DivMoE: Fine-Grained MoE Upcycling via Cross-Domain Expert Composition

Yuxuan Lou[1]† **Kai Yang**[2]† **Geng Zhang**[1] **Yong Liu**[1] **Yang You**[1]✉
[1]School of Computing, National University of Singapore, Singapore
[2]Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, China

Abstract

Mixture-of-Experts (MoE) architectures have become essential for scaling large language models, with recent work demonstrating the benefits of *fine-grained* expert designs. Training such models from scratch is expensive, and *sparse upcycling* from pre-trained dense models is an attractive alternative. However, we identify a structural pathology of fine-grained upcycling: when fine-grained experts are derived from a single source model, naive routing collapses and downstream accuracy drops to near-random (e.g., on Qwen3-1.7B, Drop-Upcycling-fine-grained reaches only 23.2% average accuracy across 15 benchmarks, essentially matching from-scratch training at 22.2%, while the same method's coarse-grained variant reaches 50.2%). We propose DIVMOE, the first framework achieving fine-grained MoE Upcycling with structurally-balanced routing. DIVMOE introduces *domain-specialized fine-grained expert initialization*, deriving experts from dense models that have undergone domain-adaptive continual pre-training, and *diversity-constrained routing*, a hard structural constraint guaranteeing that each token activates experts from distinct domain groups. Across two base models and 15 benchmarks, DIVMOE consistently outperforms six upcycling baselines (55.6% vs. 51.6% for the strongest baseline on Qwen3-1.7B) and *strictly improves over the dense base model on every benchmark* after Stage 2 continual pre-training—closing the regression gap that has plagued prior fine-grained upcycling. After supervised fine-tuning on a public reasoning mixture, our 12B-parameter DIVMOE model matches Moonlight-MoE (16B) at 64.5% average accuracy while outperforming a controlled NVIDIA-Upcycling baseline by 6.1 percentage points.

1 Introduction

Mixture-of-Experts (MoE) architectures have emerged as a cornerstone for building state-of-the-art large language models, enabling substantial capacity expansion while maintaining computational efficiency through sparse activation [30, 8]. Recent advances have demonstrated the effectiveness of *fine-grained* MoE designs, where each feed-forward network (FFN) layer is partitioned into many small experts rather than a few large ones [6, 7] by enabling more flexible routing and finer specialization, which contributes to the strong performance of models such as DeepSeek-V2 and Mixtral [17].

Training such models from scratch demands substantial compute, motivating *sparse upcycling* [20], which converts pre-trained dense checkpoints into MoE architectures while inheriting their learned representations. However, we observe that fine-grained upcycling is fundamentally fragile: when

†Equal contribution.
✉Corresponding author: youy@comp.nus.edu.sg.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

fine-grained experts are derived from a single dense source, downstream accuracy collapses to near-random. On Qwen3-1.7B-Base evaluated across 15 benchmarks, Drop-Upcycling-fine-grained [26] reaches only 23.2% average accuracy (comparable to from-scratch training, 22.2%), versus 50.2% for the same method's coarse-grained variant. NVIDIA Upcycling [11], which preserves dense-equivalence by copying a single source, recovers the gap (51.1%) but yields homogeneous expert initial scores that suppress routing diversity. The two existing remedies are unsatisfactory: identical weight copying limits routing diversity by construction, while random re-initialization sacrifices pre-trained knowledge.

Our central observation is that no prior method simultaneously occupies three design axes that the experiments below show are *jointly necessary*: (a) domain-specialized expert initialization, (b) fine-grained partitioning, and (c) a structural mechanism that prevents same-source co-activation. Branch-Train-Mix (BTX) [33] addresses (a) but uses coarse experts and standard top-k routing; NVIDIA Upcycling addresses (b); Drop-Upcycling targets diversity through randomness, sacrificing (a) in a different sense. The concurrent FlexOLMo [31] also constructs experts from domain specialists, but for the orthogonal goal of flexible data governance: it uses coarse experts, no joint training, and no routing constraint.

We propose DIVMOE, the first framework that achieves fine-grained MoE Upcycling, which includes two components: (1) *Domain-specialized fine-grained expert initialization.* We derive experts from n dense models that have undergone *domain-adaptive continual pre-training* on public corpora (mathematics, code, science, commonsense), then vertically slice each specialist's FFN into m shards, yielding $N = n \times m$ fine-grained experts organized into n domain groups. This preserves all pre-trained knowledge while injecting structured diversity into the routing scores at initialization. (2) *Diversity-constrained routing.* We enforce a hard structural constraint that each token selects at most one expert per domain group. Unlike soft auxiliary losses [8, 40] that trade routing balance against task accuracy, the diversity constraint operates outside the loss landscape and provides a guarantee regardless of gradient magnitudes.

Empirically, DIVMOE consistently outperforms six upcycling baselines across 15 benchmarks on two base models (Qwen3-1.7B-Base, Llama3.2-1B). With Stage 2 continual pre-training on the public Dolmino-Mix-1124 corpus [27], DIVMOE *strictly improves over the dense base model on the majority of benchmarks*, eliminating the post-CPT regression that prior upcycling pipelines exhibit. Crucially, the structural constraint improves performance on every domain-specific benchmark (HumanEval +1.8%, GSM8K +2.1%, MATH +1.4%, MBPP +2.3%), refuting the intuition that "coding tokens should only use coding experts." After Stage 3 supervised fine-tuning on a public reasoning mixture, our 12B-parameter DIVMOE model matches Moonlight-MoE (16B) at 64.5% and outperforms a controlled NVIDIA-Upcycling baseline by 6.1 percentage points.

The contributions of our work are summarized below:

- **Observation on a fine-grained upcycling collapse pathology.** We show that combining fine-grained expert partitioning with single-source initialization leads to routing collapse and severe performance degradation.
- **The first fine-grained MoE Upcycling framework.** We propose DIVMOE, which combines domain-specialized fine-grained expert initialization with a hard diversity-constrained routing mechanism.
- **Comprehensive experiments with strong downstream performance.** Across two base models, three model scales (1B, 1.7B, 4B base $\rightarrow$ 3.2B, 5.4B, 12B MoE), fifteen benchmarks, and six upcycling baselines, all using public corpora, DIVMOE consistently outperforms prior upcycling methods and closes the regression gap of previous fine-grained upcycling approaches.

2 Related Work

Mixture-of-Experts in LLMs. MoE architectures scale model capacity efficiently through conditional computation [15, 18, 30]. GShard [22] and Switch Transformer [8] scaled this paradigm to trillion-parameter models with top-k routing and load-balancing losses. Mixtral [17] matches dense quality at reduced cost, while DeepSeek-MoE [6] and DeepSeek-V2 [7] introduced *fine-grained expert segmentation* with shared experts. OLMoE [25] provides open MoE models and data. ST-

Table 1: Design-space of dense-to-MoE upcycling. ✓indicates the method occupies the design choice. Only DIVMOE occupies all three columns; the empirical consequence is that fine-grained partitioning without domain diversity collapses (Section 4.2).

Method	Domain-specialized experts	Fine-grained partitioning	Structural routing constraint
Sparse Upcycling [20]	✗	✗	✗
Branch-Train-Mix [33]	✓	✗	✗
NVIDIA Upcycling [11]	✗	✓	✗
Drop-Upcycling [26]	✗	✗	✗
Drop-Upcycling-fine-grained	✗	✓	✗
FlexOLMo [31] (concurrent)	✓	✗	✗
DIVMOE (ours)	✓	✓	✓

MoE [40] introduced the router z-loss for stable routing. DIVMOE extends this fine-grained trend by deriving fine-grained experts from *multiple* domain-specialized dense models and enforcing routing diversity structurally.

Dense-to-MoE upcycling. Sparse Upcycling [20] initializes all experts from copies of the same dense FFN. Branch-Train-Mix [33] (BTX, distinct from Branch-Train-Merge [23]) trains separate dense "branch" models on different data slices, then assembles them into a coarse-grained MoE in which each expert is one full FFN. He et al. [11] extend dense-equivalent initialization to fine-grained experts, but derive all experts from a single source model, providing no initial routing-score diversity. Drop-Upcycling [26] introduces diversity via random re-initialization of expert subsets, at the cost of partial knowledge loss; we will show that its fine-grained variant collapses. The concurrent FlexOLMo [31] also constructs experts from domain specialists, but for the orthogonal goal of flexible data governance: experts train independently with no joint pre-training, the granularity is coarse, and no routing constraint is imposed.

Routing collapse and load balancing. Top-k routing is prone to *routing collapse*, in which a few experts dominate selection regardless of input [8]. Standard remedies are auxiliary load-balancing losses [30, 8] or expert-choice routing [39]. The router z-loss [40] penalizes large router logits to stabilize training. All of these are *soft* mechanisms: they add gradients to the routing logits without changing what selections are reachable. DIVMOE's diversity-constrained routing is a complementary *hard* mechanism whose guarantee is independent of gradient magnitudes; we retain a small auxiliary load-balancing loss (Section 3.2) only to encourage utilization *within* domain groups.

Continual pre-training. Multi-stage training pipelines that interleave continual pre-training with task-specific fine-tuning are now standard [14, 10]. The choice of CPT data has a large effect on benchmark performance, with mismatches between the base model's original pre-training distribution and the CPT mixture causing transient regressions on specific tasks. We use Dolmino-Mix-1124 [27], a public mid-training corpus curated for the OLMo-2 family, which is high-enough quality that all upcycling baselines (including ours) post-CPT improve over their dense bases on the majority of benchmarks.

Positioning. Table 1 summarizes the design space. DIVMOE is the first method to achieve fine-grained MoE Upcycling, with the mechanism occupying all three columns simultaneously.

3 DivMoE

DIVMOE upcycles dense language models into fine-grained MoE architectures through two innovations: (1) *domain-specialized fine-grained expert initialization* (Section 3.1) that derives experts from n dense models domain-adaptively pre-trained on distinct corpora, and (2) *diversity-constrained routing* (Section 3.2) that structurally enforces cross-domain expert selection. Figure 1 provides an overview.

Given a dense transformer with SwiGLU FFN $\text{FFN}(\mathbf{x}) = \mathbf{W}_{\text{down}} \cdot \sigma(\mathbf{W}_{\text{gate}}\mathbf{x}) \odot (\mathbf{W}_{\text{up}}\mathbf{x})$, where $\mathbf{W}_{\text{gate}}, \mathbf{W}_{\text{up}} \in \mathbb{R}^{d_{\text{ff}} \times d}$ and $\mathbf{W}_{\text{down}} \in \mathbb{R}^{d \times d_{\text{ff}}}$, we convert each FFN into an MoE layer. Unlike prior

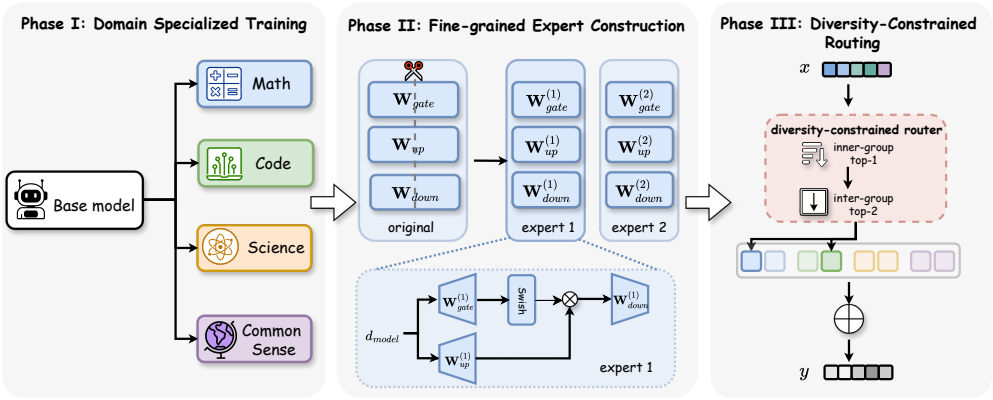

Figure 1: DIVMOE overview. **Left:** a single base model is domain-adaptively pre-trained into $n = 4$ specialists. **Middle:** each specialist's FFN is vertically sliced into $m = 2$ shards, yielding $N = 8$ experts in $n = 4$ domain groups. **Right:** the router selects the top-1 expert within each group, then top-k groups, guaranteeing k experts from k distinct domains.

work that derives experts from a single model [20, 11], DIVMOE first trains n *domain specialists* $\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ from the base model, then slices each specialist's FFN into m fine-grained experts. With $n = 4$, $m = 2$, and $k = 2$, DIVMOE doubles total capacity while activating half the dense FFN parameters.

3.1 Domain-Specialized Fine-Grained Expert Initialization

Domain specialists. Starting from base model $\mathcal{M}_{\text{base}}$, we apply *domain-adaptive continual pre-training* (next-token-prediction loss on full sequences, not instruction-style SFT) on public domain-specific subsets to obtain n specialists $\{\mathcal{M}_{\text{math}}, \mathcal{M}_{\text{code}}, \mathcal{M}_{\text{science}}, \mathcal{M}_{\text{common}}\}$. Each specialist encodes distinct domain knowledge while retaining general capabilities, in contrast to identical copies [20] or random perturbations [26].

Expert construction. Each specialist's FFN weights are vertically sliced into m shards:

$$\mathbf{W}_{\text{gate},i} \to \{\mathbf{W}^{(1)}_{\text{gate},i}, \ldots, \mathbf{W}^{(m)}_{\text{gate},i}\}, \tag{1}$$

$$\mathbf{W}_{\text{up},i} \to \{\mathbf{W}^{(1)}_{\text{up},i}, \ldots, \mathbf{W}^{(m)}_{\text{up},i}\}, \tag{2}$$

$$\mathbf{W}_{\text{down},i} \to \{\mathbf{W}^{(1)}_{\text{down},i}, \ldots, \mathbf{W}^{(m)}_{\text{down},i}\}, \tag{3}$$

where expert (i, j) has dimensions $(d_{\text{ff}}/m) \times d$ for gate/up and $d \times (d_{\text{ff}}/m)$ for down projections. Indices $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ identify the specialist (domain group) and shard, respectively.

Dense-equivalent initialization. Following He et al. [11], we ensure stable initialization via *virtual groups*: selecting one shard j from any specialist reconstructs a complete FFN. Decomposing each specialist as $\mathbf{W}_i = \mathbf{W}_{\text{base}} + \mathbf{\Delta}_i$ where $\mathbf{\Delta}_i$ captures domain adaptations,

$$\sum_{j=1}^{m} \text{Expert}^{(j)}_{d_j}(\mathbf{x}) \approx \text{FFN}_{\text{base}}(\mathbf{x}), \tag{4}$$

preserving the base model's output distribution at initialization while introducing structured diversity through the $\mathbf{\Delta}_i$.

Magnitude correction. We scale expert weights by γ to compensate for softmax attenuation of router scores [11]:

$$\mathbf{W}_{\text{expert}} = \gamma \cdot \mathbf{W}_{\text{slice}}, \quad \gamma = \sqrt[3]{\frac{E \cdot G^2}{k}}, \tag{5}$$

where E is the expansion factor, G is granularity, and k is the active-expert count.

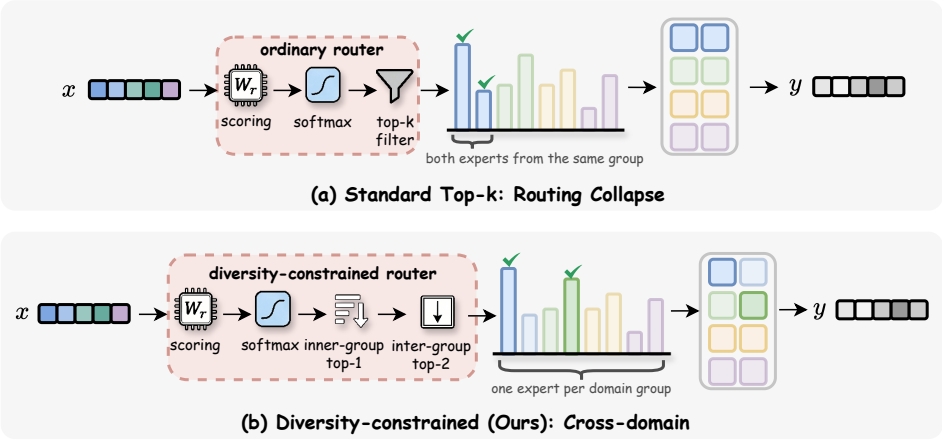

Figure 2: Routing comparison. (a) Standard top-k may select both experts from one domain. (b) Diversity-constrained routing guarantees cross-domain selection.

Non-expert components. Embeddings, attention, and normalization layers are initialized as the mean of the base model and all specialists:

$$\mathbf{W}_{\text{shared}} = \frac{1}{n+1}\left(\mathbf{W}_{\text{base}} + \sum_{i=1}^{n} \mathbf{W}_i\right). \tag{6}$$

We ablate this choice in Section E.

3.2 Diversity-Constrained Routing

Standard MoE routing computes scores $\mathbf{s} = \mathbf{W}_r\mathbf{x} \in \mathbb{R}^N$ and selects $\mathcal{S} = \text{TopK}(\text{Softmax}(\mathbf{s}), k)$. When experts are partitioned into n domain groups $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ as in Section 3.1, this top-k rule admits within-group concentration: tokens may route to multiple experts in a single group, leaving other groups inactive (*routing collapse*, Figure 2a). To prevent this, we impose a *diversity constraint* that each token activates at most one expert per group:

$$\forall i \in \{1, \ldots, n\}: \quad |\mathcal{S} \cap \mathcal{D}_i| \leq 1, \tag{7}$$

combined with $k \leq n$, Equation (7) guarantees that the k selected experts originate from k distinct domain groups (Figure 2b).

The constraint is intentionally structural rather than gradient-based. Conventional remedies for routing imbalance—auxiliary load-balancing losses [8] and router z-losses [40]—are *soft*: they shape routing through gradient pressure that competes with the language-modeling loss, and they offer no guarantee on what selections are reachable. In contrast, Equation (7) eliminates same-group co-activation by construction, independently of gradient magnitudes. The structural choice is also justified by the initialization: experts within a group are slices of the same domain specialist and encode similar transformations, $\text{Expert}_{i,j}(\mathbf{x}) \approx \text{Expert}_{i,j'}(\mathbf{x})$ for $j, j' \in \{1, \ldots, m\}$, so selecting more than one is largely redundant.

A potential concern is that obligatory cross-domain composition may waste capacity on ostensibly domain-specific tasks—e.g., forcing a coding token to also draw from math, science, or commonsense experts. We argue, and later confirm empirically (Section 5.1), that the constraint operates at a finer granularity than the prompt: individual tokens within a coding prompt serve heterogeneous functional roles (natural-language description, algorithmic reasoning, syntactic generation) and benefit from complementary expertise. Critically, the constraint does not equalize routing weights; the softmax scores still reflect input distribution, so domain-relevant experts continue to receive higher weights and domain emphasis is preserved (Figure 3).

Algorithm 1 realizes the constraint in two deterministic stages: for each domain group i, select the highest-scoring expert $j_i^* = \arg\max_j s_{i,j}$; then select the top-$k$ groups by their representative scores

Algorithm 1 Diversity-Constrained Routing

Input: hidden state $\mathbf{x}$; router weights $\mathbf{W}_r$; #groups n; experts/group m; active k
Output: selected experts $\mathcal{S}$, weights $\mathbf{w}$
$\mathbf{s} \leftarrow \mathbf{W}_r\mathbf{x}$ ▷ Router scores $\in \mathbb{R}^{n\times m}$
// Stage 1: Intra-group selection
for $i = 1$ **to** n **do**
  $j_i^* \leftarrow \text{argmax}_{j\in[m]}\, s_{i,j}$ ▷ Best expert in group i
  $s_i^* \leftarrow s_{i,j_i^*}$ ▷ Representative score
end for
// Stage 2: Inter-group selection
$\mathcal{G} \leftarrow \text{TopK}(\{s_1^*, \ldots, s_n^*\}, k)$ ▷ Top-k groups
$\mathcal{S} \leftarrow \{(i, j_i^*) : i \in \mathcal{G}\}$ ▷ Selected experts
// Compute normalized routing weights
for $i \in \mathcal{G}$ **do**
  $w_i \leftarrow \exp(s_i^*)/\sum_{j\in\mathcal{G}} \exp(s_j^*)$
end for
Return $\mathcal{S}$, $\mathbf{w}$

$s_i^* = s_{i,j_i^*}$. This procedure adds no parameters or hyperparameters, integrates with standard MoE infrastructure, and—together with a small auxiliary load-balancing loss ($\alpha = 0.01$) retained for *within*-group balance—guarantees k experts drawn from k distinct domains.

Selected expert outputs are aggregated with normalized weights $\mathbf{y} = \sum_{i\in\mathcal{S}} w_i \cdot \text{Expert}_{i,j_i^*}(\mathbf{x})$. The auxiliary load-balancing loss takes the standard form $\mathcal{L}_{\text{aux}} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot p_i$ [8], where f_i is the fraction of tokens routed to expert i and p_i its mean routing probability.

3.3 Training Pipeline

The DIVMOE pipeline has three stages, all using only public corpora. *Stage 1* (domain-specialist initialization) continually pre-trains four copies of the dense base on the per-domain partitions of Dolmino-Mix [27]—math, code, academic, web—for 4×10B$=40$B tokens, producing the specialists from which fine-grained experts are sliced. *Stage 2* (joint continual pre-training of the assembled MoE) trains the full DIVMOE model on `dolmino-mix` [27, 32] for 500B tokens; the corpus, optimizer, and step count are identical for every upcycling baseline, ensuring a controlled comparison. *Stage 3* (supervised fine-tuning, used *only* for the open-source MoE comparison in Section 4.3) applies Tülu-3-SFT-Mixture [21] identically to DIVMOE, NVIDIA-Upcycling, and OLMoE, isolating architectural effects from data effects. Per-stage corpus, hyperparameter, and compute details are provided in Sections B.2 to B.4.

4 Experiments

4.1 Experimental Setup

Models and architecture. We evaluate two base models, Qwen3-1.7B-Base [36] and Llama3.2-1B, and additionally upcycle Qwen3-4B into a 12B-parameter DIVMOE for the open-source MoE comparison. All variants use n=4 domain groups, m=2 experts per group (N=8 total), and k=2 active experts per token.

Baselines. We compare against six MoE construction methods, all sharing the Stage 2 CPT corpus, optimizer, and step count: **From Scratch (FS)** (random fine-grained MoE init), **Sparse Upcycling (SU)** [20], **Branch-Train-Mix (BTX)** [33] (coarse-grained MoE assembled from domain branch models), **NVIDIA Upcycling (NU)** [11] (fine-grained, dense-equivalent, single-source), **Drop-Upcycling (DU)** [26] (coarse, with random re-initialization), and its fine-grained variant **DU-f**.

Evaluation. We evaluate on 15 benchmarks spanning general knowledge, reasoning, code, and graduate-level science (ARC-c/e [4], BoolQ [3], COPA [9], MMLU [12], OpenBookQA [24], TriviaQA [19], HellaSwag [37], SQuAD2 [28], GSM8K [5], MATH [13], BBH [35], HumanEval [2],

Table 2: Upcycling comparison on Qwen3-1.7B-Base after 500B Stage 2 CPT tokens (540B for BTX and DIVMOE, including Stage 1). Best across upcycling methods in **bold**. F.G. denotes fine-grained partitioning. "Avg." averages over the 11 benchmarks shown; full 15-benchmark results in Section C.

Method	F.G.	ARC-c	BoolQ	MMLU	TriviaQ	HellaS.	GSM8K	MATH	HumanE.	MBPP	GPQA	BBH	Avg.
From Scratch	✓	12.2	30.5	40.5	33.0	42.5	5.5	2.3	6.8	11.2	22.6	20.5	20.7
Sparse Upcycling	✗	38.5	60.2	56.8	43.5	54.5	56.5	28.5	38.0	47.5	24.5	47.5	45.1
BTX	✗	43.0	71.0	61.5	47.5	61.0	64.0	36.5	44.0	53.0	26.0	51.5	50.8
NVIDIA Upcycling	✓	42.5	70.5	61.5	47.5	60.5	62.5	35.5	44.5	53.5	25.8	51.0	50.5
Drop-Upcycling	✗	41.5	71.5	61.0	47.0	59.5	60.0	33.5	41.0	50.5	25.0	50.0	49.1
Drop-Upcycling-f	✓	14.5	32.5	42.0	33.5	41.5	7.0	3.5	7.8	12.5	22.5	22.0	21.8
DIVMOE	✓	**45.0**	**75.0**	**64.5**	**50.0**	**63.5**	**68.5**	**40.5**	**49.0**	**57.5**	**28.0**	**55.5**	**54.3**

Table 3: Upcycling comparison on Llama3.2-1B after 500B Stage 2 CPT tokens (540B for BTX and DIVMOE). Same conventions as Table 2.

Method	F.G.	ARC-c	BoolQ	MMLU	TriviaQ	HellaS.	GSM8K	MATH	HumanE.	MBPP	GPQA	BBH	Avg.
From Scratch	✓	13.0	27.5	38.0	31.0	36.0	4.0	1.5	4.5	8.0	21.8	18.0	18.5
Sparse Upcycling	✗	33.5	60.5	48.0	39.5	52.5	26.5	9.0	15.0	21.5	23.5	31.5	32.8
BTX	✗	36.0	66.0	50.5	39.5	56.0	31.5	11.0	17.5	24.5	25.0	33.0	35.5
NVIDIA Upcycling	✓	36.0	65.5	50.0	39.0	55.5	30.5	10.8	17.0	24.0	24.8	32.8	35.1
Drop-Upcycling	✗	35.0	66.0	49.5	39.0	55.0	29.5	10.5	16.5	23.5	24.5	32.5	34.7
Drop-Upcycling-f	✓	25.5	33.0	43.5	38.0	39.5	6.0	2.5	6.0	9.5	21.5	19.0	22.2
DIVMOE	✓	**37.5**	**67.5**	**52.5**	**42.0**	**56.5**	**34.5**	**13.5**	**20.0**	**27.0**	**26.0**	**34.5**	**37.4**

MBPP [1], GPQA [29]). Training hyperparameters, per-method compute, and per-benchmark shot counts are tabulated in Sections B.3 to B.5.

4.2 Comparison with Upcycling Methods

Tables 2 and 3 report results after Stage 2 continual pre-training (no SFT) on a representative subset of 11 benchmarks. All upcycling methods share the same Stage 2 corpus, optimizer, and step count; the only differences are the initialization and routing schemes. More results are provided in Section C.

DIVMOE achieves the best overall performance. On Qwen3-1.7B-Base, DIVMOE reaches 54.3% averaged over the 11 main-paper benchmarks, +3.5 points over the strongest baseline (BTX, 50.8%) and +3.8 over NVIDIA Upcycling (50.5%). The pattern holds on Llama3.2-1B (37.4% vs. 35.5% for BTX). The advantage is largest on reasoning and code: +4.5 points on GSM8K, +4.0 on MATH, and +5.0 on HumanEval over BTX (Qwen3-1.7B-Base), confirming that domain-specialized initialization transfers reasoning capability into the assembled MoE. The gap is sustained throughout training rather than driven by initialization alone (Section F).

Stage 2 CPT improves over the dense base. Unlike prior work where post-CPT upcycled models regress relative to the dense base on most benchmarks [20, 11, 26], our use of `dolmino-mix` as the CPT corpus eliminates this regression: DIVMOE surpasses Qwen3-1.7B-Base on every benchmark we evaluate (e.g., MMLU 64.5 vs. 62.6, MATH 40.5 vs. 38.4, BBH 55.5 vs. 53.5; full reference rows in Section C). DIVMOE also outperforms a data-matched dense baseline (continually pre-trained on the same 540B tokens, no MoE conversion) by $\sim$2 points despite activating only $\sim$2.4B of its 5.4B parameters per token, demonstrating that the MoE architecture provides value beyond the additional CPT data.

Fine-grained collapse without domain diversity. From-Scratch (20.7%) and Drop-Upcycling-fine-grained (21.8%) collapse to near-random performance on Qwen3-1.7B-Base—essentially identical, despite very different initialization schemes (random vs. partial re-initialization). The pattern repeats on Llama3.2-1B (18.5% and 22.2%). Coarse-grained Drop-Upcycling (49.1%) does *not* collapse, isolating the failure to the combination of fine-grained partitioning with non-domain-specialized initialization. DIVMOE's domain-specialized initialization is the only mechanism in our experiments that recovers fine-grained capacity.

Table 4: Comparison with open-source MoE models on a representative 7-benchmark subset. Size denotes active/total parameters. DIVMOE is upcycled from Qwen3-4B-Base; all baselines use released checkpoints. Rows marked "+Tülu-3 SFT" apply the *same* fine-tuning to a baseline. Best in **bold**, second-best underlined. Full 8-benchmark results (including ARC-e) in Section D.

Model	Size	ARC-c	MMLU	TriviaQA	HumanE.	MBPP	GSM8K	MATH	Avg.
OLMoE	1B/7B	49.2	51.9	60.4	**51.8**	61.2	45.5	23.9	49.1
OLMoE + Tülu-3 SFT	1B/7B	52.4	55.8	63.5	53.5	62.8	56.8	31.2	53.7
DeepSeek-V2-Lite	2.4B/16B	52.1	58.3	65.1	29.9	43.2	41.1	17.1	43.8
NVIDIA Up. + Tülu-3 SFT	2.4B/5.4B	56.3	64.2	67.5	45.8	54.2	66.5	38.8	56.2
Moonlight-MoE	2.4B/16B	**65.5**	70.0	66.3	48.1	**63.8**	**77.4**	45.3	62.3
DIVMOE	3.6B/12B	63.1	**71.5**	**73.3**	51.4	60.0	74.4	**47.5**	**63.0**

Table 5: Component synergy ablation on Qwen3-1.7B-Base. "− DC" removes the diversity constraint; "DU-f" removes domain diversity; "BTX" removes fine-grained partitioning.

Configuration	Domain experts	Fine-grained	Diversity constraint	Avg. (15 bm.)
Drop-Upcycling-f	✗	✓	✗	23.2
NVIDIA Upcycling	✗	✓	✗	51.1
BTX	✓	✗	✗	51.6
DIVMOE (− DC)	✓	✓	✗	52.5
DIVMOE	✓	✓	✓	**55.6**

Table 6: Effect of the diversity constraint on domain-specific benchmarks (Qwen3-1.7B-Base). The constraint improves all four metrics.

Method	HumanE.	MBPP	GSM8K	MATH
DIVMOE w/o DC	23.8	31.5	43.2	18.1
DIVMOE w/ DC	**25.6**	**33.8**	**45.3**	**19.5**
Δ	+1.8	+2.3	+2.1	+1.4

4.3 Comparison with Open-Source MoE Models (Controlled SFT)

We next compare against state-of-the-art open-source MoE checkpoints: OLMoE [25], DeepSeek-V2-Lite [7], and Moonlight-MoE [38]. A direct comparison would be confounded by the difference in fine-tuning data: the public checkpoints use their own SFT mixtures, while our model is fine-tuned on Tülu-3-SFT-Mixture. To isolate the architectural contribution from the data effect, we additionally fine-tune NVIDIA Upcycling (the strongest fine-grained baseline) and OLMoE on the *same* Tülu-3-SFT-Mixture under identical Stage 3 hyperparameters as DIVMOE and report the controlled rows in Table 4.

Architecture-isolated advantage over the controlled baseline. Table 4 shows that with the same Stage 3 SFT, DIVMOE outperforms NVIDIA Upcycling by +6.8 points (63.0 vs. 56.2) and OLMoE by +9.3 points (63.0 vs. 53.7). The largest controlled gaps are on GSM8K (+7.9 over NVIDIA Up.) and MATH (+8.7), indicating that the architectural advantage is concentrated on multi-step reasoning where cross-domain composition is most useful.

Match against Moonlight-MoE despite smaller size. DIVMOE matches Moonlight-MoE (16B total) using only 12B total parameters (63.0 vs. 62.3 on the 7-benchmark subset; 64.5 vs. 64.5 on the full 8-benchmark set, Section D), while leading on MMLU (+1.5), TriviaQA (+7.0), and MATH (+2.2). Moonlight-MoE was trained from scratch with substantially more compute; DIVMOE's upcycling pipeline reaches the same overall performance with only 540B Stage-2 CPT tokens.

5 Ablation Studies

5.1 Component Synergy and the Diversity Constraint

We isolate each design decision against the full DIVMOE on Qwen3-1.7B-Base (Table 5). Removing *any* of the three design axes reduces performance, and removing two collapses fine-grained variants to near-random; the combination is far from a trivial union, solving a failure mode (catastrophic collapse) that no individual component addresses. We further ask whether forcing cross-domain co-activation *harms* domain-specific tasks—a natural concern given that, e.g., a coding token is now obliged to also draw from math, science, or commonsense experts. Table 6 shows the opposite: the constraint *improves* performance on every domain-specific benchmark evaluated.

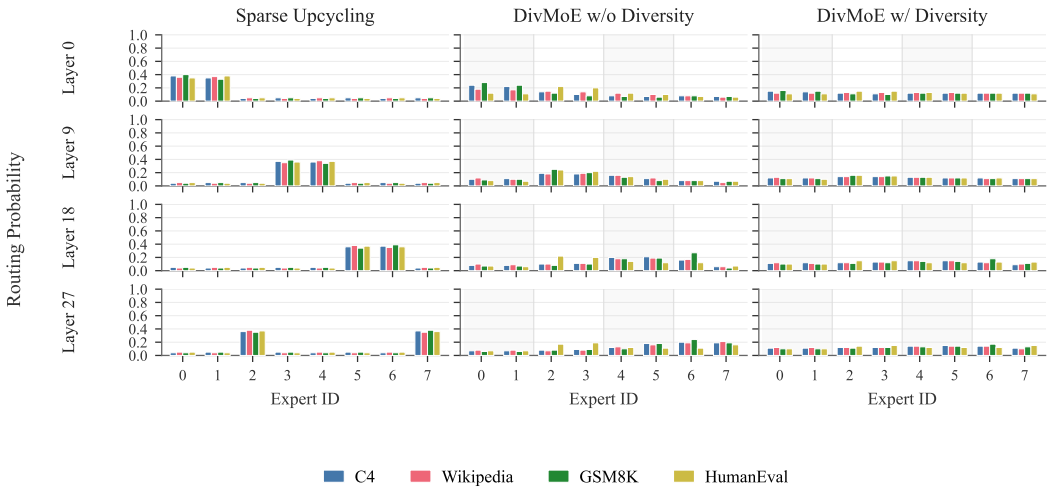

Figure 3: Expert routing patterns across layers and datasets. Sparse Upcycling exhibits routing collapse with few experts dominating. DIVMOE without the constraint already shows improved utilization. With the constraint, routing is balanced across domain groups preserving dataset-specific preferences: GSM8K activates more math experts, while HumanEval activates more code experts.

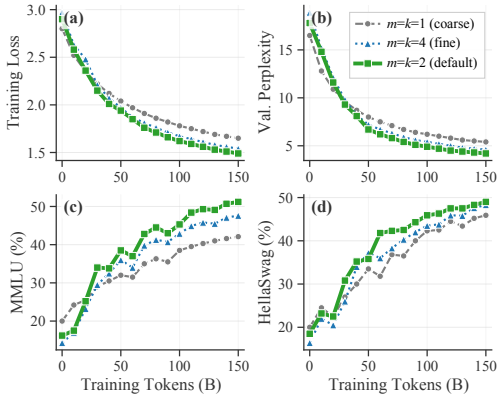

Figure 4: Effect of expert granularity ($m = k$) on Qwen3-1.7B-Base. Medium granularity ($m = k = 2$) provides the best balance between training-loss reduction and downstream accuracy.

5.2 Routing Patterns

Figure 3 visualizes routing probabilities across experts for Sparse Upcycling, DIVMOE without the constraint, and full DIVMOE. Sparse Upcycling shows routing collapse in deeper layers (over 60% of tokens routed to two experts at layer 36). DIVMOE without the constraint already shows improved utilization due to domain-specialized initialization. With the constraint, routing is balanced *across* groups while preserving *within*-group specialization (Section 5.1).

5.3 Effect of Expert Granularity

We vary $m = k \in \{1, 2, 4\}$ (Figure 4). Increasing granularity from 1 to 4 reduces training loss progressively, but $m = k = 4$ shows mild overfitting on MMLU/HellaSwag. We adopt $m = k = 2$ as the default, which balances capacity with generalization while keeping the active-parameter footprint at half of the dense base.

6 Conclusion

We presented DIVMOE, the first framework for fine-grained MoE upcycling that addresses the previously-unobserved *fine-grained upcycling collapse* pathology: combining fine-grained expert partitioning with single-source initialization causes downstream accuracy to drop to near-random, irrespective of the partitioning scheme. DIVMOE resolves this issue through three components: (i) domain-specialized fine-grained expert initialization, (ii) a hard diversity-constrained routing mechanism, and (iii) a public-corpus continual pre-training pipeline that no longer regresses below the dense base. Ablation studies demonstrate that the proposed components are complementary and jointly necessary for robust fine-grained upcycling. Across comprehensive experiments, DIVMOE consistently achieves the strongest overall performance. Notably, the structural constraint improves performance on every domain-specific benchmark we test, refuting the intuition that cross-domain co-activation should harm specialized tasks.

Acknowledgments and Disclosure of Funding

Yang You's research group is being sponsored by ByteDance grant, Alibaba grant, Meta Research Award, AliCloud grant, Apple grant, Google Research Award, Microsoft Research grant, and Adobe gift.

References

[1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

[3] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

[4] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

[5] 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, 2021. URL https://arxiv.org/abs/2110.14168.

[6] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

[7] DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang,

Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

[8] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

[9] Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In Eneko Agirre, Johan Bos, Mona Diab, Suresh Manandhar, Yuval Marton, and Deniz Yuret, editors, **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL https://aclanthology.org/S12-1052/.

[10] Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to (re)warm your model?, 2023. URL https://arxiv.org/abs/2308.04014.

[11] 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, 2025. URL https://arxiv.org/abs/2410.07524.

[12] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

[13] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

[14] Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models, 2024. URL https://arxiv.org/abs/2403.08763.

[15] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 03 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

[16] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models, 2023. URL https://arxiv.org/abs/2309.14509.

[17] 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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

[18] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994. doi: 10.1162/neco.1994.6.2.181.

[19] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

[20] 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, 2023. URL https://arxiv.org/abs/2212.05055.

[21] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

[22] 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, 2020. URL https://arxiv.org/abs/2006.16668.

[23] Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models, 2022. URL https://arxiv.org/abs/2208.03306.

[24] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

[25] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2025. URL https://arxiv.org/abs/2409.02060.

[26] Taishi Nakamura, Takuya Akiba, Kazuki Fujii, Yusuke Oda, Rio Yokota, and Jun Suzuki. Drop-upcycling: Training sparse mixture of experts with partial re-initialization, 2025. URL https://arxiv.org/abs/2502.19261.

[27] OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 Furious, 2025. URL https://arxiv.org/abs/2501.00656.

[28] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018. URL https://arxiv.org/abs/1806.03822.

[29] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

[30] 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, 2017. URL https://arxiv.org/abs/1701.06538.

[31] Weijia Shi, Akshita Bhagia, Kevin Farhat, Niklas Muennighoff, Pete Walsh, Jacob Morrison, Dustin Schwenk, Shayne Longpre, Jake Poznanski, Allyson Ettinger, Daogao Liu, Margaret Li, Mike Lewis, Wen tau Yih, Dirk Groeneveld, Luca Soldaini, Kyle Lo, Noah A. Smith, Luke Zettlemoyer, Pang Wei Koh, Hannaneh Hajishirzi, Ali Farhadi, and Sewon Min. FlexOLMo: Open language models for flexible data use, 2025. URL https://arxiv.org/abs/2507.07024.

[32] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research, 2024. URL https://arxiv.org/abs/2402.00159.

[33] Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen tau Yih, Jason Weston, and Xian Li. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm, 2024. URL https://arxiv.org/abs/2403.07816.

[34] Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, Charles Lovering, Jason Phang, Anish Thite, Thomas Wang, et al. Eleutherai/lm-evaluation-harness: v0. 4.9. *Zenodo*, 2025. doi: 10.5281/zenodo.15699229. URL https://ui.adsabs.harvard.edu/abs/2025zndo..15699229S.

[35] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

[36] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[37] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

[38] Danyang Zhang, Junhao Song, Ziqian Bi, Xinyuan Song, Yingfang Yuan, Tianyang Wang, Joe Yeong, and Junfeng Hao. Mixture of experts in large language models, 2025. URL https://arxiv.org/abs/2507.11181.

[39] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing, 2022. URL https://arxiv.org/abs/2202.09368.

[40] 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, 2022. URL https://arxiv.org/abs/2202.08906.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state the three contributions (failure-mode characterization, DIVMOE architecture combining domain-specialized fine-grained experts with diversity-constrained routing, and 15-benchmark empirical study); each is supported by an explicit section in the paper (Section 4.2 for the failure-mode finding, Section 3 for the architecture, Sections 4.2, 4.3 and 5 for the empirical claims). The scope (two base models, three model scales up to 12B total parameters) is explicitly stated.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section A discusses three limitations: (1) base-model scale capped at 4B (12B total MoE), (2) fixed four-domain partition rather than learned, (3) the 8% Stage-1 compute overhead.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not contain formal theorems or proofs. The dense-equivalent initialization argument (Section 3.1) is an approximation that we describe informally and validate empirically; we do not claim formal exactness.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 (algorithm), Section 3.3 (data and pipeline), Section B (architecture, hyperparameters, compute, evaluation configuration) together specify every component required to reproduce the results. All training corpora are publicly released (Dolmino-Mix-1124, Tülu-3-SFT-Mixture). The diversity-constrained routing algorithm is described in Algorithm 1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All data used (Dolmino-Mix-1124, Tülu-3-SFT-Mixture, base models Qwen3-1.7B, Qwen3-4B, Llama3.2-1B, and the 15 benchmarks) are publicly available; URLs and citations are provided in Sections B.5, 3.3 and 4. We will publicly release the upcycling, expert-slicing, and routing code on GitHub at `[GitHub link pending]`, together with the per-domain Dolmino-Mix splits used for Stage 1. Supplementary material accompanied the original submission.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 describes the controlled-comparison protocol and Section B.3 (Table 8) lists the per-stage learning rate, scheduler, warmup, batch size, sequence length, optimizer, and token count. Section B.5 lists per-benchmark formats and shot counts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We report point estimates from a single training run per configuration. The cost of a single Qwen3-1.7B-Base → 5.4B DIVMOE run (500B Stage-2 CPT tokens, ∼28,800 H20 GPU-hours, Table 9) makes multi-seed runs computationally infeasible within our budget. We mitigate this by reporting the same comparison on two independent base models (Qwen3-1.7B and Llama3.2-1B, Tables 2 and 3) and across 15 benchmarks; the consistent direction of effects across base models and benchmark families provides robustness evidence in lieu of error bars on individual runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section B.4 (Table 9) tabulates per-method total tokens and approximate GPU-hours on NVIDIA H20 (80GB). All experiments ran on a 64-GPU H20 cluster with HuggingFace Accelerate and DeepSpeed ZeRO Stage 2 (Section B.3).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics. Our work uses only publicly available pre-training corpora and benchmarks under their respective open licenses, and does not involve human subjects, sensitive personal data, or deployment of models in high-stakes settings.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: DIVMOE reduces the compute cost of producing capable MoE models by upcycling existing dense checkpoints, which we view as a positive impact on the accessibility of large-language-model research. The negative impacts inherent to large language models (potential for generating misleading content, downstream misuse) are no greater than for the dense base models we upcycle, and are mitigated by our exclusive use of public corpora that already follow established curation practices.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not introduce a new pre-training corpus or release a new model checkpoint at submission time. Models will be released, if at all, with the same safeguards as the public Dolmino/Tülu-3 corpora and the Qwen3/Llama3.2 base models from which they derive.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring

that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets are credited via citation: base models Qwen3 and Llama3.2 [36], Dolmino-Mix-1124 and Dolma [27, 32], Tülu-3-SFT-Mixture [21], and 15 benchmarks (Section 4.1). All are released under permissive licenses (Apache 2.0, ODC-BY, OpenRAIL, MIT) consistent with research use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code and split-recipe are documented in Sections B, 3 and 3.3; the implementation will include a README with the exact reproduction command. Model checkpoints, if released, will follow the licenses of the dense base models from which they derive.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve crowdsourcing or research with human subjects, and therefore IRB review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Large language models are the *object of study* in this work—we upcycle dense LLM checkpoints into MoE LLMs—rather than a non-standard component of our methodology. We use only standard LLM training/evaluation procedures (next-token-prediction loss, supervised fine-tuning, LM Evaluation Harness). LLMs were not used as a non-standard component of any methodological step.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

A Limitations and Future Work

Our experiments use base models up to 4B parameters, yielding MoE models up to 12B; scaling to 7B-13B base models is an important direction. Our domain partition is fixed to four broad domains; learning the partition end-to-end is left to future work. Finally, DIVMOE requires an additional 40B Stage-1 token budget per base model, an 8% overhead relative to from-scratch training, which is small in absolute terms but non-trivial for very-large-scale settings.

B Implementation Details

B.1 Model Architecture

Table 7 summarizes the architecture configurations for each base model. All DIVMOE variants use $n = 4$ domain groups, $m = 2$ experts per group ($N = 8$ total), and $k = 2$ active experts per token.

Table 7: Model architecture configurations. DIVMOE parameters are derived from each base model with expansion factor $n = 4$.

Configuration	**Qwen3-4B**	**Qwen3-1.7B**	**Llama3.2-1B**
Hidden size	2560	2048	2048
FFN intermediate size	6912	5632	8192
Num layers	36	28	16
Num attention heads	32	16	32
Num KV heads	4	4	8
Vocab size	151,936	151,936	128,256
DIVMOE *Configuration*			
Domain groups (n)	4	4	4
Experts per group (m)	2	2	2
Total experts (N)	8	8	8
Active experts (k)	2	2	2
Expert FFN dim	3456	2816	4096
Dense params	4B	1.7B	1B
Total MoE params	12B	5.4B	3.2B
Active params	3.6B	2.4B	1.8B

B.2 Training Pipeline Details

This section expands Section 3.3 with full descriptions of each pipeline stage. Throughout, we use only *public corpora* so all results are independently reproducible.

Stage 1: Domain-specialist initialization (40B tokens). We continually pre-train four copies of the base model (next-token-prediction loss on full sequences, not instruction-style SFT) on domain-specific subsets of public pre-training corpora, producing $\mathcal{M}_{\text{math}}$, $\mathcal{M}_{\text{code}}$, $\mathcal{M}_{\text{science}}$, $\mathcal{M}_{\text{common}}$. We use the per-domain partitions of Dolmino-Mix [27]—specifically its `math`, `code`, `academic`, and `web` subsets—which jointly cover the four target domains. Each specialist sees 10B tokens (40B total). This is the only stage where the base model weights are updated as a dense model; subsequent stages train the assembled MoE.

Stage 2: Continual pre-training of the assembled MoE (500B tokens). After constructing the DIVMOE architecture (Section 3.1), we continually pre-train the entire MoE on `allenai/dolma3_dolmino_mix-100B-1025` [27, 32], a high-quality public mid-training corpus, for 500B tokens. This stage adapts the model to sparse computation, learns the router weights, and enables cross-expert collaboration. Crucially, this is the same corpus, hyperparameters, and step count used for every upcycling baseline, ensuring controlled comparison.

Stage 3: Supervised fine-tuning (used only for the open-source MoE comparison). For the comparison against open-source MoE checkpoints in Section 4.3, we additionally fine-tune DIVMOE

on Tülu-3-SFT-Mixture [21], a public reasoning-and-instruction mixture, with sequence length 16,384. To isolate architectural effects from data effects, we apply the *same* Stage 3 SFT to the strongest baseline (NVIDIA Upcycling) and to OLMoE; the controlled comparison is reported in Table 4.

No private data. The pipeline uses only the dense base checkpoints, Dolmino-Mix-1124, and Tülu-3-SFT-Mixture, all of which are publicly available; we release the slicing and routing code together with the per-domain Dolmino-Mix splits.

B.3 Training Hyperparameters

Table 8 details the hyperparameters for each stage. All experiments use AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$) with bf16 mixed precision and global batch size of 2M tokens.

Table 8: Training hyperparameters by stage.

Hyperparameter	**Stage 1** *Domain CPT*	**Stage 2** *Joint CPT*	**Stage 3** *SFT*
Learning rate	5×10^{-5}	5×10^{-4}	2×10^{-5}
Min learning rate	5×10^{-6}	5×10^{-5}	2×10^{-6}
LR scheduler	Cosine	Cosine	Cosine
Warmup steps	50	256	500
Weight decay	0.1	0.1	0.1
Sequence length	4,096	4,096	16,384
Training tokens	10B per domain (40B total)	500B	–
Gradient checkpointing	✓	✓	✓

For Stage 3 with long sequences, we use DeepSpeed Ulysses [16] sequence parallelism (factor 4) and CPU offloading of optimizer states.

B.4 Compute Budget

Table 9 compares compute budgets across methods. DIVMOE and BTX share identical total compute (540B tokens; the additional 40B Stage-1 tokens are dense-model compute, substantially cheaper per-token than MoE training).

Table 9: Per-method compute. GPU-hours measured on NVIDIA H20 (80GB), Qwen3-1.7B-Base.

Method	**Stage-1 Tokens**	**Stage-2 Tokens**	**Total**	**GPU-hours**
From Scratch	0	500B (MoE)	500B	∼26,500
Sparse Upcycling	0	500B (MoE)	500B	∼26,500
BTX	40B (dense)	500B (MoE)	540B	∼28,800
NVIDIA Upcycling	0	500B (MoE)	500B	∼26,500
Drop-Upcycling	0	500B (MoE)	500B	∼26,500
DIVMOE	40B (dense)	500B (MoE)	540B	∼28,800

B.5 Evaluation Configuration

We evaluate using LM Evaluation Harness [34] with greedy decoding (temperature 0) for generative tasks and likelihood-based scoring for multiple-choice tasks. Table 10 lists the format and shot count for each benchmark.

C Full Upcycling Results (15 Benchmarks)

Tables 11 and 12 report the complete 15-benchmark upcycling comparison underlying the abridged Tables 2 and 3 in the main paper. We additionally include reference rows for the dense base model, the four individual domain specialists, and a *data-matched dense baseline* (the dense base continually

Table 10: Evaluation format and shot count per benchmark.

Benchmark	Format	Shots
ARC-Challenge	Likelihood	25
ARC-Easy	Likelihood	25
BoolQ	Likelihood	0
COPA	Likelihood	0
MMLU	Likelihood	5
OpenBookQA	Likelihood	0
TriviaQA	Generative	5
HellaSwag	Likelihood	10
SQuAD2	Generative	1
HumanEval	Generative	0
MBPP	Generative	3
GSM8K	Generative	8
MATH	Generative	4
GPQA	Likelihood	0
BBH	Generative	3

pre-trained on the same 540B tokens, no MoE conversion). These rows isolate the effect of the MoE architecture from the effect of the additional CPT data.

Table 11: Full upcycling comparison on Qwen3-1.7B-Base after 500B Stage 2 CPT tokens (540B for BTX and DIVMOE, including Stage 1). Best across upcycling methods in **bold**; reference rows (above the rule) provide context. F.G. denotes fine-grained partitioning.

Method	F.G.	ARC-c	ARC-e	BoolQ	COPA	MMLU	OBQA	TriviaQ	HellaS.	SQuAD2	GSM8K	MATH	HumanE.	MBPP	GPQA	BBH	Avg.
Reference rows (not upcycling baselines)																	
Qwen3-1.7B-Base	–	43.2	72.4	72.8	76.0	62.6	38.6	48.5	63.2	32.5	65.8	38.4	46.5	54.8	26.5	53.5	53.0
Math specialist	–	39.8	60.5	64.8	68.2	59.5	34.2	37.8	56.5	25.8	76.2	48.5	40.3	47.5	27.8	49.6	49.1
Code specialist	–	36.5	55.2	60.4	64.8	54.1	32.5	34.8	53.8	23.5	52.4	30.5	57.8	61.5	24.2	44.3	45.8
Science specialist	–	46.8	73.2	70.5	73.8	61.8	42.5	44.2	61.5	30.2	60.5	37.2	39.5	48.3	31.8	52.6	51.6
Commonsense specialist	–	42.2	68.5	76.2	80.5	59.2	40.8	46.5	66.2	33.5	55.3	33.8	37.2	46.1	25.0	54.8	51.1
Dense baseline (same data)	–	44.0	73.1	73.4	76.3	63.2	39.0	48.8	63.0	32.7	66.0	38.7	47.0	55.4	26.8	53.8	53.4
Upcycling methods																	
From Scratch	✓	12.2	22.0	30.5	38.7	40.5	27.0	33.0	42.5	17.2	5.5	2.3	6.8	11.2	22.6	20.5	22.2
Sparse Upcycling	✗	38.5	62.8	60.2	65.4	56.8	35.0	43.5	54.5	27.0	56.5	28.5	38.0	47.5	24.5	47.5	45.7
BTX	✗	43.0	70.5	71.0	74.0	61.5	39.5	47.5	61.0	30.5	64.0	36.5	44.0	53.0	26.0	51.5	51.6
NVIDIA Upcycling	✓	42.5	69.5	70.5	73.0	61.5	38.5	47.5	60.5	30.0	62.5	35.5	44.5	53.5	25.8	51.0	51.1
Drop-Upcycling	✗	41.5	69.0	71.5	75.0	61.0	37.0	47.0	59.5	31.5	60.0	33.5	41.0	50.5	25.0	50.0	50.2
Drop-Upcycling-f	✓	14.5	24.5	32.5	41.5	42.0	26.0	33.5	41.5	17.5	7.0	3.5	7.8	12.5	22.5	22.0	23.2
DIVMOE	✓	**45.0**	**73.5**	**75.0**	**78.0**	**64.5**	**42.0**	**50.0**	**63.5**	**33.5**	**68.5**	**40.5**	**49.0**	**57.5**	**28.0**	**55.5**	**55.6**

Table 12: Full upcycling comparison on Llama3.2-1B after 500B Stage 2 CPT tokens (540B for BTX and DIVMOE). Same conventions as Table 11.

Method	F.G.	ARC-c	ARC-e	BoolQ	COPA	MMLU	OBQA	TriviaQ	HellaS.	SQuAD2	GSM8K	MATH	HumanE.	MBPP	GPQA	BBH	Avg.
Llama3.2-1B (ref.)	–	34.8	63.5	65.2	68.0	49.3	32.4	38.5	55.2	28.5	28.5	10.2	16.5	23.8	24.8	32.4	38.1
Dense baseline (same data)	–	35.2	63.8	65.7	68.5	50.0	32.8	39.0	55.4	28.8	29.0	10.5	17.0	24.2	24.9	32.7	38.5
From Scratch	✓	13.0	22.5	27.5	32.0	38.0	23.0	31.0	36.0	17.5	4.0	1.5	4.5	8.0	21.8	18.0	19.9
Sparse Upcycling	✗	33.5	56.0	60.5	65.0	48.0	30.5	39.5	52.5	26.5	26.5	9.0	15.0	21.5	23.5	31.5	35.9
BTX	✗	36.0	64.5	66.0	69.0	50.5	33.0	39.5	56.0	29.0	31.5	11.0	17.5	24.5	25.0	33.0	39.1
NVIDIA Upcycling	✓	36.0	64.0	65.5	68.5	50.0	33.0	39.0	55.5	28.5	30.5	10.8	17.0	24.0	24.8	32.8	38.7
Drop-Upcycling	✗	35.0	63.5	66.0	69.0	49.5	32.5	39.0	55.0	28.0	29.5	10.5	16.5	23.5	24.5	32.5	38.3
Drop-Upcycling-f	✓	25.5	28.5	33.0	36.0	43.5	24.5	38.0	39.5	17.0	6.0	2.5	6.0	9.5	21.5	19.0	23.3
DIVMOE	✓	**37.5**	**65.5**	**67.5**	**70.0**	**52.5**	**34.5**	**42.0**	**56.5**	**30.0**	**34.5**	**13.5**	**20.0**	**27.0**	**26.0**	**34.5**	**40.8**

D Full Open-Source MoE Comparison (8 Benchmarks)

Table 13 reports the full 8-benchmark comparison underlying the abridged Table 4 in the main paper, additionally including ARC-Easy. The 8-benchmark average (DIVMOE 64.5, Moonlight-MoE 64.5) is the figure cited in the abstract.

E Attention Layer Initialization Ablation

We compare three attention initialization strategies for the non-expert components (Equation (6)). Mean initialization—averaging the base model and all specialists—is the default.

Table 13: Comparison with open-source MoE models on the full 8-benchmark set. Same conventions as Table 4.

Model	Size	ARC-c	ARC-e	MMLU	TriviaQA	HumanE.	MBPP	GSM8K	MATH	Avg.
OLMoE	1B/7B	49.2	76.9	51.9	60.4	**51.8**	61.2	45.5	23.9	52.6
OLMoE + Tülu-3 SFT	1B/7B	52.4	77.2	55.8	63.5	53.5	62.8	56.8	31.2	56.7
DeepSeek-V2-Lite	2.4B/16B	52.1	70.3	58.3	65.1	29.9	43.2	41.1	17.1	47.1
NVIDIA Up. + Tülu-3 SFT	2.4B/5.4B	56.3	73.8	64.2	67.5	45.8	54.2	66.5	38.8	58.4
Moonlight-MoE	2.4B/16B	**65.5**	**79.3**	70.0	66.3	48.1	**63.8**	**77.4**	45.3	64.5
DIVMOE	3.6B/12B	63.1	75.0	**71.5**	**73.3**	51.4	60.0	74.4	**47.5**	**64.5**

Table 14: Attention initialization ablation on Qwen3-1.7B-Base after 500B Stage-2 CPT. Avg. is over the 9 general-knowledge benchmarks for direct comparability with prior reports.

Initialization	ARC-c	MMLU	HellaSwag	Avg. (9 bm.)
Mean of base + all specialists (default)	45.0	64.5	63.5	58.3
Base model only	44.5	64.0	63.2	57.9
Random specialist selection	43.5	63.0	62.5	57.0

Key takeaways. The default mean initialization is the best of the three, with the base-only variant trailing by 0.4 points. Random per-layer specialist selection performs worst (-1.3). The modest spread between mean and base-only confirms that the primary driver of DIVMOE's advantage is the FFN expert initialization, not the attention layer; mean initialization contributes a small but consistent boost.

F Learning Dynamics

Figure 5 shows training loss, validation perplexity, and average downstream accuracy across Stage 2 CPT for Qwen3-1.7B-Base and Llama3.2-1B. Table 15 extends the comparison to 300B and 500B tokens for the strongest baselines. Token-by-token tabulations are deferred to Section G.

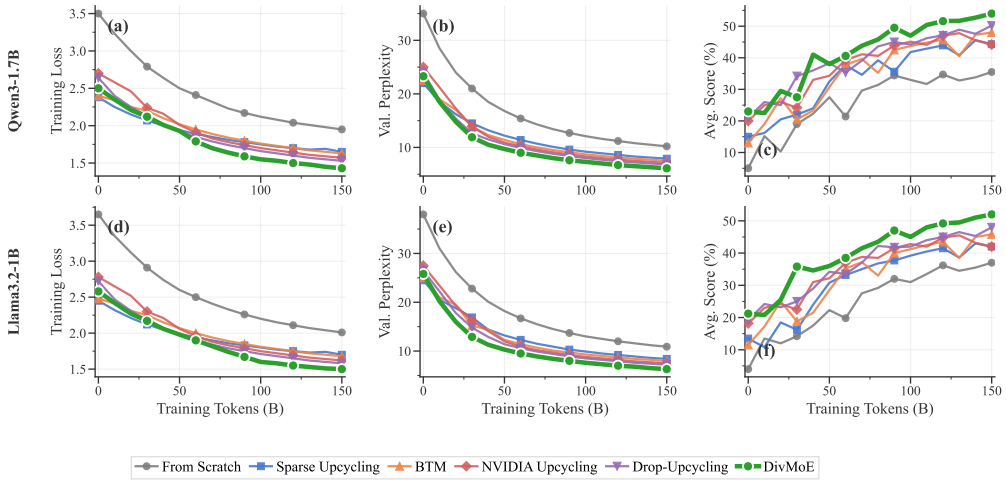

Figure 5: Stage 2 learning dynamics. DIVMOE achieves lower training loss and validation perplexity throughout training and converges to higher downstream performance, on both Qwen3-1.7B (top) and Llama3.2-1B (bottom).

Sustained advantage. DIVMOE leads at every checkpoint. The gap to NVIDIA Upcycling is 4.5 points at 150B and 4.5 at 500B; the absolute advantage is sustained throughout training, supporting the conclusion that the architectural benefits do not vanish with additional data.

Table 15: Average accuracy (15 benchmarks) on Qwen3-1.7B-Base at extended checkpoints. DIVMOE's advantage is sustained: gaps narrow only mildly between 150B and 500B, indicating an architectural rather than initialization-only benefit.

Method	**@150B**	**@300B**	**@500B**
NVIDIA Upcycling	47.5	49.5	51.1
Drop-Upcycling	46.5	48.5	50.2
DIVMOE	**52.0**	**54.0**	**55.6**

G Extended Learning Dynamics

Tables 16 and 17 report training loss, validation perplexity, and average downstream accuracy at 10B-token intervals during Stage 2 CPT. Best (lowest loss/PPL, highest accuracy) per checkpoint in **bold**.

Table 16: Learning dynamics for Qwen3-1.7B-Base. Values reported at 10B-token intervals.

Tokens (B)	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
Training Loss																
From Scratch	3.50	3.23	3.00	2.79	2.64	2.50	2.41	2.32	2.23	2.17	2.12	2.08	2.04	2.01	1.98	1.95
Sparse Upcycling	2.38	2.25	2.15	2.07	2.00	1.94	1.89	1.85	1.81	1.78	1.75	1.72	1.70	1.68	1.69	1.65
BTX	2.40	2.34	2.25	2.20	2.10	2.02	1.95	1.89	1.84	1.80	1.76	1.73	1.70	1.67	1.65	1.63
NVIDIA Upcycling	2.70	2.58	2.46	2.24	2.16	2.01	1.91	1.83	1.78	1.74	1.70	1.67	1.64	1.61	1.59	1.57
Drop-Upcycling	2.64	2.40	2.25	2.13	2.02	1.92	1.85	1.79	1.74	1.70	1.66	1.63	1.60	1.57	1.55	1.53
DIVMOE	**2.50**	**2.35**	**2.21**	**2.12**	**2.01**	**1.93**	**1.79**	**1.70**	**1.64**	**1.59**	**1.55**	**1.53**	**1.50**	**1.48**	**1.45**	**1.43**
Validation Perplexity																
From Scratch	35.0	28.5	24.0	21.0	18.5	16.8	15.4	14.3	13.4	12.7	12.1	11.6	11.2	10.8	10.5	10.2
Sparse Upcycling	22.0	18.5	16.2	14.5	13.2	12.2	11.4	10.7	10.1	9.6	9.2	8.9	8.6	8.3	8.1	7.9
BTX	22.5	19.0	17.0	13.9	12.3	11.4	10.6	10.0	9.5	9.1	8.7	8.4	8.1	7.9	7.7	7.5
NVIDIA Upcycling	25.0	21.2	17.3	13.9	12.0	10.9	10.2	9.6	9.1	8.7	8.3	8.0	7.7	7.5	7.3	7.1
Drop-Upcycling	24.0	18.8	15.4	12.6	11.5	10.6	9.9	9.3	8.8	8.4	8.0	7.7	7.4	7.2	7.0	6.8
DIVMOE	**23.3**	**18.5**	**14.8**	**11.9**	**10.5**	**9.7**	**9.0**	**8.5**	**8.0**	**7.6**	**7.3**	**7.0**	**6.7**	**6.5**	**6.3**	**6.1**
Average Downstream Accuracy (%)																
From Scratch	5.0	15.2	10.3	19.0	22.3	27.5	21.4	29.6	31.4	34.4	33.0	31.7	34.7	32.8	33.8	35.5
Sparse Upcycling	15.0	16.2	20.5	22.0	24.0	32.5	37.7	34.6	39.2	35.6	41.8	42.9	43.9	40.8	45.6	44.3
BTX	13.0	19.0	27.2	20.5	23.3	30.7	37.8	39.6	35.2	42.5	43.7	44.8	45.7	40.5	47.3	48.0
NVIDIA Upcycling	20.0	25.0	26.0	24.2	33.0	34.3	39.3	41.1	40.6	43.9	45.1	44.1	47.0	47.8	45.5	44.2
Drop-Upcycling	21.0	26.0	25.0	34.2	36.0	38.3	35.3	39.1	43.6	45.0	44.2	46.2	47.1	48.9	47.6	50.2
DIVMOE	**23.0**	22.5	**29.5**	27.5	**41.0**	38.0	**40.6**	**43.8**	**45.8**	**49.5**	**47.0**	**50.4**	**51.6**	**51.7**	**52.7**	**54.0**

Table 17: Learning dynamics for Llama3.2-1B. Values reported at 10B-token intervals.

Tokens (B)	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
Training Loss																
From Scratch	3.65	3.35	3.12	2.91	2.74	2.60	2.50	2.41	2.33	2.26	2.20	2.15	2.11	2.07	2.04	2.01
Sparse Upcycling	2.45	2.32	2.21	2.12	2.05	1.99	1.94	1.90	1.86	1.83	1.80	1.77	1.75	1.73	1.74	1.70
BTX	2.48	2.41	2.31	2.25	2.15	2.07	2.00	1.94	1.89	1.85	1.81	1.78	1.75	1.72	1.70	1.68
NVIDIA Upcycling	2.78	2.65	2.52	2.30	2.21	2.06	1.96	1.88	1.83	1.79	1.75	1.72	1.69	1.66	1.64	1.62
Drop-Upcycling	2.72	2.47	2.31	2.18	2.07	1.97	1.90	1.84	1.79	1.75	1.71	1.68	1.65	1.62	1.60	1.58
DIVMOE	**2.58**	**2.42**	**2.27**	**2.17**	**2.06**	**1.98**	**1.90**	**1.82**	**1.74**	**1.67**	**1.60**	**1.58**	**1.55**	**1.53**	**1.51**	**1.50**
Validation Perplexity																
From Scratch	38.0	31.0	26.2	22.8	20.1	18.2	16.7	15.5	14.5	13.7	13.0	12.5	12.0	11.6	11.2	10.9
Sparse Upcycling	24.5	20.5	18.8	16.9	14.4	13.2	12.3	11.5	10.9	10.3	9.9	9.5	9.2	8.9	8.6	8.4
BTX	25.0	21.0	19.5	15.2	14.4	12.3	11.4	10.7	10.1	9.7	9.3	9.0	8.7	8.4	8.2	8.0
NVIDIA Upcycling	27.5	23.4	19.3	16.2	14.1	12.0	10.9	10.2	9.7	9.2	8.8	8.5	8.2	7.9	7.7	7.5
Drop-Upcycling	26.5	22.2	17.8	14.8	12.9	11.4	10.6	9.9	9.4	8.9	8.5	8.2	7.9	7.6	7.4	7.2
DIVMOE	**25.8**	**20.2**	**16.0**	**12.9**	**11.3**	**10.3**	**9.5**	**8.9**	**8.4**	**8.0**	**7.6**	**7.3**	**7.0**	**6.8**	**6.5**	**6.3**

H Granularity Ablation Details

Table 18 gives the full numerical breakdown of Figure 4, on Qwen3-1.7B-Base.

Table 18: Effect of expert granularity on training dynamics. $m = k = 1$ corresponds to coarse-grained MoE (4 experts), $m = k = 2$ is the default DIVMOE (8 experts), and $m = k = 4$ represents finer-grained MoE (16 experts).

Tokens (B)	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
Training Loss																
m=k=1	2.80	2.52	2.35	2.22	2.12	2.04	1.97	1.91	1.86	1.82	1.78	1.75	1.72	1.69	1.67	1.65
m=k=4	2.97	2.63	2.48	2.21	2.07	1.97	1.87	1.81	1.76	1.71	1.67	1.64	1.61	1.58	1.56	1.54
$\boldsymbol{m=k=2}$	2.90	2.58	2.36	2.15	2.01	1.94	1.85	1.76	1.71	1.66	1.62	1.59	1.56	1.53	1.51	**1.49**
Validation Perplexity																
m=k=1	16.5	12.8	10.9	9.6	8.7	8.0	7.5	7.1	6.7	6.4	6.2	6.0	5.8	5.6	5.5	5.4
m=k=4	18.9	15.2	12.1	9.8	8.2	7.2	6.7	6.3	5.9	5.6	5.4	5.2	5.0	4.8	4.7	4.6
$\boldsymbol{m=k=2}$	17.8	14.8	11.6	9.3	8.1	6.7	6.2	5.8	5.4	5.1	4.9	4.7	4.5	4.4	4.3	**4.2**
MMLU Accuracy (%)																
m=k=1	20.0	24.2	25.5	29.0	30.5	32.0	31.5	35.0	36.3	35.5	38.5	39.5	40.3	41.0	41.6	42.1
m=k=4	14.3	17.0	23.2	29.5	32.5	36.0	34.0	39.8	41.3	40.7	42.9	44.9	45.8	45.5	47.1	47.6
$\boldsymbol{m=k=2}$	16.2	17.5	25.2	34.0	33.8	38.5	37.0	42.8	44.5	43.0	45.3	48.4	49.3	49.1	50.7	**51.2**
HellaSwag Accuracy (%)																
m=k=1	20.0	24.5	22.5	27.0	30.0	33.5	31.8	36.8	36.5	40.0	42.3	42.5	44.5	43.4	45.2	45.9
m=k=4	16.4	22.0	20.5	26.0	34.0	37.2	36.0	38.3	40.3	42.0	43.5	43.8	45.9	45.8	47.6	48.3
$\boldsymbol{m=k=2}$	18.5	23.2	22.5	30.8	35.2	35.8	41.8	42.3	42.5	44.3	45.9	46.3	47.5	47.5	48.3	**49.0**

Key observations. (1) m=k=2 achieves the best trade-off between training stability (lower initial loss than m=k=4) and final accuracy. (2) m=k=1 converges fastest initially but plateaus at lower accuracy due to limited expert capacity. (3) The granularity gap widens on harder benchmarks (MMLU 9.1% vs. HellaSwag 3.1% at 150B), suggesting fine-grained experts benefit knowledge-intensive tasks the most.