UNIPOOL: LEARNING EXPERT-TO-LAYER OWNERSHIP FROM BRIEF GLOBAL ACCESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most Mixture-of-Experts (MoE) transformers preassign each expert to one layer for the whole of training. We study whether a brief global-access phase can learn a compatible expert-to-layer allocation that is then executed privately. UNIPOOL$_{\text{lock}}$ gives every layer its own router over one global pool, balances aggregate pool usage, and scores experts with a scale-stable NormRouter. After the short ownership-learning phase, it assigns each expert to one layer, locks the disjoint allocation, and continues training as a layer-private MoE. With the same expert-FFN budget and routed expert FLOPs, this recipe lowers held-out loss relative to vanilla MoE by 0.024–0.037 across five dense-equivalent scales from 182M to 1.5B, including a 0.0247 reduction at 1.5B after 60B tokens. The ownership-learning phase lasts 2K steps, about 3.3% of training, and post-lock step time is within -0.7% to $+2.8\%$ of vanilla MoE in our throughput measurements. Controls locate the gain in full-pool training and in the allocation it produces: a random disjoint allocation fixed at initialization, trained with the same router and losses, matches vanilla, whereas locking the allocation learned during the full-pool phase recovers nearly all of the persistent full-pool gain, and substituting a random allocation at the lock forfeits about half of it. Keeping full-pool access throughout training (UNIPOOL$_{\text{full}}$) also improves over vanilla MoE at four scales and outperforms it with only 66.7% (182M) to 50% (469M and 650M) of its expert parameters.

1 INTRODUCTION

Mixture-of-Experts (MoE) models scale parameter capacity while keeping per-token computation sparse (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). Nearly all modern MoE transformers give each layer a private set of expert FFNs and a private router (Jiang et al., 2024; Dai et al., 2024; DeepSeek-AI, 2024a;b). This design commits each expert to one layer before any expert has been trained, so layer–expert ownership is never informed by data.

Several observations suggest that the fixed allocation wastes capacity. Same-layer experts in trained MoEs share dominant weight subspaces, can be re-routed to similar experts with little loss, and can be pruned heavily, with redundancy concentrated in deep layers (Huang et al., 2026; Wu et al., 2026; Bai et al., 2025). Our own routing probe (Section 3) replaces the learned router of one deep layer in three production MoEs with uniform random routing and loses only 1.0–1.6 downstream accuracy points, so a deep layer's learned choice among its own experts adds little over a random one. These findings describe a symptom. We ask whether the cause is that ownership is fixed before training, when no layer can yet prefer any expert, and whether deciding it after the model has formed such preferences yields a better MoE at no lasting cost.

We propose **UNIPOOL** (**Uni**fied Expert **Pool**; Figure 1). Our main method, UNIPOOL$_{\text{lock}}$, gives every layer its own router over one global pool during a brief ownership-learning phase, then locks a disjoint layer–expert allocation for the remaining training. Two components make this phase trainable. A **pool-level auxiliary loss** balances aggregate utilization across depth instead of forcing every layer to use every expert. **NormRouter** (Zheng et al., 2026) computes L2-normalized, ReLU-sparsified scores whose scale does not depend on layer-specific logit norms. Keeping every layer on the full pool throughout training is costly, however, because exposing all M experts to every layer fragments grouped GEMMs. UNIPOOL$_{\text{lock}}$ therefore uses full-pool access only as a short **ownership-learning phase**. After it, UNIPOOL$_{\text{lock}}$ assigns each expert to exactly one layer from

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

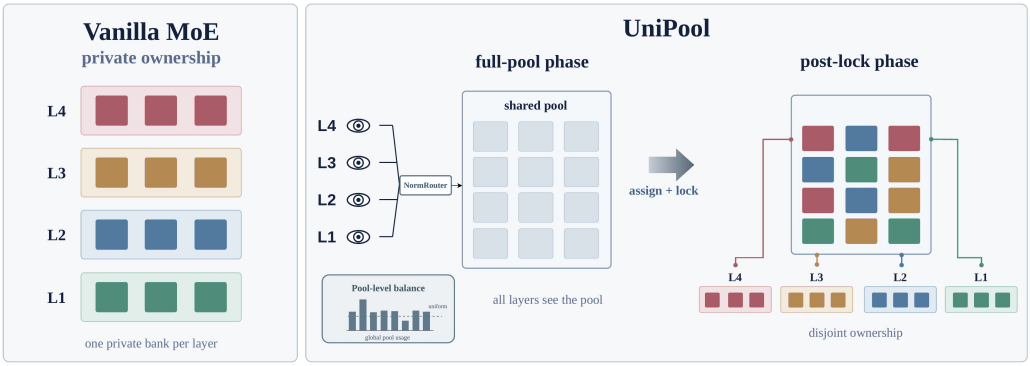


Figure 1: **Expert ownership in vanilla MoE and UNIPOOL.** *Left:* vanilla MoE restricts each layer to a private expert bank throughout training. *Middle:* during a short full-pool phase, every layer routes over one shared pool through its own NormRouter, and a pool-level balance loss equalizes aggregate expert usage across depth rather than within each layer. *Right:* UNIPOOL then assigns each expert to exactly one layer from the observed routing statistics and locks this disjoint allocation (colors mark the owning layer); the remaining steps train a layer-private MoE. The persistent full-pool variant $\text{UNIPOOL}_{\text{full}}$ keeps the middle-panel access pattern throughout training.

routing statistics, attenuates off-allocation scores, and locks a disjoint allocation. From then on the model is a layer-private MoE with vanilla's number of experts per layer but an allocation that was learned rather than fixed.

The central empirical finding is that both the full-pool warm-up and the allocation it produces matter, and that their benefit survives when full-pool access is kept only briefly. Freshly initialized experts are exchangeable, so any disjoint allocation fixed before training is vanilla MoE up to a relabeling of experts; accordingly, at 182M a random disjoint allocation fixed at initialization, trained with the same pool, router, and losses, matches vanilla MoE (1.9319 vs. 1.9317). Locking the allocation selected at step 1K from full-pool routing statistics and enforced at step 2K improves loss to 1.9040, on par with persistent full-pool training (1.9029), whereas locking that same random allocation after the same 2K full-pool steps reaches only 1.9173. A second seed of the same recipe reaches 1.9017, so the two seeds average to 1.9029, exactly the persistent full-pool result. $\text{UNIPOOL}_{\text{lock}}$ improves over vanilla MoE at all five scales we test, recovers nearly all of the persistent full-pool gain in every setting where both exist, and at 1.5B active parameters trained on 60B tokens it reduces test loss from 1.6320 to 1.6073.

Our contributions are as follows.

- **Expert ownership as a learned quantity.** Standard MoEs commit each expert to a layer while the experts are still exchangeable, so the allocation carries no information; our matched controls show that deciding ownership after brief global access, from the routing preferences it produces, recovers the persistent full-pool gain, whereas substituting a random allocation at the lock forfeits about half of it.
- **$\text{UNIPOOL}_{\text{lock}}$.** We combine a global expert pool, a pool-level balancing loss, and NormRouter routing with an ownership-learning phase and a disjoint lock, so that the model trains and executes as a layer-private MoE after the first 2K steps.
- **Consistent gains at vanilla-like cost.** With matched expert-FFN budgets and routed expert FLOPs, $\text{UNIPOOL}_{\text{lock}}$ improves over vanilla MoE from 182M to 1.5B dense-equivalent active scales, with post-lock step time within -0.7% to $+2.8\%$ of vanilla (Table 6).
- **Parameter-efficient persistent full-pool training.** Keeping full-pool access throughout training ($\text{UNIPOOL}_{\text{full}}$) improves over vanilla MoE at four scales and outperforms it with only 66.7% (182M) to 50% (469M and 650M) of its expert parameters, providing a parameter-efficiency reference for how expert budgets can scale with depth.

2 Related Work

Sparse MoE and scaling. Sparsely gated expert layers (Shazeer et al., 2017) were scaled through GShard, Switch Transformer, and ST-MoE (Lepikhin et al., 2021; Fedus et al., 2022; Zoph et al., 2022), supported by systems such as DeepSpeed-MoE, Tutel, and MegaBlocks (Rajbhandari et al., 2022; Hwang et al., 2023; Gale et al., 2023), and adopted by open models including Mixtral, the DeepSeek series, OpenMoE, and LLaMA-MoE (Jiang et al., 2024; Dai et al., 2024; DeepSeek-AI, 2024a;b; Xue et al., 2024; Zhu et al., 2024). Scaling studies examine routed capacity and expert granularity (Clark et al., 2022; Ludziejewski et al., 2024; He, 2024). These models commit experts to layers at initialization, before training can inform ownership; UniPool decides it after a brief global-access phase.

Routing and load balancing. The standard recipe combines softmax routing with the Switch auxiliary loss, which balances token fractions and routing probabilities within each layer (Fedus et al., 2022). Alternatives include expert-choice routing (Zhou et al., 2022), linear assignment (Lewis et al., 2021), hash routing (Roller et al., 2021), sigmoid gating with auxiliary-loss-free balancing (DeepSeek-AI, 2024b; Wang et al., 2024), and ReLU-based routing (Wang et al., 2025). When experts are shared across layers, dead-expert prevention should be defined over the whole pool rather than within each layer; UniPool therefore balances at the pool level and adopts NormRouter scores (Zheng et al., 2026).

Cross-layer parameter sharing and expert reuse. Universal Transformers and ALBERT share entire blocks across depth (Dehghani et al., 2019; Lan et al., 2020), and recursive variants relax such sharing with layer-wise adapters or token-level recursion depths (Bae et al., 2025a;b). MoEUT (Csordás et al., 2024) repeats a small group of shared MoE blocks across depth, and Mixture-of-Modules (Gong et al., 2024) assembles computation dynamically from a shared module set. Closer to our setting, ReXMoE (Tan et al., 2025) lets each router also reach adjacent-layer experts, and MoUE (Chen et al., 2026) combines layer-private experts with universal experts visible along structured depth windows. DeepSeekMoE's shared experts are always active but remain within one layer (Dai et al., 2024). All of these methods prescribe the sharing pattern in advance. UniPool instead learns an allocation from global competition and then removes sharing entirely, and Section 5.3 compares it with predetermined patterns under a matched expert budget.

Changing MoE structure during training. Sparse upcycling initializes an MoE from a dense checkpoint (Komatsuzaki et al., 2023), Drop-Upcycling partially re-initializes the copied experts to restore diversity (Nakamura et al., 2025), EvoMoE anneals a dense-to-sparse gate (Nie et al., 2021), and DS-MoE trains densely while inferring sparsely (Pan et al., 2024). These methods change *how many* experts a token uses over training. UniPool keeps sparsity and expert compute fixed and changes *which layer owns which expert*.

3 Motivating Observation: Substitutable Experts Under Fixed Ownership

Prior analyses find substantial within-layer redundancy in trained MoEs. Same-layer expert weights in Qwen and DeepSeek MoEs share highly similar dominant spectral subspaces (Huang et al., 2026). Rerouting tokens to the most similar same-layer expert preserves accuracy on Qwen1.5-MoE, DeepSeek-V2-Lite, and Qwen3-30B-A3B (Wu et al., 2026). Pruning roughly half the experts of Mixtral 8×7B costs about 8% relative quality, with the strongest similarity in deep layers (Bai et al., 2025). We complement these parameter- and output-level analyses with a probe of the router. In Qwen1.5-MoE, DeepSeek-V2-Lite, and Qwen3-30B-A3B, we replace the learned top-k router of one deep-half MoE layer with uniform random assignment, repeat this for every deep-half layer, and average downstream accuracy over layers (Table 1).

Table 1: Single-layer routing randomization in production MoEs: average raw zero-shot accuracy (%) over five tasks. Random is averaged over deep-half layers; per-task results are in Appendix Table 17.

Model	Top-k	Random	Δ
Qwen1.5-MoE	67.92	66.29	−1.6
DeepSeek-V2-Lite	54.19	53.03	−1.2
Qwen3-30B-A3B	73.02	72.06	−1.0

The average drop is only 1.0–1.6 points. Under fixed private ownership, a deep layer's learned choice among its own experts thus adds little over a random choice: the experts it was given are only weakly differentiated. This is consistent with the hypothesis that how experts are assigned to layers matters more than how many each layer holds. If fixed ownership is the cause, learning ownership should make experts less substitutable even when every layer keeps the same number of experts. Section 5.3 tests the allocation rule under an identical expert budget, and Section 5.4 repeats this probe on our own models, including the locked model.

4 METHOD

UNIPOOL has three training components, global pool routing (Section 4.1), pool-level balancing (Section 4.2), and NormRouter (Section 4.3), plus an ownership lock (Section 4.4). The default method is $\text{UNIPOOL}_{\text{lock}}$: it uses full-pool access only to learn an allocation and then locks that allocation for layer-private execution. $\text{UNIPOOL}_{\text{full}}$ keeps full-pool access throughout training; it serves as a quality reference and enables the reduced-budget analysis in Section 5.4.

4.1 GLOBAL EXPERT POOL

In a standard MoE transformer with L layers and E experts per layer, layer l owns experts $\{e_{l,1}, \ldots, e_{l,E}\}$ and a router r_l:

$$\text{FFN}_l(x) = \sum_{i \in \text{Top-}k(r_l(x))} g_{l,i}(x)\, e_{l,i}(x), \tag{1}$$

where $g_{l,i}(x)$ is the gate that r_l assigns to expert i. UNIPOOL replaces the L private banks with one pool $\mathcal{E} = \{e_1, \ldots, e_M\}$ and keeps a separate router per layer:

$$\text{FFN}_l(x) = \sum_{i \in \text{Top-}k\,\mathcal{V}_l(r_l(x))} g_{l,i}(x)\, e_i(x), \tag{2}$$

where $\mathcal{V}_l \subseteq [M]$ is the set of experts visible to layer l and Top-$k\,\mathcal{V}_l$ selects the k highest-scoring experts within it. During full-pool access $\mathcal{V}_l = [M]$ for every layer, so any layer may select any expert. Routers stay layer-specific because hidden states differ across depth. In matched comparisons $M = EL$, so UNIPOOL stores the same expert FFNs and activates the same number per token as vanilla MoE; reduced-pool experiments set $M < EL$.

4.2 POOL-LEVEL AUXILIARY LOSS

The Switch auxiliary loss for layer l is

$$\mathcal{L}_{\text{aux}}^{(l)} = \alpha\, E \sum_{i=1}^{E} f_i^{(l)} P_i^{(l)}, \tag{3}$$

where $f_i^{(l)}$ is the fraction of layer-l tokens dispatched to expert i and $P_i^{(l)}$ is its mean routing score in that layer. With private experts, balancing each layer is the right way to avoid dead parameters. With a shared pool, an expert that one layer ignores may serve another, so per-layer balancing forces every layer to spread traffic over the whole pool and suppresses layer preferences. We instead balance aggregate utilization. With $\overline{f}_i = \frac{1}{L}\sum_l f_i^{(l)}$ and $\overline{P}_i = \frac{1}{L}\sum_l P_i^{(l)}$,

$$\mathcal{L}_{\text{pool}} = \alpha_{\text{pool}}\, M \sum_{i=1}^{M} \overline{f}_i\, \overline{P}_i = \frac{1}{L} \sum_{l=1}^{L} \alpha_{\text{pool}}\, M \sum_{i=1}^{M} \overline{f}_i\, P_i^{(l)}. \tag{4}$$

The decomposition lets each layer compute its own term. Because $\overline{f}_i$ depends on all layers, we use its value from the previous micro-batch, which avoids cross-layer tensor dependencies under activation checkpointing. Appendix K details this one-step-lagged estimate.

4.3 NORMROUTER

With softmax routing, layers with larger logit norms route more sharply, and in a shared pool these differences directly change which layer claims which experts. We adopt NormRouter (Zheng et al., 2026):

$$s_i = \sigma\, c\, \max\left(0, \frac{z_i}{\|z\|_2 + \epsilon}\right), \qquad z = W_l h, \tag{5}$$

where σ is a learnable scalar initialized to 1, c is a fixed constant that sets the initial scale of the selected scores (Appendix L), and ϵ ensures numerical stability. The norm $\|z\|_2$ is computed over all M logits of layer l. L2 normalization removes layer-dependent logit scale, and ReLU zeroes roughly half of the candidates. Experts are selected by s_i, and both auxiliary losses use each expert's mean score in place of its mean softmax probability. After the lock, we retain the M-row router and the same full-pool normalization, so the visibility mask changes the candidate set without changing score calibration.

4.4 LEARNING OWNERSHIP AND LOCKING IT

Why the default method locks the allocation. For a batch of T tokens, top-k routing over M experts gives each expert about Tk/M tokens. With $M = EL$ visible experts per layer instead of E, the same routed FLOPs are split into L times more, smaller grouped GEMMs. Full-pool steps are 22%–36% slower than vanilla MoE (Table 6). Masking unused experts does not help, because a conventional dispatcher still materializes M columns and launches M expert groups.

Ownership-learning phase. UNIPOOL starts with full-pool access. To make layer preferences concentrate before the lock, we add a target-cardinality regularizer

$$\mathcal{L}_{\text{card}} = \frac{\lambda_{\text{card}}}{L} \sum_{l=1}^{L} \left(H(p_l) - \log K_{\text{tar}}\right)^2, \tag{6}$$

where p_l is layer l's normalized routing-mass distribution over experts and $K_{\text{tar}} = E$, so each layer's effective number of experts $\exp(H(p_l))$ approaches its eventual allocation size. The pool loss simultaneously keeps aggregate usage balanced, which is exactly what a disjoint allocation needs: every layer concentrates on a few experts, and different layers concentrate on different ones.

Allocation, attenuation, and lock. At step 1K we collect routing counts on the held-out 3% validation split (disjoint from the final test tokens) and solve a small assignment problem that gives every expert at most one owner layer. Since $M = EL$ and each layer must receive at least E experts, the solution gives every layer exactly $K_l = E$ experts and uses every expert once; among such allocations it maximizes the retained routing mass. The allocation $\mathcal{C}_l$ is therefore disjoint across layers. Over steps 1K–2K we attenuate off-allocation scores before top-k selection,

$$\widetilde{s}_{l,i}(t) = s_{l,i}(t)\left[1 - \alpha(t)\,\mathbf{1}\{i \notin \mathcal{C}_l\}\right], \tag{7}$$

with $\alpha(t)$ rising from 0 to 1 on a cosine schedule. At step 2K we set $\mathcal{V}_l = \mathcal{C}_l$ permanently. From then on each expert is trained and executed by one layer only, for the remaining 96.7% of a 60K-step run. The 1.5B run follows the same schedule.

Compact dispatch. After the lock, scores and the pool loss are still computed over the global M-expert index, with the mask applied before top-k selection. We gather the dispatch map into each layer's local coordinates, so token permutation, token-count metadata, and grouped GEMM operate on $|\mathcal{V}_l| = E$ expert groups that reference the original parameters without copying them. Execution is then identical in shape to a vanilla MoE layer. Appendix G lists the schedule, assignment constraints, and implementation invariants.

5 EXPERIMENTS

5.1 SETUP

Models and data. We train LLaMA-style backbones (Touvron et al., 2023) at five dense-equivalent active-parameter scales, 182M, 469M, 650M, 830M, and 1.5B (Appendix Table 10). The four scales

Table 2: Test loss (lower is better). Within each column, the MoE variants match the expert-FFN budget and routed expert FLOPs; the wider shared-pool routers add 0.10–0.37% stored and up to 2.2% active parameters (Appendix Table 10). UNIPOOL$_{\text{lock}}$ is the main method: it locks a learned disjoint allocation at step 2K; UNIPOOL$_{\text{full}}$ is the persistent full-pool reference and was not trained at 1.5B. The 182M UNIPOOL$_{\text{lock}}$ entry is the mean of two same-recipe seeds (1.9017 and 1.9040). The five scales use the default 8E/top-1 budget, with 30B training tokens up to 830M and 60B at 1.5B; the last two columns add full-width experts to the 12-layer 182M backbone, giving about 268M (16E/top-2) and 438M (32E/top-4) active parameters. Dense models match the active parameters of the 8E/top-1 MoEs up to 830M. Δ is UNIPOOL$_{\text{lock}}$ minus vanilla MoE.

	8E/top-1					*182M backbone*	
	182M	**469M**	**650M**	**830M**	**1.5B**	**16E/top-2**	**32E/top-4**
L/*H*	12/768	24/1024	36/1024	48/1024	36/1600	12/768	12/768
Dense	2.0420	1.8860	1.8318	1.8032	–	–	–
Vanilla MoE	1.9317	1.7982	1.7568	1.7309	1.6320	1.8589	1.7974
UNIPOOL$_{\text{lock}}$	**1.9029**	**1.7615**	1.7324	1.6944	**1.6073**	1.8291	1.7702
UNIPOOL$_{\text{full}}$	**1.9029**	1.7636	**1.7260**	**1.6923**	–	**1.8277**	**1.7672**
Δ	-0.0288	-0.0367	-0.0244	-0.0365	-0.0247	-0.0298	-0.0272

up to 830M train on the Pile (Gao et al., 2020) for 60K steps with global batch 512 and sequence length 1,024, about 30B tokens, using AdamW (Loshchilov & Hutter, 2019), cosine learning-rate decay, and bf16 in Megatron-LM (Shoeybi et al., 2019). The 1.5B backbone has 36 layers of width 1,600 and trains with global batch 1,024 for 57,220 steps, about 60B tokens.

MoE configurations. Vanilla MoE uses eight private experts per layer, top-1 softmax routing, and a per-layer auxiliary loss. UNIPOOL uses a pool of $M = 8L$ experts, top-1 NormRouter routing, and the pool loss. Both store the same expert FFNs and activate one expert per layer and token; UNIPOOL's wider routers add 0.10%–0.37% stored parameters (Appendix Table 10). We additionally add experts to the 12-layer 182M backbone with 16E/top-2 and 32E/top-4, keeping each expert at full FFN width; this raises the model to about 268M and 438M active parameters (1.46B and 2.82B total for UNIPOOL), and each setting is compared only with vanilla MoE of the same configuration.

Seeds. The 182M vanilla and UNIPOOL$_{\text{full}}$ results average three seeds, with test-loss standard deviation 0.0023. The 182M UNIPOOL$_{\text{lock}}$ entry averages two same-recipe seeds (1.9017 and 1.9040); the other scale results use one seed, and control C in Section 5.3 is the second seed of this recipe. We treat differences below about 0.005 as ties. All reported final losses are measured on the held-out test split, which is disjoint from the 3% validation split used to choose allocations.

5.2 MAIN RESULTS

UNIPOOL improves over vanilla MoE at every scale. Table 2 shows that the default UNIPOOL$_{\text{lock}}$ reduces vanilla test loss at all five scales, by 0.0244–0.0367 at 30B tokens and by 0.0247 at 1.5B after 60B tokens, and by 0.0298 and 0.0272 under the 16E/top-2 and 32E/top-4 budgets. Relative to persistent full-pool training, the lock changes loss by -0.0021 to $+0.0064$ across the six settings where both exist. UNIPOOL$_{\text{full}}$ itself improves over vanilla at all four 30B-token scales, by 0.0288–0.0386. At the four 8E/top-1 scales with matched dense baselines, all MoE variants reach far lower test loss than the dense models.

The gap holds throughout training. UNIPOOL$_{\text{full}}$ stays below vanilla MoE after warmup at 182M, 469M, and 650M, and the gap opens early rather than only at the final checkpoint (Appendix Figure 4). The sharing-scope runs follow the same ordering over training as at the end, with the global pool lowest and the layer-private model highest (panel d).

Downstream accuracy follows loss. We evaluate ARC-Easy and ARC-Challenge (Clark et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), LAMBADA (Paperno et al., 2016), and RACE (Lai et al., 2017) zero-shot (Table 3).

Table 3: Zero-shot downstream accuracy (%, raw `acc`) of vanilla MoE, UNIPOOL$_{\text{full}}$, and UNIPOOL$_{\text{lock}}$. The bottom block adds full-width experts to the 12-layer 182M backbone, giving about 268M and 438M active parameters. UNIPOOL$_{\text{full}}$ was not trained at 1.5B. **Bold** marks the best method for each scale and task; Avg is the unweighted mean.

Setting	Scale	Method	ARC-E ↑	ARC-C ↑	PIQA ↑	HellaSwag ↑	WinoGrande ↑	LAMBADA ↑	RACE ↑	Avg ↑
8E / top-1	182M	Vanilla MoE	45.71	19.97	63.11	29.98	**50.99**	32.78	28.61	38.74
		UNIPOOL$_{\text{full}}$	**46.72**	20.48	**64.36**	30.66	**50.99**	**34.56**	**29.47**	**39.61**
		UNIPOOL$_{\text{lock}}$	46.58	**20.56**	64.14	**30.77**	50.81	34.43	29.35	39.52
8E / top-1	469M	Vanilla MoE	50.51	21.08	66.32	32.72	51.14	40.21	29.38	41.62
		UNIPOOL$_{\text{full}}$	**53.16**	21.42	**67.30**	**33.90**	52.72	**42.19**	**31.10**	**43.11**
		UNIPOOL$_{\text{lock}}$	52.88	**21.57**	66.94	33.71	**52.85**	41.82	30.88	42.95
8E / top-1	650M	Vanilla MoE	51.94	21.25	67.03	34.53	**53.04**	43.74	29.76	43.04
		UNIPOOL$_{\text{full}}$	52.02	**22.61**	67.90	35.55	52.49	44.28	31.67	43.79
		UNIPOOL$_{\text{lock}}$	**52.17**	22.52	**68.12**	**35.68**	52.41	**44.46**	**31.80**	**43.88**
8E / top-1	830M	Vanilla MoE	52.53	23.89	**68.93**	35.36	52.33	43.14	30.53	43.82
		UNIPOOL$_{\text{full}}$	**56.57**	25.00	68.77	**36.90**	52.49	**47.37**	**32.63**	**45.67**
		UNIPOOL$_{\text{lock}}$	56.12	**25.18**	68.22	36.58	**52.64**	46.74	32.25	45.39
8E / top-1	1.5B	Vanilla MoE	56.99	25.34	71.16	39.30	55.17	50.98	33.59	47.50
		UNIPOOL$_{\text{lock}}$	**58.50**	**25.85**	**72.31**	**40.49**	**56.27**	**52.07**	**33.63**	**48.45**
16E / top-2	268M	Vanilla MoE	48.82	**21.59**	65.07	31.83	49.72	36.48	28.80	40.33
		UNIPOOL$_{\text{full}}$	**49.24**	20.22	**65.45**	**32.33**	54.22	**37.86**	**29.19**	**41.22**
		UNIPOOL$_{\text{lock}}$	48.93	20.34	65.03	32.14	**54.37**	37.38	28.88	41.01
32E / top-4	438M	Vanilla MoE	50.08	21.08	66.43	32.91	**51.54**	39.41	29.00	41.49
		UNIPOOL$_{\text{full}}$	52.44	**22.27**	67.41	34.32	50.51	40.77	30.62	42.62
		UNIPOOL$_{\text{lock}}$	**52.60**	22.19	**67.62**	**34.44**	50.45	**40.92**	**30.75**	**42.71**

UNIPOOL$_{\text{full}}$ raises average accuracy by 0.87, 1.49, 0.75, and 1.85 points from 182M to 830M, winning 25 of 28 scale–task cells; ARC-E, HellaSwag, and LAMBADA improve at every scale, by up to 4.23 points on LAMBADA at 830M. UNIPOOL$_{\text{lock}}$ tracks UNIPOOL$_{\text{full}}$ downstream as it does in loss: its average stays within 0.3 points of UNIPOOL$_{\text{full}}$ in the six settings where both were evaluated, and it raises average accuracy over vanilla MoE by 0.68–1.57 points across seven settings, winning 43 of 49 cells. At 1.5B, UNIPOOL$_{\text{lock}}$ raises average accuracy from 47.50 to 48.45 and improves all seven tasks.

Post-lock throughput matches vanilla MoE. Table 6 separates the two phases of each UNIPOOL$_{\text{lock}}$ run. Dispatching over the full pool costs 22%–36% more per step than vanilla MoE. After the lock, compact dispatch cuts step time by 16%–27% and brings it to within -0.7% to $+2.8\%$ of vanilla, so the end-to-end overhead of UNIPOOL$_{\text{lock}}$, weighting the two phases by their 2K and 58K steps, is $+0.6\%$ to $+3.5\%$ while its loss stays close to UNIPOOL$_{\text{full}}$ (Table 2). The residual post-lock overhead is the cost of scoring the global M-expert index and pool statistics, which we retain so that the objective is unchanged after the lock.

5.3 WHERE DOES THE GAIN COME FROM?

An allocation fixed before training carries no information. Table 4 groups 182M runs by when the layer-to-expert allocation is decided. At initialization the experts are exchangeable, so a disjoint allocation fixed at step 0 is vanilla MoE up to a relabeling of experts, whichever experts it gives to which layer. Control A confirms this and serves as a sanity check for the other controls: it uses UNIPOOL's pool, router, and losses, and differs from UNIPOOL$_{\text{lock}}$ only in drawing its disjoint allocation at random at step 0, yet it matches vanilla to within 0.0002. Control A therefore isolates the gain to full-pool training itself rather than to the pool, the router, or the losses on their own. Sharing patterns fixed before training, including routed shared cores, adjacent-layer reuse, and windowed universal experts, are not equivalent to vanilla, yet they also land between 1.9262 and 1.9411, around vanilla's 1.9317. The strongest fixed pattern, a routed shared core, improves on vanilla by 0.0055 at 182M and 0.0167 at 469M, less than half of UNIPOOL's gain; a DeepSeekMoE-style shared expert under the 16E/top-2 budget behaves similarly (Appendix Table 9).

Locking the learned allocation recovers the full-pool gain. Control C trains with the full pool, fixes the learned allocation at 1K, attenuates the other experts, and locks it at 2K, exactly as UNIPOOL$_{\text{lock}}$ does. It improves on A by 0.0279 and matches UNIPOOL$_{\text{full}}$ (1.9029): brief full-pool training followed by the lock recovers the gain of persistent full-pool training. Control B isolates the allocation itself by following the same schedule but locking A's random allocation instead; it

Table 4: 182M test loss (8E/top-1) by when ownership is decided. A and B share one random allocation; C and UNIPOOL$_{\text{lock}}$ are two seeds of the same recipe. All controls: Appendix Table 8.

Run	Loss ↓
Fixed at initialization	
Vanilla MoE	1.9317
A: random disjoint	1.9319
Routed shared core	1.9262
ReXMoE-R2 (adjacent reuse)	1.9325
MoUE-stateless (windowed)	1.9411
Global access, then locked	
B: random allocation	1.9173
C: learned allocation	1.9040
UNIPOOL$_{\text{lock}}$	**1.9017**
Never fixed	
UNIPOOL$_{\text{full}}$	1.9029

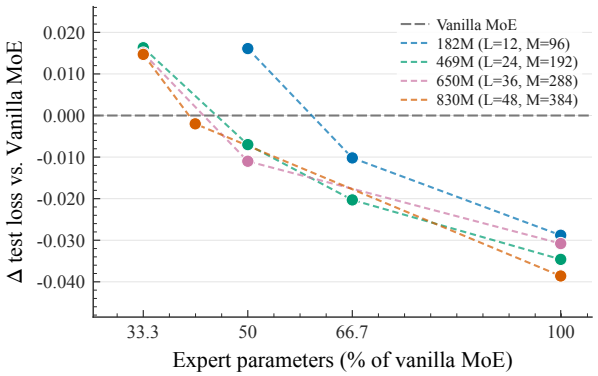


Figure 2: Reduced pools: change in final test loss of UNIPOOL$_{\text{full}}$ relative to vanilla MoE with eight experts per layer (vertical axis) versus pool size as a fraction of vanilla's expert parameters, at unchanged top-1 active compute.

Table 5: Ablations of UNIPOOL$_{\text{full}}$ at 182M. Δ is relative to vanilla MoE; G: number of expert pools.

Configuration	Loss ↓	Δ
Components		
Vanilla MoE + softmax (G=12)	1.9317	–
Vanilla MoE + NormRouter	1.9375	+0.0058
Vanilla MoE, sigmoid, aux-free	1.9239	−0.0078
Shared + layer aux + softmax	1.9480	+0.0163
Shared + pool aux + softmax	1.9180	−0.0137
UNIPOOL$_{\text{full}}$ (G=1)	**1.9029**	−0.0288
Sharing scope		
G=6	1.9121	−0.0196
G=4	1.9099	−0.0218
G=2	1.9213	−0.0104

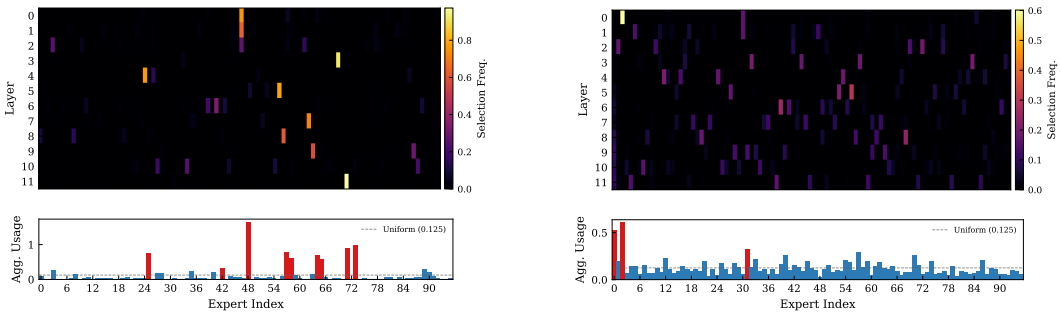
Table 6: Step-time overhead of UNIPOOL$_{\text{lock}}$ relative to vanilla MoE before the lock (*Full*), after it (*Compact*), and end-to-end over 60K steps (*E2E*), on 8 GPUs. Raw step times: Table 14.

Setting	Full	Compact	E2E
182M, 8E	+22%	+2.8%	+3.5%
182M, 16E	+25%	+1.3%	+2.1%
182M, 32E	+36%	−0.7%	+0.6%
469M, 8E	+33%	+2.5%	+3.5%

improves on A by 0.0146, so discarding the learned allocation at the lock forfeits about half of the gain, and retaining the learned allocation provides the remaining improvement. Since B and C differ only in the locked allocation, their gap isolates the learned allocation's value given the shared phase, not the phase's effect on the expert weights themselves, which a matched warm-start control would test. C and the reported UNIPOOL$_{\text{lock}}$ run are two seeds of the same recipe (1.9040 and 1.9017), whose 0.0023 spread matches the seed standard deviation at 182M. Consistent with the small gap between UNIPOOL$_{\text{lock}}$ and UNIPOOL$_{\text{full}}$, persistent cross-layer reuse in UNIPOOL$_{\text{full}}$ is limited: a token touches 94.1% distinct experts across 12 layers, 89.5% across 24, and 83.4% across 36, against 94.5%, 94.2%, and 94.2% under independent uniform routing, so reuse beyond chance appears only in the deeper models (Appendix I).

5.4 WHAT THE FULL POOL REVEALS

Reduced pools improve parameter efficiency. Figure 2 is a UNIPOOL$_{\text{full}}$ parameter-efficiency study: it shrinks the shared pool below $8L$ while keeping one active expert per layer and token. Pools with 66.7% of vanilla's expert parameters outperform vanilla at 182M (1.9215 vs. 1.9317), and pools with 50% outperform it at 469M and 650M by 0.007 and 0.011. At 182M, we directly compare the 50% shared pool (M=48) with a parameter-matched private baseline using four private experts per layer: the shared pool reaches 1.9479 versus 1.9664 for the private baseline under the same global batch size, data, schedule, and seed. At 830M a pool with 41.7% of the expert parameters matches vanilla to within 0.002. Across these jointly scaled configurations, the smallest competitive fraction decreases with depth.



(a) Softmax + per-layer aux loss

(b) UNIPOOL$_{\text{full}}$: NormRouter + pool aux loss

Figure 3: Expert usage in a 182M shared pool, per layer (top) and aggregated (bottom). Bottom panels show aggregate usage summed across layers (uniform $L/M = 1/E = 0.125$; red bars exceed it). (a) Per-layer balancing collapses aggregate usage onto a few experts. (b) The pool loss with NormRouter balances aggregate usage while each layer concentrates on its own subset of experts.

More experts and looped execution. UNIPOOL$_{\text{full}}$ improves over vanilla MoE at 8E/top-1, 16E/top-2, and 32E/top-4 on the 182M backbone (Table 2), so the gain persists when each layer stores and activates more full-width experts. With two recurrent expert steps per layer, UNIPOOL$_{\text{full}}$ reduces loss from 1.9077 to 1.8727 at 182M and from 1.7768 to 1.7389 at 469M (Appendix D). In this looped variant, attention runs once per layer and the routed expert sublayer is applied twice with the same router and experts, so expert parameters and per-token expert compute stay matched.

Component ablations. Table 5 decomposes UNIPOOL$_{\text{full}}$ at 182M. Sharing the pool with per-layer balancing and softmax routing is worse than vanilla (1.9480 vs. 1.9317) and collapses aggregate usage onto a few experts (Figure 3a). Replacing it with the pool loss improves loss to 1.9180, and NormRouter improves it further to 1.9029. With both, aggregate usage is balanced while each layer concentrates on its own subset of the pool (Figure 3a–b), the layer–expert preference that the lock later turns into ownership. NormRouter does not help private experts (1.9375), so its benefit depends on the shared candidate set. UNIPOOL$_{\text{full}}$ also beats the stronger private sigmoid-gating, auxiliary-loss-free baseline at 182M (1.9239) by 0.0210. Grouping layers into G pools interpolates between private (G=12) and global (G=1) sharing: every intermediate G beats vanilla and the global pool is best.

Learned ownership makes experts less substitutable. Repeating the probe of Section 3 on our models (Table 7), randomizing one deep-half layer costs vanilla MoE 1.3 points at 469M, matching the production models. We match the randomized choice-set sizes across models; the full protocol is given in Appendix J. The same probe costs UNIPOOL$_{\text{lock}}$ 3.3 points, although each of its layers holds exactly as many experts as a vanilla layer and the random choice is drawn from that locked allocation. It costs UNIPOOL$_{\text{full}}$ 4.1 points when sampling from each layer's eight most-used experts, and a similar 3.8 when sampling from the entire pool. With matched choice-set sizes, the larger drop is consistent with stronger specialization under global competition; because gates are re-normalized at intervention, the probe is a sensitivity test rather than a causal test of ownership.

Table 7: Single-layer routing randomization on our 469M models: deep-half average over five tasks, length-normalized accuracy (so TOP-K differs from Table 3). †: sampled from each layer's eight most-used experts.

Model	TOP-K	RANDOM	Δ
Vanilla MoE	45.10	43.83	-1.3
UNIPOOL$_{\text{lock}}$	47.05	43.77	-3.3
UNIPOOL$_{\text{full}}$	47.16	43.10†	-4.1

6 CONCLUSION

UNIPOOL$_{\text{lock}}$ learns which experts each layer owns through brief global access, then locks this allocation and trains a layer-private MoE. Across five scales from 182M to 1.5B, the learned allocation improves over vanilla MoE and recovers nearly all of the gain of persistent full-pool training, with post-lock step time within $+2.8\%$ of vanilla (Table 6). The persistent full-pool reference also beats vanilla at 50%–66.7% of its expert-parameter budget. Appendix B discusses limitations.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript language editing and LaTeX format conversion. They were not used to generate experimental results or alter empirical data. The authors reviewed the AI-assisted text and formatting, verified the manuscript against the underlying experiments and source material, and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 5.1 describes the experimental setup, and Appendices C and F report model configurations and training hyperparameters. Appendix G specifies the ownership-learning schedule, assignment, and compact-dispatch invariants, and Appendix H specifies the ownership controls. Appendices A, D, J, K, and L cover the sharing baselines, looped experts, evaluation protocols, pool-loss implementation, and NormRouter initialization.

REFERENCES

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *International Conference on Learning Representations*, 2025a.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 107088–107133, 2025b.

Sikai Bai, Haoxi Li, Jie Zhang, Zicong Hong, and Song Guo. DiEP: Adaptive mixture-of-experts compression through differentiable expert pruning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 62474–62499, 2025.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Yilong Chen, Naibin Gu, Junyuan Shang, Zhenyu Zhang, Yuchen Feng, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Mixture of universal experts: Scaling virtual width via depth-width transformation. *arXiv preprint arXiv:2603.04971*, 2026. URL https://arxiv.org/abs/2603.04971.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George Bm van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew J. Johnson, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Marc'Aurelio Ranzato, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4057–4086. PMLR, 2022.

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. *arXiv preprint arXiv:1803.05457*, 2018.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. MoEUT: Mixture-of-experts universal transformers. In *Advances in Neural Information Processing Systems*, 2024.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

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

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

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, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Zhuocheng Gong, Ang Lv, Jian Guan, Wei Wu, Huishuai Zhang, Minlie Huang, Dongyan Zhao, and Rui Yan. Mixture-of-modules: Reinventing transformers as dynamic assemblies of modules. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20924–20938, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1164.

Xu Owen He. Mixture of a million experts. *arXiv preprint arXiv:2407.04153*, 2024.

Ruijun Huang, Fang Dong, Xin Zhang, Anrui Chen, Hengjie Cao, Zhendong Huang, Jixian Zhou, Mengyi Chen, Yifeng Yang, Mingzhi Dong, Yujiang Wang, Jinlong Hou, Qin Lv, Robert P. Dick, Yuan Cheng, Fan Yang, Tun Lu, Chun Zhang, and Li Shang. SD-MoE: Spectral decomposition for effective expert specialization. In *International Conference on Machine Learning*, 2026.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, HoYuen Chau, Peng Cheng, Fan Yang, Mao Yang, and Yongqiang Xiong. Tutel: Adaptive mixture-of-experts at scale. In *Proceedings of Machine Learning and Systems*, volume 5, 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*, 2024.

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.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, 2017.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

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.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling laws for fine-grained mixture of experts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33270–33288. PMLR, 2024.

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.

Xiaonan Nie, Xupeng Miao, Shijie Cao, Lingxiao Ma, Qibin Liu, Jilong Xue, Youshan Miao, Yi Liu, Zhi Yang, and Bin Cui. EvoMoE: An evolutional mixture-of-experts training framework via dense-to-sparse gate. *arXiv preprint arXiv:2112.14397*, 2021.

Bowen Pan, Yikang Shen, Haokun Liu, Mayank Mishra, Gaoyuan Zhang, Aude Oliva, Colin Raffel, and Rameswar Panda. Dense training, sparse inference: Rethinking training of mixture-of-experts language models. *arXiv preprint arXiv:2404.05567*, 2024.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144.

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 *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18332–18346. PMLR, 2022.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34, 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.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Zheyue Tan, Zhiyuan Li, Tao Yuan, Dong Zhou, Weilin Liu, Yueqing Zhuang, Yadong Li, Guowei Niu, Cheng Qin, Zhuyu Yao, Congyi Liu, Haiyang Xu, Boxun Li, Guohao Dai, Bo Zhao, and Yu Wang. ReXMoE: Reusing experts with minimal overhead in mixture-of-experts. *arXiv preprint arXiv:2510.17483*, 2025. URL https://arxiv.org/abs/2510.17483.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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.

Ziteng Wang, Jun Zhu, and Jianfei Chen. ReMoE: Fully differentiable mixture-of-experts with ReLU routing. In *International Conference on Learning Representations*, 2025.

Juntong Wu, Jialiang Cheng, Fuyu Lv, Ou Dan, and Li Yuan. SERE: Similarity-based expert re-routing for efficient batch decoding in MoE models. In *International Conference on Learning Representations*, 2026.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. OpenMoE: An early effort on open mixture-of-experts language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 55625–55655. PMLR, 2024.

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*, pp. 4791–4800, 2019.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Chuanyang Zheng, Jiankai Sun, Yihang Gao, Enze Xie, Yuehao Wang, Peihao Wang, Ting Xu, Matthew Chang, Liliang Ren, Jingyao Li, Jing Xiong, Kashif Rasul, Mac Schwager, Anderson Schneider, Zhangyang Wang, and Yuriy Nevmyvaka. Understanding the mixture-of-experts with nadaraya-watson kernel. In *International Conference on Learning Representations*, 2026.

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. *Advances in Neural Information Processing Systems*, 35, 2022.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15913–15923, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.890. URL https://aclanthology.org/2024.emnlp-main.890/.

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.