inference simulation is intentionally conservative toward PKM:

- **MoE**: perfect per-window expert cache, each expert loaded at most once per window W, so flash cost $= \min(E,\, W) \cdot P_e \cdot q_w$ per layer.
- **PKM**: no value caching, every retrieval incurs a full flash read of $k_{\text{total}} \cdot d \cdot q_w$ per PKM layer.

PKM's throughput advantage is therefore a lower bound on the real-device gap.

Communication overlap assumption. The training roofline models GPU-to-GPU communication as sequential with compute. Under chunk-pipelined collectives (e.g. Megatron-LM interleaved dispatch), communication and compute can overlap and the formula reduces to $t_{\text{step}} = \max(t_{\text{compute}},\, t_{\text{HBM}},\, t_{\text{A2A}})$; the sequential assumption is therefore conservative toward MoE.

E.8 Hardware Assumptions for Coupled Resource Scaling

We calibrate the coupled scaling path in Section 5.4 to a configuration benchmarked in the Megatron Core MoE technical report (NVIDIA, 2026): training Qwen3-235B, an MoE model with 235B total and 22B active parameters, on 256 H100 GPUs in BF16 precision, the precision used to train our models. This configuration achieved 320 TFLOP/s per GPU. The report does not state its MFU; relative to the H100 dense BF16 peak of 989.5 TFLOP/s, the measured throughput corresponds to an MFU of $320/989.5 \approx 32.3\%$.

Dividing the total parameter count by the number of GPUs gives $235\text{B}/256 \approx 0.918$ billion parameters per GPU. This is an aggregate ratio, not a single-device storage limit.

The measured throughput corresponds to $320\ \text{TFLOP/s} \times 86{,}400\ \text{s} = 27.648$ EFLOP of compute per GPU-day. With $16n$ GPUs trained for n days, $C(n) = 16n \cdot n \cdot 27.648 \times 10^{18} = 4.424 \times 10^{20} n^2$ FLOP and $T(n) = 16n \cdot 0.918 \times 10^9 = 14.69n \times 10^9$ parameters.