EMO: FRUSTRATINGLY EASY PROGRESSIVE TRAINING OF EXTENDABLE MOE

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse Mixture-of-Experts (MoE) models offer a powerful way to scale model size without increasing compute, as per-token FLOPs depend only on k active experts rather than the total pool of E experts. Yet, this asymmetry creates an MoE efficiency paradox in practice: adding more experts balloons memory and communication costs, making actual training inefficient. We argue that this bottleneck arises in part because current MoE training allocates too many experts from the beginning, even though early-stage data may not fully utilize such capacity. Motivated by this, we propose **EMO**, a simple progressive training framework that treats MoE capacity as *expandable memory* and grows the expert pool over the course of training. EMO explicitly models sparsity in scaling law to derive stage-wise compute-optimal token budgets for progressive expansion. Empirical results show that EMO matches the performance of a fixed-expert setup in large-scale experiments while improving wall-clock efficiency. It offers a surprisingly simple yet effective path to scalable MoE training, preserving the benefits of large expert pools while significantly reducing both training time and GPU cost.

1 INTRODUCTION

Sparse Mixture-of-Experts (MoE) architectures decouple model capacity from per-token computation, making it possible to scale parameters without a proportional increase in floating-point operations (FLOPs) (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). This decoupling has powered some of the most capable language models to date, including GLaM (Du et al., 2022), Mixtral (Jiang et al., 2024a), Kimi K2 (Team et al., 2025) and DeepSeek-V4 (DeepSeek-AI, 2026). However, wall-clock time has not been fully decoupled with E. As E grows, all-to-all communication, optimizer-state memory, and small per-expert general matrix multiplications (GEMMs) all grow with it (Lepikhin et al., 2021; Gale et al., 2023; Rajbhandari et al., 2022; Yan et al., 2026), so realized step time tracks E even when FLOPs do not. Figure 1 quantifies this gap on our setup: with fixed 4B activated parameters, increasing E from 8 to 128 leaves theoretical FLOPs unchanged but inflates wall-clock step time by $1.72\times$, a gap that widens further as activated size scales up. Thus, the theoretical efficiency of MoE is largely a FLOPs-level promise, its realized training cost still grow substantially with the total number of experts. This raises a question that remains under-appreicated in the pretraining setting: does training actually need all E experts from the start?

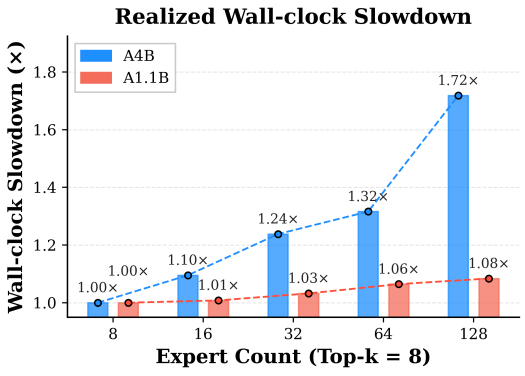


Figure 1: Increasing expert count E with fixed *top-k* activated experts substantially slows down training, especially at larger scales. *A4B* denotes 4B activated parameters (out of 36B total at E=128); *A1.1B* denotes 1.1B activated parameters (out of 9.6B at E=128). All experiments are conducted on 4 nodes of 8×H200 GPUs.

A useful lens is to view each expert as a unit of *parametric memory*: a specialized subnetwork that stores and retrieves patterns relevant to a subset of the data (Shazeer et al., 2017; Roller et al., 2021; Bengio et al., 2013). When training data is limited, a large expert pool is both unnecessary

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.

and inefficient—it amplifies communication costs and memory, and provides capacity that cannot yet be utilized; as data scales, additional experts (memory) become beneficial. Recent MoE scaling laws are consistent with this view: the compute-optimal expert count grows with the token budget (Ludziejewski et al., 2025; Clark et al., 2022). This suggests a simple principle:

MoE capacity should grow progressively with data as an expandable memory.

Building on this principle, we propose **EMO** (**E**xtendable **M**ixture-**o**f-Experts), a progressive training framework that incrementally expands the expert pool during training. Our approach starts from a small (or dense) model and upcycles it into larger MoE models through multiple expansion stages, enabling efficient utilization of both compute and data. Prior work on routing and load balancing (Fedus et al., 2022; Clark et al., 2022; Zhou et al., 2022), systems-level optimization (Lepikhin et al., 2021; Gale et al., 2023; Rajbhandari et al., 2022; Yan et al., 2026; Zhang et al., 2025; Jiang et al., 2024b), and sparse upcycling (Komatsuzaki et al., 2023; Nakamura et al., 2025; He et al., 2024) all fix E throughout training; EMO instead lets E grow with the data, not relying on architectural manipulations beyond standard MoE layers or auxiliary objectives.

However, the effectiveness of EMO hinges on when to expand. If training remains at small E for too long, the model may underutilize the data needed by its later expanded capacity; if expansion occurs too early, training incurs the full large-E wall-clock cost for most of the run. To address this question, we study *optimal stage-wise token allocation* through fitting a sparsity scaling law that explicitly model the effect of expert count E. Concretely, we run a series of scaling-law experiments, calibrated on a sweep across $E \in \{2, \ldots, 256\}$ on our data and architecture. Under this, we predict the performance of each stage and assign precisely the data every expert capacity justifies. The resulting schedule front-loads cheap small-E training and reserves large-E for the end.

We validate this at scale by progressively expanding a 1.1B dense model into a 9.6B-A1.1B MoE with E=128 over five stages on 1.92T tokens. EMO reaches a final pretraining loss of 1.017, compared to 0.994 for the fixed-E=128 baseline (a 2.3% relative gap) and 1.065 for the fixed-E=32 baseline, while saving 10% GPU hours. Downstream evaluation on eight benchmarks spanning reasoning, knowledge, and commonsense shows the same: EMO clearly outperforms the fixed-E=32 baseline and remains comparable to the fixed-E=128 ceiling.

2 Preliminary

2.1 Mixture of Experts

MoE models extend standard transformer architectures by replacing the feed-forward network (FFN) with E experts and a routing function that sparsely selects a subset for each token (Shazeer et al., 2017; Fedus et al., 2022). Given token representation x, the router assigns probabilities over experts and activates only the top-k:

$$\text{MoE}(x) = \sum_{i \in \mathcal{K}(x)} p_i(x) \cdot E_i(x),$$

where $\mathcal{K}(x) \subset \{1, \ldots, E\}$ with $|\mathcal{K}(x)| = k$ denotes the selected expert indices and $p_i(x)$ are the normalized routing weights. Since $k \ll E$ in practice, per-token floating-point operations (FLOPs) scale with k, enabling parameter count to grow quasi-independently of computation.

2.2 The MoE Efficiency Paradox

Unlike dense transformers, where every parameter participates in every forward pass and FLOPs tracks training wall-clock closely, MoE breaks this correspondence. Per-step FLOPs follow:

$$\text{FLOPs}_{\text{step}} \approx 6 \cdot B \cdot N_{\text{act}},$$

where B is batch size and the activated parameter count N_{act} scales with the number of active experts k rather than the total expert pool E. Wall-clock time, however, grows with E through three system-level overheads that scale with the expert pool: all-to-all communication from expert parallelism (Lepikhin et al., 2021; Rajbhandari et al., 2022), aggregate memory for parameters,

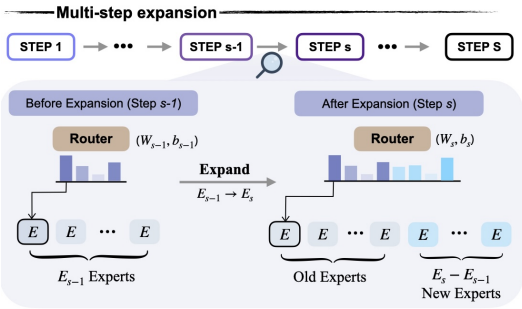


Figure 2: Overview of EMO. EMO performs multi-step expansions; each step we increase model's total expert number with appropriate initialization for new experts and routers.

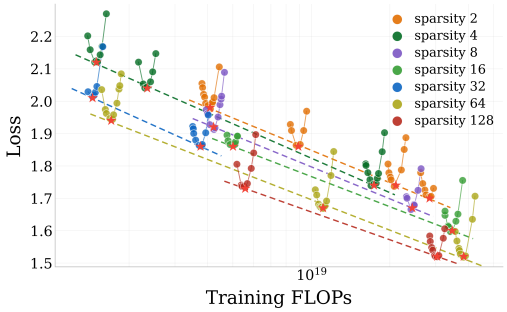


Figure 3: Sparsity scaling law. Optimal data–compute changes with E, motivating a sparsity-aware token allocation. We leverage this to derive per-stage optimal token allocations.

gradients, and optimizer states (Fedus et al., 2022), and GPU underutilization from small per-expert GEMMs and routing kernels (Gale et al., 2023; Yan et al., 2026).

Figure 1 illustrates this gap: at $N_{\text{act}} = 1.1$B, increasing E from 8 to 128 inflates wall-clock step time by 1.08×; at $N_{\text{act}} = 4$B, the same expansion costs 1.72×, and the gap is expected to widen further at larger activated sizes. We refer to this FLOPs–wall-clock mismatch as the *MoE efficiency paradox*.

2.3 Upcycling

Rather than training MoE from scratch, *sparse upcycling* (Komatsuzaki et al., 2023) initializes an MoE from a pre-trained dense checkpoint by replicating the FFN weights across experts:

$$E_1(x) := E_2(x) \ldots := E_E(x) := FFN(x)$$

where E_i denotes the i-th expert and E is the total number of experts. This initialization preserves the behavior of the original dense model when experts are identical and routing is uniform. However, single-step upcycling can be *aggresive* when experts number is large and is not sufficiently efficient.

3 EMO: Extendable Mixture of Experts

To smoothly and efficiently move from small MoE or dense model to large MoE model, EMO introduces a multi-step expansion strategy of expert number E with a fixed activated expert number k. The algorithm is described in Algorithm 1. At each stage, the model's expert pool grows from E_s to E_{s+1} experts as in Figure 2, and training continues on additional data with the enlarged capacity. The framework requires two major design decisions: (1) *when* to expand in each stage, to optimize the balance between performance and costs (§3.1); (2) *how* to initialize the expansion (§3.3).

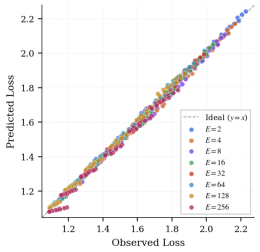
Algorithm 1 EMO: Progressive MoE Training

Input: Initial model $\theta^{(0)}$ with E_0 experts, total token budget D, final stage expert E_S
Input: Expansion schedule $\{(E_s, d_s)\}_{s=1}^{S}$ ▷ Sec. 3.1: compute optimal token counts
Output: Trained model $\theta^{(S)}$ with E_S experts

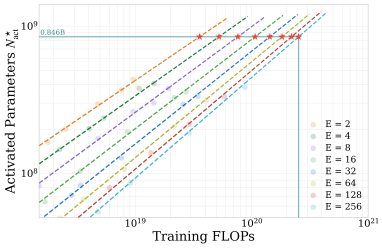
1: **for** $s = 1$ to S **do**
2: $\quad\theta^{(s-1)} \leftarrow \text{EXPAND}(\theta^{(s-1)},\ E_{s-1} \to E_s)$ ▷ Sec. 3.3
3: $\quad\theta^{(s)} \leftarrow \text{TRAIN}(\theta^{(s-1)},\ d_s \text{ tokens})$
4: **end for**
5: **return** $\theta^{(S)}$

3.1 Expert-Aware Token Allocation

Given a total budget of D tokens and expansion schedule $E_1 < E_2 < \cdots < E_S$, how should tokens be distributed across stages? Training too long at small E_s wastes model capacity; expanding too



(a) Step 1: Scaling-law fit.



(b) Step 2: Estimate D_s^* across E.

Stage	Sparsity	E	Cum. Tokens (B)	Stage Budget
1	0%	8	4.85	23.54%
2	50%	16	6.80	9.47%
3	75%	32	10.20	16.50%
4	87.5%	64	14.70	21.84%
5	93.75%	128	20.60	28.64%

(c) Step 3: Compute stage-wise budget d_s^* in our schedule.

Figure 4: Stage-wise, expert-aware token allocation. We study how to optimally allocate tokens in progressive training given fixed activated parameters and token budget. As sparsity-aware scaling law makes progressive training *predictable*, we estimate cumulative per-expert optimal token allocations first, then normalize them into our exapansion schedule with total token budget.

early wastes wall-clock time. We derive a principled allocation by fitting a scaling law that explicitly models the effect of expert count E on data efficiency, then validate it empirically.

Step 1: Fit the scaling law. To allocate tokens across expansion stages, we need a loss model that captures how data efficiency changes with expert count E. Standard compute-optimal scaling laws (Hoffmann et al., 2022) model loss as $L(N, D)$, treating all models with the same activated size identically – regardless of the total number of experts E:

$$L(N, D) = aN^{-\alpha} + bD^{-\beta} + c,$$

where a, b are scale coefficients, α, β are scaling exponents, c is the irreducible entropy floor. This is insufficient for EMO, where the same N_{act} serves different expert configurations at different stages. To address this issue, we leverage unified MoE scaling law Ludziejewski et al. (2025) that jointly models the dependence on activated parameters, total experts, and dataset size:

$$L(N_{\text{act}}, E, D) = m(E)N_{\text{act}}^{\mu(E)} + n(E)D^{\nu(E)} + c, \tag{1}$$

where the coefficients and exponents are explicitly parameterized as functions of E:

$$m(E) = aE^{\delta}, \quad n(E) = bE^{\omega}, \quad \mu(E) = \alpha + \gamma \ln E, \quad \nu(E) = \beta + \zeta \ln E$$

The exponent $\nu(E)$ controls how quickly loss decreases with additional tokens, indicating the optimal data-to-parameter ratio shifts as experts are added.

With the same data as our main experiments, we fit Eq. (1) on a grid of small-scale runs varying $E \in \{1, 2, 4, ..., 256\}$, N_{act} and D. Figure 4a shows predicted vs. observed loss across all configurations, validating the effectiveness of step 1[1]. The fitted parameters are reported in Appendix C. Figure 3 visualizes the sparsity scaling law on our dataset.

Step 2: Compute per-expert optima. As N_{act} is fixed throughout training, loss depends only on E_s and cumulative tokens D_s at each stage s. We determine the optimal token allocation across stages using the fitted scaling law $L(N_{\text{act}}, E, D)$ and compute-optimal training principles. At each stage s with expert count E_s, we solve for the *compute-optimal token count* under fixed compute F:

$$D_s^* = \arg\min_D L(N_{\text{act}}, E_s, D) \quad \text{s.t.} \quad F = 6N_{\text{act}}D. \tag{2}$$

D_s^* is the optimal *cumulative* token count for training entirely with E_s experts (Figure 4b).

Step 3: Normalize into a schedule. We convert cumulative optima into incremental allocations:

$$d_s^* = D_s^* - D_{s-1}^*, \qquad D_0^* = 0. \tag{3}$$

Given a total token budget D_{total}, we normalize to obtain the final per-stage allocation:

$$d_s = D_{\text{total}} \cdot \frac{d_s^*}{\sum_{s'=1}^{S} d_{s'}^*}. \tag{4}$$

[1]The training fit has $R^2 = 0.9957$ and RMSE 0.0085; held-out RMSE is 0.0092.

Intuitively, D_s^* captures the data requirement of a model with E_s experts under optimal scaling, and the differences d_s^* reflect how much *additional* data is justified when expanding capacity from E_{s-1} to E_s. Normalization preserves these proportions while respecting the total token budget.

The fitted law uses fixed-expert data requirements to guide token allocation without explicitly modeling expansion transients. In our main experiment, EMO doubles the expert pool from $E_1 = k = 8$ to $E_S = 128$, with the resulting schedule shown in Figure 4c.

3.2 When to Expand: Validating the Allocation.

The scaling law tells us *how many* tokens each stage deserves, but does the predicted allocation actually sit in a favorable region of the quality–cost landscape? We test this by growing an $E = 16$ model to $E = 32$ at three fixed fractions of the total token budget (25%, 50%, 75%), bracketing the scaling-law-derived expansion point at approximately 45%.

The scaling law targets the right region. Figure 5 shows the final losses of all three expansions fall between the fixed $E = 16$ and fixed $E = 32$ baselines. Expanding at 25% achieves the lowest loss (1.069), while expanding at 50% and 75% reach 1.071 and 1.076 respectively—each step later in timing costs quality but saves wall-clock time. The loss difference between 25% and 50% expansion is only 0.002, while the gap widens to 0.005 between 50% and 75%. This indicates that the quality–cost curve is relatively flat in the 25–50% region and steepens beyond 50%. Our scaling-law-derived allocation at ∼45% falls squarely in this flat region, capturing most of the quality benefit of early expansion while avoiding the full wall-clock cost of the earliest (25%) schedule. Appendix E.3 quantifies this trade-off: the 50% schedule has the lowest loss-penalty-to-time-saving ratio among the tested timings. Figure 6 confirms the same pattern on downstream benchmarks.

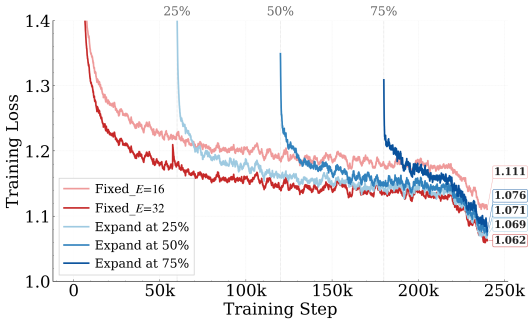


Figure 5: Validating Token Allocation: increasing experts $E = 16 \rightarrow 32$ @25%, 50% and 75%.

Takeaway 1. *The quality–cost trade-off is favorable near the scaling-law-predicted expansion point: expanding later than d_s^* loses quality rapidly, while expanding much earlier diminishes returns.*



Figure 6: Downstream curves across different expansion timing ($E = 16 \rightarrow 32$). Comparing to FIXED_E=32, EMO@25% outperforms on both MMLU and GSM8K, performs comparably on HellaSwag and ARC-E. Even EMO@75% performs much better than FIXED_E=16.

3.3 Expert Expansion and Initialization

Consider an expansion step s that increases the expert number from E_{s-1} to E_s, three components must be initialized: the *new expert weights*, the *router weights*, and the *router bias*. Let the model parameters at step $s-1$ be:

$$\theta_{s-1} = \{\theta_i^{\text{old}}\}_{i=1}^{E_{s-1}}, \quad W_{s-1} \in \mathbb{R}^{E_{s-1} \times d_r}, \quad b_{s-1} \in \mathbb{R}^{E_{s-1}}.$$

After expansion, the parameters become:

$$\theta_s = \{\theta_i^{\text{old}}\}_{i=1}^{E_{s-1}} \cup \{\theta_j^{\text{new}}\}_{j=E_{s-1}+1}^{E_s}, W_s = \begin{bmatrix} W_{s-1} \\ W_{\text{new}} \end{bmatrix}, \quad b_s = \begin{bmatrix} b_{s-1} \\ b_{\text{new}} \end{bmatrix},$$

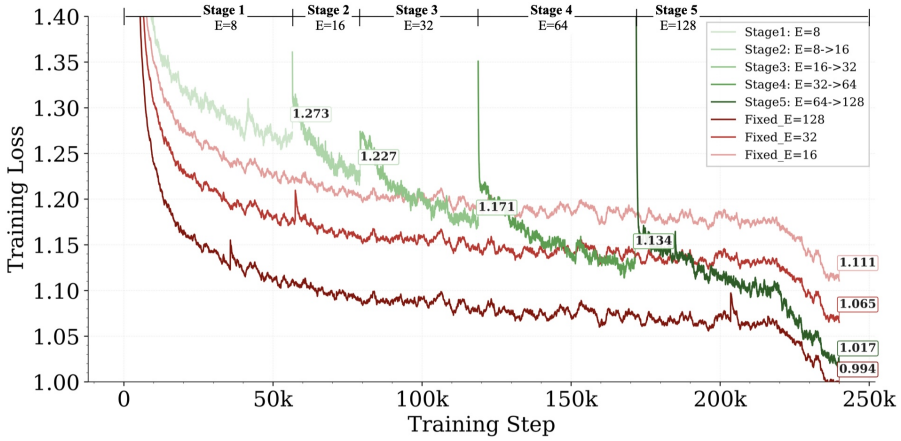


Figure 7: Training-loss comparisons under fixed FLOPs. EMO starts from $E = 8$ and progressively expands to $E = 128$. EMO reaches a comparable loss as FIXED_E=128 baseline while being more efficient in training and GPU memory. EMO greatly outperforms FIXED_E=32 and FIXED_E=16.

where $W_{\text{new}} \in \mathbb{R}^{(E_s - E_{s-1}) \times d_r}$ and $b_{\text{new}} \in \mathbb{R}^{E_s - E_{s-1}}$ correspond to the newly added experts.

Figure 2 illustrates how the expansion works. We ablate initialization strategies in §5 and find that EMO is robust across choices: all configurations converge to similar final loss, with the main difference being the size of the transient spike at expansion. For elegance, in all main experiments, we use Gaussian initialization for new experts and router weights, and reset all router biases to zero. Optimizer states are reset at each expansion with a short learning rate warmup.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Architecture and training. We train decoder-only Transformer MoE models with 1.1B activated parameters (including embeddings) and fixed top-$k = 8$ routing. EMO expands through five expert counts, $8 \to 16 \to 32 \to 64 \to 128$, following the scaling-law allocation in Figure 4c. We use a warm-stable-decay (WSD) (Hu et al., 2024) learning-rate schedule with a 2K-step initial warmup and a 500-step warmup after each expansion. After warmup, the learning rate remains constant and decays only during the final 10% of training. Appendix B provides architecture specifications, per-stage token budgets, optimizer hyperparameters, and hardware configuration.

Baselines We compare EMO with three fixed-expert baselines, $E \in \{16, 32, 128\}$. All models are trained from random initialization with the same backbone architecture, optimizer hyperparameters, peak learning rate, final decay schedule, and load-balancing mechanisms.

Data. All models are pretrained on 1.92T tokens from a shared mixture of web, code, mathematical, and multilingual corpora. We evaluate held-out perplexity and downstream tasks every 5K steps; Appendix B.2 provides the data mixture and evaluation protocol.

4.2 MAIN RESULTS

Figure 7 compares EMO with fixed-expert MoE baselines trained under the same total token budget and activated parameter count. EMO starts from a much smaller expert number ($E = 8$ at the beginning), but reaches the same low-loss regime as the FIXED_$E = 128$ baseline by the end of training, with a final loss of 1.017 versus 0.994. At the same time, EMO clearly outperforms smaller fixed-expert baselines such as FIXED_$E = 16$ and FIXED_$E = 32$, showing that progressive expansion avoids the capacity bottleneck of small expert pools while avoiding the cost of using the largest expert pool throughout training. Each expansion introduces a transient loss spike, but the loss recovers within roughly 10K steps, suggesting that the newly added experts are integrated quickly rather than causing persistent optimization instability. Together, these results show that EMO

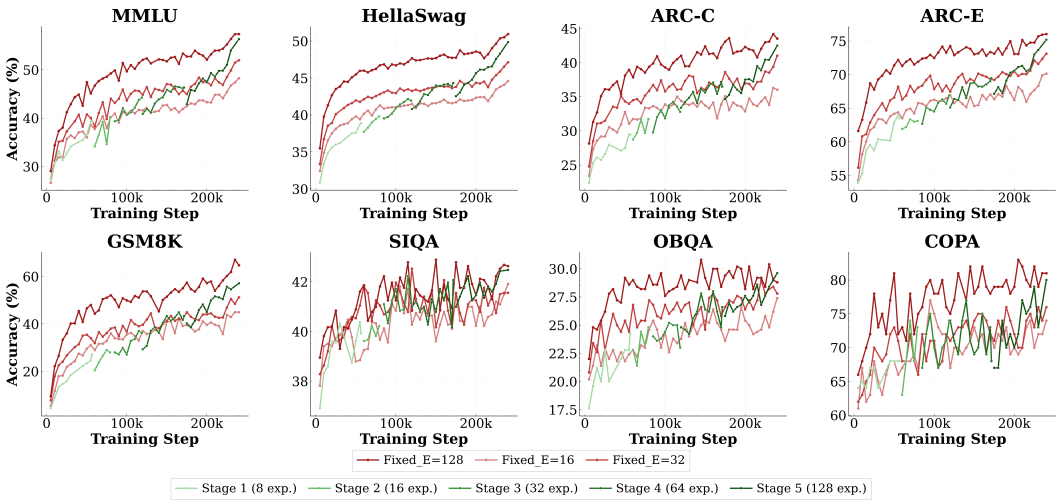


Figure 8: Benchmark curves during training. We evaluate EMO and fixed-expert baselines on eight downstream benchmarks. EMO is competitive with or stronger than FIXED_E=128. Meanwhile, EMO generally exceeds FIXED_E=32 and FIXED_E=16 in downstream tasks.

provides a stable and efficient training trajectory: smaller expert pools save training time in stages 1–4, while stage 5 recovers the benefits of the full $E = 128$ configuration.

Figure 8 reports accuracy on multiple benchmarks (See full downstream results and validation perpexity in Appendix D). After the final expansion, EMO is competitive with the fixed $E = 128$ baseline on most benchmarks and clearly improves over smaller fixed-expert baselines. On GSM8K, the fixed $E = 128$ baseline remains stronger at the final checkpoint, suggesting that some reasoning-heavy benchmarks may benefit more from exposing the large expert pool throughout training. Overall, these results show that treating MoE capacity as expandable parametric memory improves the quality–cost trade-off of MoE training: EMO retains much of the benefit of large expert pools while substantially reducing the cost of reaching them.

Takeaway 2. *EMO shows that large-expert MoE performance does not require large-expert MoE cost from the first step, and progressive expansion recovers most of the final capacity benefit while significantly reducing early-stage training overhead.*

5 Analysis

5.1 Expansion Initialization

We first study how to initialize newly introduced experts and router parameters when expanding the expert pool. Figure 9 compares three strategies for the $16 \rightarrow 32$ expansion:

- *Gaussian init*: randomly initialize the new experts and router parameters.
- *Gaussian init with router bias=0*: randomly initialize new experts and router weights, while resetting router biases to zero to clear prior load-balancing state.
- *Copy from old ckpts*: initialize new experts and router parameters from existing checkpoints.

For reference, the gray curve shows the FIXED_E= 16 baseline without expansion. All three expansion strategies substantially outperform the non-expanded $E = 16$ baseline, indicating that the benefit of EMO is not tied to a single fragile initialization choice. The strategies differ only marginally in final training loss given a sufficient token budget. At the expansion boundary, all methods exhibit a transient loss spike, but the spike disappears quickly during warmup and training remains stable, suggesting that new experts and routers learn fast even when starting from scratch.

In particular, *Copy from old ckpts* starts with the highest spike but stabilizes fastest. The bias-reset variant produces a larger spike than plain Gaussian init because resetting the router bias discards the load-balancing behavior learned in the previous stage and abruptly changes token assignments; both

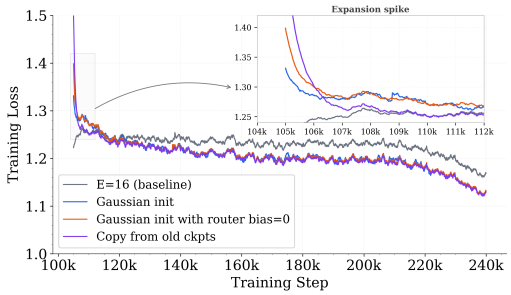


Figure 9: Expert and router initialization. EMO is robust to initialization: all configurations converge to similar final loss, with differences mainly in the transient spike.

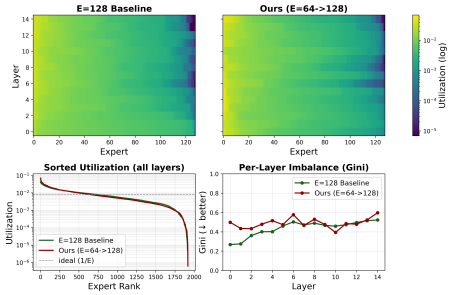


Figure 10: Expert utilization on validation data. Top: per-layer, per-expert loads; bottom-left: utilization across all layers; bottom-right: per-layer Gini (0 = uniform, 1 = collapsed).

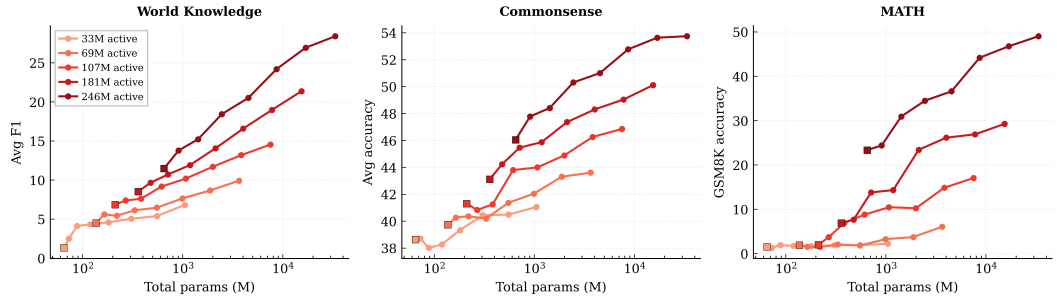


Figure 11: MoE as expandable memory. We evaluate parts of our scaling law MoE models on several world knowledge benchmarks (e.g.,TriviaQA (Joshi et al., 2017), NQ (Kwiatkowski et al., 2019)).We evaluate multiple commonsense benchmarks including HellaSwag (Zellers et al., 2019), WinoGrande Sakaguchi et al. (2021) etc.; math is evalated on GSM-8K Cobbe et al. (2021).

nonetheless converge to similar loss after warmup. As our token budget far exceeds the spike regime, we use the bias-reset variant in our main experiments for simplicity.

When expanding the expert pool, optimizer states from the previous stage can either be carried over together with the model parameters or reset for the new stage. We find the improvement from carrying them over is marginal: differences vanish after roughly 500 warmup steps. We reset optimizer states at each expansion to avoid dimension mismatches when adding new experts.

Takeaway 3. *EMO is robust to both initialization and optimizer-state handling at expansion boundaries: expert learning is fast, and the choice mainly affects the size of the transient spike.*

5.2 Expert utilization comparison

We measure routing balance on 5K validation sequences by collecting tokens per expert from every MoE router. Both FIXED_$E = 128$ and our progressively expanded model exhibit similar distributions (Figure 10 top): experts in middle and last few layers receive relatively larger load. Quantitatively, the Gini coefficient is 0.44 for the baseline and 0.50 for our expanded model, a $\sim$14% relative gap that is concentrated in the middle layers (Figure 10 bottom right); the expanded model therefore has somewhat greater load inequality by this metric. The expert utilization curves (Figure 10 bottom left) overlap closely. The new-to-old load ratios of 0.9958–1.0081 further show comparable aggregate use of both expert groups (Appendix E.1). Progressive expansion thus achieves similar routing behavior to the baseline, despite the new experts being trained for shorter time. We also assess representation diversity using the representation-collapse (RC) metric: EMO shows less collapse than the corresponding fixed-expert baselines at $E = 16, 32, 128$ (Appendix E.2).

5.3 MoE AS EXPANDABLE MEMORY

EMO is motivated by viewing MoE as expandable memory: increasing the expert pool enlarges the number of addressable parameters, improving the model's ability to store and retrieve knowledge. We test this in Figure 11 by varying E from 2 to 256 under different activated-parameter budgets. On memory-intensive tasks such as TriviaQA, larger expert pools consistently improve performance, supporting the role of experts as additional parametric memory. Similar trends appear on commonsense reasoning tasks, while GSM8K benefits more clearly only when total parameter count is sufficiently large. This also explains why the fixed $E = 128$ baseline remains stronger on GSM8K: exposing the full expert pool throughout training may give reasoning skills more time to organize across experts. These results support as strong foundations and motivations for EMO: if experts function as addressable parametric memory, then it is not necessary to expose the full memory capacity from the beginning of training. Instead, EMO gradually expands this memory as training progresses, reducing early-stage overhead while preserving the benefits of large expert pools.

6 RELATED WORK

Sparse MoE models and routing. Mixture-of-Experts models increase parameter capacity by routing each token to a sparse subset of experts (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). Large-scale MoE language models such as GLaM, Mixtral, DeepSeekMoE, DeepSeek-V2, and OLMoE demonstrate that sparse activation can scale model capacity efficiently (Du et al., 2022; Jiang et al., 2024a; Dai et al., 2024; Liu et al., 2024; Muennighoff et al., 2025; Shi et al., 2024a). A large body of work improves routing behavior and expert utilization, including load-balancing losses, expert-choice routing, stable transfer designs, and auxiliary-loss-free balancing (Fedus et al., 2022; Zhou et al., 2022; Zoph et al., 2022; Wang et al., 2024). These methods improve routing within a fixed expert pool; EMO addresses when that pool should grow.

MoE upcycling. Sparse upcycling converts a pretrained dense model into an MoE by replacing dense FFN layers with expert layers and copying dense weights into multiple experts (Komatsuzaki et al., 2023), reducing the cost of training sparse models from scratch. Recent work improves dense-to-MoE conversion through expert re-initialization (Nakamura et al., 2025), virtual-group initialization and weight scaling for fine-grained MoEs (He et al., 2024), domain-specialized branches (Sukhbaatar et al., 2024), and instruction-tuning upcycling (Hui et al., 2025). These methods primarily focus on constructing a fixed MoE, whereas EMO is complementary: it repeatedly expands the expert pool during pretraining and allocates data across expansion stages.

Systems optimizations for MoE training. Expert-parallel MoE training requires all-to-all communication to dispatch and combine tokens across devices (Lepikhin et al., 2021; Rajbhandari et al., 2022). Systems work reduces this overhead through optimized dispatchers, communication–computation overlap, grouped GEMM, block-sparse computation, kernel fusion, and memory optimizations (Gale et al., 2023; Zhao et al., 2025; DeepSeek-AI, 2024; Yan et al., 2026; Korthikanti et al., 2023). Early systems such as FastMoE and Tutel made distributed MoE training practical (He et al., 2021; Hwang et al., 2023), while MegaBlocks and Megatron-Core MoE further improve sparse computation and routing efficiency (Gale et al., 2023; Yan et al., 2026). These methods lower the cost of a fixed MoE configuration, whereas EMO is orthogonal: it reduces training cost by delaying expensive large-expert configurations until later stages.

7 CONCLUSION

We introduced EMO, a simple progressive training framework for Extendable Mixture-of-Experts models. Motivated by the MoE efficiency paradox, EMO avoids committing to a large expert pool from the start and instead grows expert capacity during training. With a sparsity-aware token allocation strategy, EMO achieves a favorable quality–cost trade-off, approaching fixed large-expert baselines while using cheaper intermediate configurations. It is also robust to expansion initialization choices and maintains broad expert utilization after expansion. One limitation is that the scaling-law fit does not explicitly account for learning rate, batch size, or optimizer settings. Scaling EMO to larger activated and total parameter regimes is an important direction for future work.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting and editing, literature review, formatting, and the interpretation and presentation of existing results. The authors are responsible for verifying all AI-assisted content and take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the EMO algorithm, expert expansion procedure, and scaling-law-based token allocation in Sections 3.1–3.3. Our experimental setup, including model architecture, training schedule, baselines, data composition, and evaluation protocol, is documented in Section 4 and Appendix B, with scaling-law fitting details and additional results provided in Appendices C–E. We will release all code, training configurations, and model checkpoints upon acceptance.

REFERENCES

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, Heyan Huang, and Furu Wei. On the representation collapse of sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2204.09179.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, 2024.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

DeepSeek-AI. Deepseek-v4: A million-token context that agents can actually use. https://huggingface.co/blog/deepseekv4, 2026.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

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

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

Ethan He, Abhinav Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. Upcycling large language models into mixture of experts. *arXiv preprint arXiv:2410.07524*, 2024.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. FastMoE: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*, 2021.

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. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 30016–30030, 2022.

Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling*, 2024.

Tingfeng Hui, Zhenyu Zhang, Shuohuan Wang, Yu Sun, Hua Wu, and Sen Su. Upcycling instruction tuning from dense to mixture-of-experts via parameter merging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13012–13031, 2025.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. In *Proceedings of Machine Learning and Systems*, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024a.

Chenyu Jiang, Ye Tian, Zhen Jia, Shuai Zheng, Chuan Wu, and Yida Wang. Lancet: Accelerating mixture-of-experts training via whole graph computation-communication overlapping. *Proceedings of Machine Learning and Systems*, 6:74–86, 2024b.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1601–1611, 2017.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *International Conference on Learning Representations*, 2023.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 5:341–353, 2023.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Jan Ludziejewski, Maciej Pióro, Jakub Krajewski, Maciej Stefaniak, Michał Krutul, Jan Małaśnicki, Marek Cygan, Piotr Sankowski, Kamil Adamczewski, Piotr Miłoś, et al. Joint moe scaling laws: Mixture of experts can be memory efficient. In *International Conference on Machine Learning*, pp. 41056–41073. PMLR, 2025.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. In *International Conference on Learning Representations*, 2025.

Taishi Nakamura, Takuya Akiba, Kazuki Fujii, Yusuke Oda, Rio Yokota, and Jun Suzuki. Drop-upcycling: Training sparse mixture of experts with partial re-initialization. In *International Conference on Learning Representations*, 2025.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *advances in neural information processing systems*, 34:17555–17566, 2021.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685, 2024.

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. In *International Conference on Learning Representations*, 2017.

Chufan Shi, Cheng Yang, Xinyu Zhu, Jiahao Wang, Taiqiang Wu, Siheng Li, Deng Cai, Yujiu Yang, and Yu Meng. Unchosen experts can contribute too: Unleashing moe models' power by self-contrast. *Advances in Neural Information Processing Systems*, 37:136897–136921, 2024a.

Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, and Wai Lam. A thorough examination of decoding methods in the era of llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8601–8629, 2024b.

Sainbayar Sukhbaatar, Naman Goyal, Xian Li, Mike Lewis, Angela Fan, Sergey Edunov, and Omer Levy. Branch-train-MiX: Mixing expert LLMs into a mixture-of-experts LLM. *arXiv preprint arXiv:2403.07816*, 2024.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

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

Zijie Yan, Hongxiao Bai, Xin Yao, Dennis Liu, Tong Liu, Hongbin Liu, Pingtian Li, Evan Wu, Shiqing Fan, Li Tao, et al. Scalable training of mixture-of-experts models with megatron core. *arXiv preprint arXiv:2603.07685*, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4791–4800, 2019.

Shulai Zhang, Ningxin Zheng, Haibin Lin, Ziheng Jiang, Wenlei Bao, Chengquan Jiang, Qi Hou, Weihao Cui, Size Zheng, Li-Wen Chang, et al. Comet: Fine-grained computation-communication overlapping for mixture-of-experts. *Proceedings of Machine Learning and Systems*, 7, 2025.

Chenggang Zhao, Shangyan Zhou, Liyue Zhang, Chengqi Deng, Zhean Xu, Yuxuan Liu, Kuai Yu, Jiashi Li, and Liang Zhao. Deepep: an efficient expert-parallel communication library. *DeepSeek-AI, Hangzhou, China, Tech. Rep*, 2025.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A PRELIMINARIES

A.1 NOTATION

To aid readability, we provide a list of key symbols used throughout this paper.

Symbol	Description
N	Total number of model parameters
N_{act}	Active number of model parameters
L	Pretraining Loss (Categorical Cross-Entropy)
F	Training compute budget (in FLOPs)
N^*_{act}	Optimal active number of parameters
E	Expansion factor (number of experts per MoE layer)
K	Number of selected experts per token
D	Dataset size (number of training tokens)
D^*	Compute-optimal training token size
d^*_s	incremental training token size at expansion stage s
S	Number of total expansion stages
$\alpha, \beta, \gamma, \zeta, \delta, a, b, c, m, n$	Coefficients in the parametric scaling law equation

A.2 MOE SCALING LAWS

Hoffmann et al. (2022) adapts scaling laws to MoE by expressing loss as a function of activated model size N_{act} and dataset size D:

$$L(N_{\text{act}}, D) = mN_{\text{act}}^{\mu} + nD^{\nu} + c,$$

Clark et al. (2022) studies scaling under fixed datasets while varying both model size and expert count:

$$L(N_{\text{act}}, E) = a\hat{E}^{\delta} N_{\text{act}}^{\alpha+\gamma \ln E},$$

Then, combining with the joint scaling law Ludziejewski et al. (2025), we models the loss that depend on activated parameters, total experts, and dataset size:

$$L(N_{\text{act}}, E, D) = m(E)N_{\text{act}}^{\mu(E)} + n(E)D^{\nu(E)} + c,$$

B DETAILED EXPERIMENT SETUP

B.1 MAIN EXPERIMENT SETUP

All main experiments use decoder-only Transformer MoE models with the same backbone architecture, optimizer hyperparameters, peak learning rate, final decay schedule, and data mixture. The model has 16 Transformer layers, hidden dimension 2048, 16 query heads, 4 key-value heads, SwiGLU feed-forward blocks, RMSNorm, RoPE with base 100000, and one dense layer before the MoE layers. Each MoE layer uses one shared expert and a routed expert pool with top-$k = 8$ activated routed experts per token. The routed expert hidden dimension is 768, the router uses a sigmoid score function with router bias enabled, and entropy-based load balancing is applied with coefficient 10^{-4}. The global batch size is 8M tokens, the context length is 8192, and the total training budget is 240K optimizer steps, corresponding to 1.92T tokens. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.05, gradient clipping at 1.0, and peak learning rate 9×10^{-4}. The learning-rate schedule is warm-stable-decay: 2K warmup steps, a stable phase, and a linear decay beginning at 90% of the total token budget to a final learning-rate ratio of 0.01. Each expansion resumes from the previous checkpoint, initializes new experts and router rows with Gaussian weights, resets optimizer states and zeros router biases, and applies a 500-step warmup.

Progressive schedule. EMO starts from random initialization at $E = k = 8$, with all routed experts initially active and no external pretrained checkpoint. It doubles the expert pool through $8 \to 16 \to 32 \to 64 \to 128$. The stage fractions are 23.5%, 9.5%, 16.5%, 21.8%, and 28.6%, respectively (Figure 4c); each stage consumes its allocated token budget d_s.

Baselines. The fixed-expert models, denoted as FIXED_16, FIXED_32, and FIXED_128, are randomly initialized and trained from scratch with fixed $E \in \{16, 32, 128\}$ and the same total token budget as EMO. All runs use the same entropy-based load-balancing loss and router-bias mechanism. Fixed$_{E=128}$ tests the objective of approaching full-capacity training at lower realized cost, while Fixed$_{E=16,32}$ tests the benefit of expanding beyond smaller pools. Appendix E.4 additionally presents an extrapolated one-shot comparison.

Infrastructure. Training is performed on 32 NVIDIA H200 GPUs with data parallelism. To evaluate EMO under a highly optimized training stack, we use BF16 mixed precision, FlashAttention 3 (Shah et al., 2024), fused transformer-block kernels, and selective activation recomputation for efficiency. We also employ multi-source sequence packing based on a best-fit-decreasing heuristic to minimize padding waste, with document masking. For MoE-specific optimization, we use entropy-based load balancing and recompute router activations to reduce activation memory (Korthikanti et al., 2023). All evaluated models are base models without instruction tuning; we therefore use greedy decoding (temperature $= 0.0$) across all evaluation tasks (Shi et al., 2024b).

B.2 TRAINING DATA

The pretraining mixture contains math, code, web, multilingual Arabic, QA, and other general text sources with mixture weights 30.1%, 16.49%, 15.48%, 5.0%, 7.97%, and 24.96%, respectively. We use best-fit sequence packing with document masks to reduce padding waste. Validation perplexity is evaluated every 5K steps on held-out web, multilingual, code, academic, and other validation slices. Downstream evaluation is also run every 5K steps on BOOLQ, HELLASWAG, NATURAL QUESTIONS, PIQA, SIQA, WINOGRANDE, OPENBOOKQA, ARC-EASY, ARC-CHALLENGE, RACE, COPA, MMLU, Arabic MMLU, TRUTHFULQA, and GSM8K.

Table 1: Data mixture and evaluation setup for the main experiments.

Item	Setting
Pretraining mixture	Math 30.1%, code 16.49%, web 15.48%, Arabic 5.0%, QA 7.97%, other general text 24.96%
Validation domains	Web, multilingual, code, academic, and other held-out validation slices

C SCALING LAW EXPERIMENTS

C.1 SCALING LAW EXPERIMENT SETUP

The scaling-law experiments use the same tokenizer, data mixture, context length, optimizer family, router design, and validation protocol as the main runs, but sweep smaller activated model sizes and expert counts to make the fit computationally tractable. We train a grid of MoE models with $E \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$. For these scaling runs, top-k is fixed at 2, so changing E changes the total expert pool while keeping sparse activation controlled.

C.2 SCALING LAW FITTING

Table 2: Solutions to Eq. (4)

a	α	δ	γ	b	β	ω	ζ	c	E_{start}	E_{max}
34.89	-0.1805	-0.2276	0.0065	35.01	-0.1775	0.5524	-0.0259	0.3341	1.8095	294.2177

We fit Eq. (1) using the collected validation losses, varying N_{act}, E, and D. Following Hoffmann et al. (2022), the coefficients are optimized with LBFGS under a Huber loss with threshold 0.01, and the final model selection is performed by grid search over initialization and coefficient constraints. Training R^2 is 0.9957, with training and held-out RMSE of 0.0085 and 0.0092, respectively.

Figures 12 and 13 show IsoFLOP curves over tokens and activated parameters.

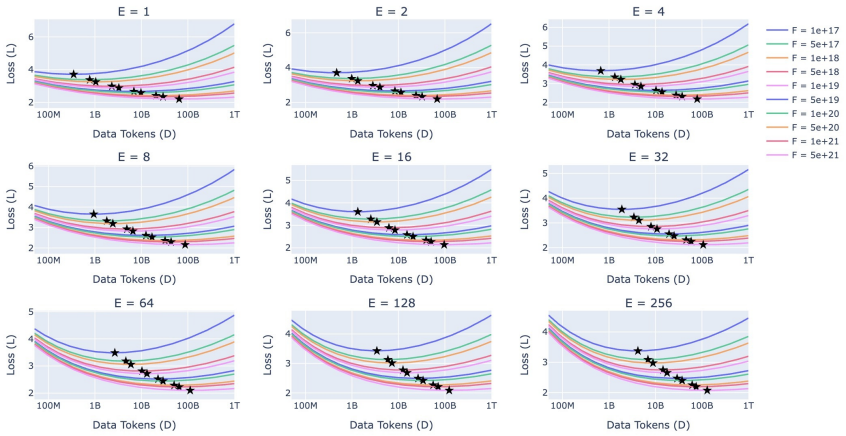


Figure 12: IsoFLOP curves over training tokens; markers indicate compute-optimal points.

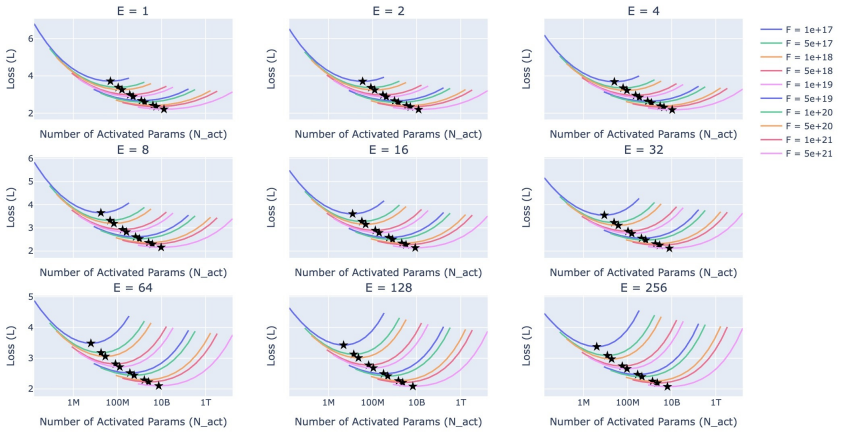


Figure 13: IsoFLOP curves over activated parameters; markers indicate compute-optimal points.

D Additional Analysis

D.1 Downstream Performance

Tables 3 and 4 complement the validation-loss curves with downstream evaluation. The preliminary table compares different $E = 16 \rightarrow 32$ expansion timings, while the core table compares the full multi-stage EMO run against fixed-expert baselines. Across tasks, the downstream results follow the same pattern as the loss curves: progressive expansion is consistently stronger than fixed small-expert baselines and is competitive with the fixed large-expert baseline, especially when accounting for the wall-clock savings from not training with the largest expert pool from the beginning.

Table 3: Downstream performance of expansion timings at the final checkpoint (240k steps).

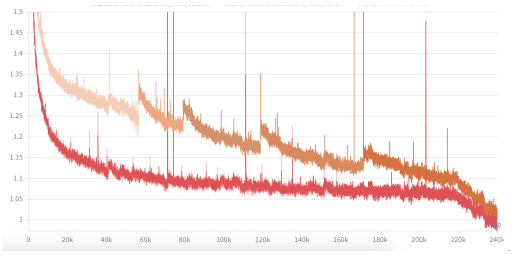
Model	MMLU↑	HellaSwag↑	ARC-C↑	ARC-E↑	PIQA↑	BoolQ↑	GSM8K↑	SIQA↑	OBQA↑	COPA↑	WinoGrande↑	NQ (EM)↑	TriviaQA (EM)↑	RACE-H↑	RACE-M↑
FIXED_$E = 16$	48.27	44.59	36.05	70.15	73.01	59.02	44.96	41.91	27.40	74.00	58.48	9.11	22.58	40.05	**57.17**
FIXED_$E = 32$	**51.98**	**47.14**	**41.03**	**73.11**	73.78	**71.71**	51.25	41.56	27.80	**76.00**	60.46	**11.86**	**27.52**	39.59	55.15
Expand @25%	**52.70**	46.63	36.91	72.69	**73.83**	70.09	52.62	41.76	**29.00**	75.00	60.93	10.78	27.03	**40.57**	55.50
Expand @50%	50.66	46.27	38.03	71.67	73.78	69.42	**53.75**	41.66	27.80	75.00	60.85	10.53	26.49	40.25	55.22
Expand @75%	49.88	46.01	37.51	71.25	**73.83**	66.33	50.34	**42.17**	26.60	73.00	**61.64**	10.06	24.93	39.85	54.60

D.2 Training Loss

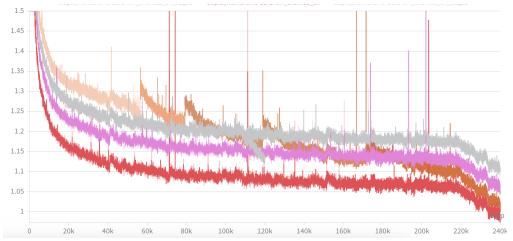
Below we show the un-smoothed training loss:

Table 4: Downstream accuracy (%) at each curve's last checkpoint; column-best values are bold.

Model	MMLU↑	HellaSwag↑	ARC-C↑	ARC-E↑	PIQA↑	BoolQ↑	GSM8K↑	SIQA↑	OBQA↑	COPA↑	WinoGrande↑	RACE-H↑	RACE-M↑
FIXED_E=16	48.27	44.59	36.05	70.15	73.01	59.02	44.96	41.91	27.40	74.00	58.48	40.05	**57.17**
FIXED_E=32	51.98	47.14	41.03	73.11	73.78	71.71	51.25	41.56	27.80	76.00	60.46	39.59	55.15
FIXED_E=128	**57.40**	**50.95**	**43.53**	**76.03**	**76.44**	**74.89**	**64.73**	**42.63**	28.80	**81.00**	**63.06**	40.31	56.20
Stage 1 ($E = 8$)	39.29	38.88	29.44	63.51	69.91	66.85	27.14	39.92	24.20	68.00	56.99	37.36	52.51
Stage 2 (8→16)	40.21	39.81	31.76	63.09	70.13	67.80	27.98	40.23	24.80	73.00	56.20	37.91	51.95
Stage 3 (16→32)	44.27	41.93	32.79	66.89	71.27	69.72	36.16	41.15	23.00	74.00	57.85	38.51	53.20
Stage 4 (32→64)	46.34	44.33	36.48	69.47	73.50	70.52	43.29	42.07	26.60	68.00	58.56	39.71	54.81
Stage5 (64→128)	56.34	49.86	42.49	75.18	74.81	70.40	57.09	42.48	**29.60**	80.00	61.88	**40.39**	55.92



(a) Training Loss (not smoothed). Red is Fixed_E=128, the rest are EMO 5 stages.



(b) Training Loss (not smoothed) with all baselines. Gray is Fixed_E=16, pink is Fixed_E=32.

D.3 Validation PPL

Figure 15 reports validation perplexity for the preliminary $E = 16 \rightarrow 32$ study, which tests the scaling-law allocation by comparing timings at a single expansion boundary. Expanding at 25% or 50% approaches Fixed$_{E=32}$ perplexity in most domains.

EMO and fixed-expert baselines achieve similar held-out perplexity (Figure 16).

E Additional Routing and Expansion Analysis

E.1 Utilization of Newly Added Experts

To examine whether progressively added experts receive comparable traffic to existing experts, Table 5 reports the aggregate new-to-old expert load ratio at each expanded capacity. The ratios remain close to one (0.9958–1.0081), indicating that routing does not systematically favor the older expert group at the reported stages. We also compare the mean-layer MaxVio and auxiliary loss with fixed-expert baselines at the same expert count. EMO has lower auxiliary loss in all three available comparisons and lower MaxVio at $E = 32$ and $E = 128$; at $E = 16$, its MaxVio is slightly higher. Together with the utilization curves, these results support balanced use of new experts.

Table 5: Routing diagnostics at successive expanded capacities. Fixed denotes the baseline with the same expert count. There is no Fixed$_{E=64}$ baseline, so the paired MaxVio and AuxLoss entries at $E = 64$ are N/A; the new-to-old load ratio at this stage is measured.

Stage	New/old load	Mean-layer MaxVio EMO / Fixed	AuxLoss EMO / Fixed
Stage 2 ($E = 16$)	1.0081	0.0580 / 0.0527	0.0027 / 0.0038
Stage 3 ($E = 32$)	0.9958	0.0948 / 0.1164	0.0059 / 0.0080
Stage 4 ($E = 64$)	1.0056	N/A	N/A
Stage 5 ($E = 128$)	0.9976	0.3011 / 0.3348	0.0124 / 0.0158

E.2 Representation-Collapse Diagnostic

Balanced token counts alone do not characterize representation diversity. We therefore separately evaluate the representation-collapse (RC) metric of Chi et al. (2022) on the validation set described in Appendix B. Table 6 reports the last checkpoint of each listed EMO stage and the corresponding fixed-expert baseline. Larger RC values indicate less collapse. EMO has higher RC in each available

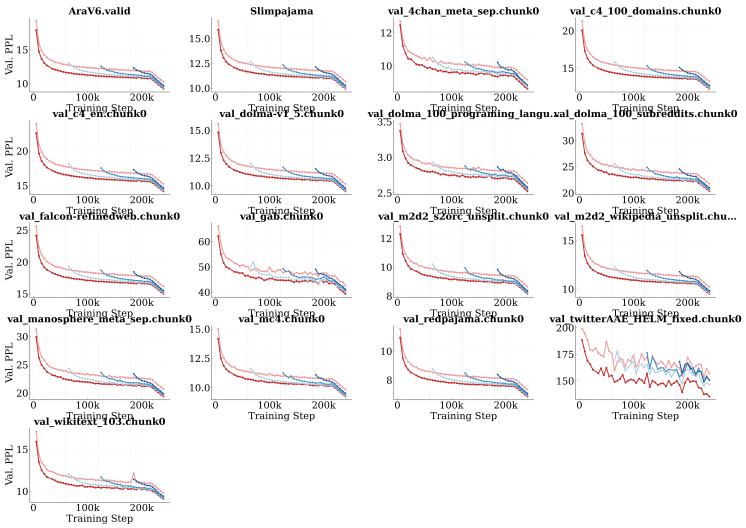


Figure 15: Validation Perplexity of expansion timing experiments (Expand@25%,50%, 75%).

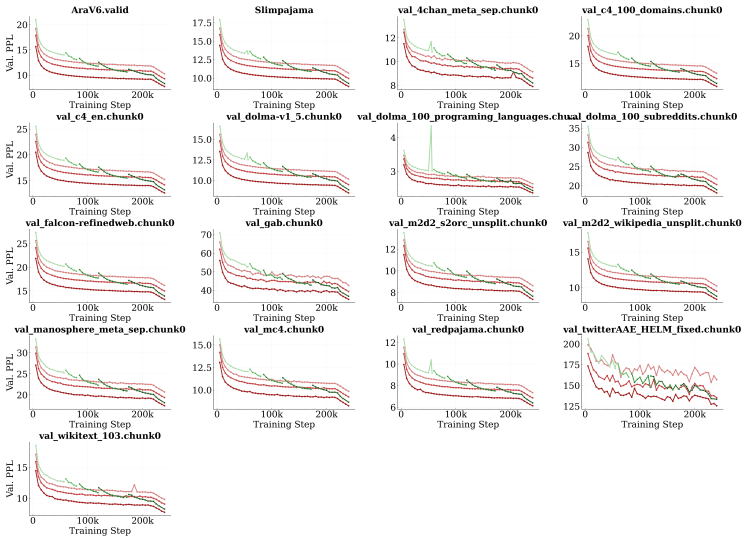


Figure 16: Validation Perplexity of main experiments.

comparison, providing direct evidence that its smaller expert pool in earlier stages does not lead to greater collapse at these evaluated stage endpoints. At $E = k = 8$, EMO has not yet undergone expansion; we report these diagnostics for the subsequent stages. We interpret RC through comparisons at the same E, rather than through its absolute change across different expert counts.

Table 6: Representation-collapse diagnostic at the endpoints of the listed EMO stages, compared with fixed-expert baselines of the same capacity. Higher is better.

Expert count	RC (EMO)	RC (Fixed)
$E = 16$	58.96	47.50
$E = 32$	146.10	111.56
$E = 128$	1046.49	781.91

Expansion may introduce additional router directions as the router, experts, and backbone adapt jointly. The RC results support representation diversity but do not establish this mechanism.

E.3 Quality and Time Savings Across Expansion Timings

Table 7 supplements the $E = 16 \to 32$ timing ablation with its quality–efficiency trade-off. For this timing analysis, the fixed-$E = 32$ reference loss is $L_{\text{ref}} = 1.062$. We express the relative loss increase as $100(L - L_{\text{ref}})/L_{\text{ref}}$ and divide it by the percentage of training time saved. This ratio summarizes the relative loss penalty per percentage point of time saved for the tested schedule; a smaller value indicates a more favorable trade-off by this criterion.

Table 7: Quality and time savings in the single-expansion timing analysis. The reference loss of 1.062 belongs to this analysis. Reported values are rounded, so recomputed ratios may differ slightly.

Expansion time	Final loss	Time saved (%)	Loss increase (%)	Ratio
0% (Fixed$_{E=32}$)	1.062	0.00	0.00	N/A
25%	1.069	2.27	0.66	0.290
50%	1.071	4.55	0.85	0.187
75%	1.076	6.82	1.32	0.193

The 50% schedule has the lowest ratio among these tested timings, while the 75% schedule saves more time with a larger final-loss increase. The scaling-law allocation places expansion at approximately 45%, close to the favorable 50% setting. This agreement supports using the fitted scaling law to guide allocation along the chosen expansion path. The ratio is a descriptive criterion for this experiment, and does not establish a universal optimal expansion time.

E.4 Extrapolated One-Shot Comparison

We complement the measured fixed-expert comparisons with a one-shot $E = 8 \to 128$ extrapolation based on explicit assumptions about the pre-expansion trajectory and post-expansion recovery.

Under the budget-matching calculation for the same total token and GPU-hour budgets as EMO, the one-shot transition is placed at approximately 52.2% of training (about 125K steps). During the shared first 56K steps, the $E = 8$ trajectory has a loss gap of approximately 0.0455 relative to Fixed$_{E=16}$, with observed values between 0.0436 and 0.0477. Assuming that this gap persists, extrapolating along the full Fixed$_{E=16}$ trajectory gives an $E = 8$ loss of approximately 1.240 at the transition. Fixed$_{E=128}$ has a loss of 1.082 at that point, giving an estimated transition deficit of 0.158.

To model recovery, we use the existing $E = 16 \to 32$ timing ablation: at a 50% expansion time, 22.8% of the transition deficit remains at the end of training. The timing-based interpolation to 52.2% gives an estimated remaining fraction of 24.9%. Transferring that recovery fraction to one-shot expansion yields

$$\widehat{L}_{\text{one-shot}} = 0.994 + 0.249\,(1.240 - 1.082) \approx 1.033, \tag{5}$$

where 0.994 is the measured final loss of Fixed$_{E=128}$. Under these assumptions, the projected loss exceeds EMO's measured 1.017, suggesting a benefit from intermediate stages.

The extrapolation assumes both persistence of the early $E = 8$ loss gap and transfer of the recovery behavior from a twofold expansion to a sixteenfold expansion. These transitions retain different proportions of existing experts (50% and 6.25%, respectively), so their recovery need not be identical. The estimate therefore provides evidence conditional on the stated trajectory and recovery model; it is not a bound on one-shot performance or a direct experimental ranking.