MIXTURE OF LAYERS WITH HYBRID ATTENTION: PARALLEL THIN BLOCKS FOR SPARSE TRANSFORMER COMPUTE

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose parallel attention processing with a new block-decomposed model, Mixture of Layers (MoL). Comparing with MoE models, the routing in MoL operates with parallel attention blocks instead of experts inside one block. MoL uses K parallel thin blocks of width $d_{\text{thin}} \ll d_{\text{model}}$, connected to the full block width d_{model} by learned down and up projections, and a top-k routing that selects which of these blocks are processing which token. Sparse routing over multiple blocks degrade attention quality, as each block is processing fewer tokens. To address this, we introduce a hybrid attention design, with one shared block processing all input tokens using dense softmax attention (holds global context), and Gated DeltaNet linear attention in the routed blocks handling sparse token subsets. We scaled MoL to 2.08B (0.63B active) on FineWeb-Edu and found that the model outperforms an iso-active dense baseline by 0.17 PPL, while it underperforms a 1.3B dense baseline by 3.21 PPL and trains about $2\times$ slower. Evaluation on 8-task lm-eval-harness shows scores following PPL results, and scores model between the two dense baselines. We tested single-GPU prefill on A100, H100 SXM and H200 GPUs. At 128K context it is $1.2\times$ faster than the 1.3B dense model on A100, and up to $1.5\times$ faster at 256K tokens on H100 SXM and H200, while short-context prefill and decode remain slower.

1 INTRODUCTION

The standard way to scale a language model is to stack more sequential transformer blocks, which buys depth at the cost of memory and latency. Mixture-of-Experts (MoE) transformers (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Zoph et al., 2022) decouple compute from parameter count by routing tokens to expert subnetworks. The layer itself, however, stays monolithic: every position still runs one full-width block.

Hybrid models pair two attention mechanisms in one network, most often across layers: Qwen3-Next (Qwen Team, 2025) interleaves Gated DeltaNet with gated attention at a 3:1 ratio, and Zamba (Glorioso et al., 2024) connects an SSM backbone with periodic attention, so each layer still runs one mechanism and the mixture appears only in the average over depth. Hymba (Dong et al., 2025) is the exception, running attention and SSM heads side by side inside a layer. To place both mechanisms within the same layer, the layer require to have parts.

This paper asks whether the layer itself can be decomposed into parallel thin blocks routed like experts, and what that costs in quality and speed against dense and MoE-FFN models. Our contributions:

1. **Mixture of Layers (MoL).** We replace each full-width transformer block with K parallel thin blocks of width $d_{\text{thin}}{=}d_{\text{model}}/4$, wired back to the full width with learned down/up projections and mixed by top-k block routing (Section 2). The router picks attention as well as the FFN. A gather/scatter sparse dispatch runs each block only on its routed tokens, up to $4.34\times$ faster than dense-masked routing on the forward pass at 10% active blocks.
2. **Hybrid attention for sparse routing.** Routing over more blocks sparsely makes each block to see fewer tokens, and attention quality degrades because of that. So the routed blocks use Gated DeltaNet (Yang et al., 2025a), an $O(T)$ linear attention, on their sparse token subsets, and one

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.

shared block runs full softmax attention for every token. At d_{thin}=256, DeltaNet in the routed blocks beats softmax by 0.85 PPL at matched parameters (3 seeds); in single-block runs the edge disappears once d_{thin}=d_{model} (Section 3).

3. **A controlled comparison over three corpora and two scales, with mixed results.** MoL leads a dense softmax model of equal total size on single-epoch Cosmopedia v2 (0.175 PPL, softmax in every block) and of matched active size on FineWeb-Edu (0.77 for the widest topology at 80M, 0.17 at 2.08B total / 0.63B active), but trails dense on multi-epoch WikiText-103 (0.51), a full-rank MoE-FFN of the same total and active size at 80M (0.81) and a dense 1.3B model (3.21) (Sections 5.3 to 5.5). It trains 1.9–2.3× slower at billion scale and decodes slower. Its one efficiency gain is single-GPU prefill at 64K–256K tokens, depending on the GPU (Section 5.6). Beyond WikiText-103, each MoL result is a single seed.

2 The MoL Architecture

2.1 Thin Block Wrapper

MoL replaces selected full-width transformer blocks with **split stages**, each containing K independent thin blocks. Each thin block consists of a down-projection, a standard transformer block at reduced dimensionality, and an up-projection.

$$\text{ThinBlock}(x) = W_{\text{up}} \cdot \big(\text{Block}_{d_{\text{thin}}}(W_{\text{down}} \cdot x) - W_{\text{down}} \cdot x\big) \tag{1}$$

where $W_{\text{down}} \in \mathbb{R}^{d_{\text{thin}} \times d_{\text{model}}}$, $W_{\text{up}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{thin}}}$, and $\text{Block}_{d_{\text{thin}}}$ is a complete transformer block (RMSNorm → multi-head attention → RMSNorm → FFN) operating at dimensionality d_{thin}. The subtraction removes the inner residual; ThinBlock outputs only the delta the block computes.

2.2 Split Stage with Block Routing

A split stage runs K thin blocks, selects k via a learned router, and takes a router-weighted average of their outputs:

$$\text{SplitStage}(x) = x + \sum_{i \in \text{top-}k(K)} w_i \cdot \text{ThinBlock}_i(x) \tag{2}$$

where top-k selects the k highest-scoring blocks. The router is a bias-free linear map from the split-stage input x (at d_{model}) to one logit per candidate block. The weights w are the softmax of these logits, the k largest entries are kept and renormalised to sum to one, and no noise is added during training. The outer residual $x + \ldots$ preserves gradient flow even when the thin blocks produce small updates. On WikiText-103, selective block activation outperforms uniform composition by 0.99 PPL (Table 1; 3 seeds, 8.7× the pooled seed standard deviation of 0.11).

Load balancing. We use a squared coefficient-of-variation (CV^2) loss, weighted by α=0.05, on the per-block softmax weights averaged over the batch and the sequence (the continuous load, not the discrete top-k counts), following standard MoE practice. It prevents block collapse without requiring auxiliary tokens or capacity constraints. How the per-layer terms are reduced differs between settings (Appendix E).

2.3 Sparse Dispatch

With block-level routing, each thin block processes only the tokens routed to it. *Sparse dispatch* exploits this: it **gathers** a block's routed tokens, runs the block on the resulting compact tensors, and **scatters** the outputs back, so a block's cost follows the tokens it receives, not the sequence length. Both implementations pad every routed block to the token count of the busiest block, which ties the cost of a split stage to its busiest block rather than to the mean load (Section 5.6). The runs of Table 1 and MoL K=5 top-3 in Table 3 do not use sparse dispatch: in these *full-attention* runs every thin block processes every token and the router, where present, only selects which block outputs are mixed, so they are not sparse in compute. Every S+$k$of$N$ or "sparse" run uses sparse dispatch.

Correctness. For softmax thin blocks, sparse dispatch produces numerically identical outputs to dense restricted attention (max logit diff 5.7×10^{-6} on GPU, 2.4×10^{-7} on CPU). The DeltaNet case is discussed in Section 3.2.

Speedup. Measured on the forward pass of 8-layer models (B=16, T=512) against the same model with restricted (dense-masked) attention, the speedup scales with sparsity, from 1.53× at 57% active to 4.34× at 10% (4.94× with `torch.compile`; Appendix E). Each point is a different, freshly initialised model (106M to 243M total) whose near-uniform routing is the best case for padding. End-to-end training is slower than dense at every scale we measured (Appendix G).

3 Hybrid Attention

3.1 The Attention Coverage Problem

Scaling to more blocks with sparse dispatch raises an attention coverage problem: at 3-of-15, each block sees only 20% of the sequence, compared to 60% at 3-of-5, and the block-averaged spacing between the positions it does receive rises from 1.8 to 25.6 (blocks that receive fewer than two tokens count as one gap of the full sequence length). A softmax-only sparse 3-of-15 configuration achieves PPL 35.34 on WikiText-103, worse than 3-of-5 (32.04 at the same seed) despite 2.2× more parameters.

3.2 Architecture: Shared + Routed

Each split stage contains:

- **Shared block** (block 0): always active on all tokens, full softmax attention, providing global context at every layer (at T=255 removing it costs no quality; Appendix I).
- **Routed blocks** (blocks 1–$N-1$): top-k selected by the router, sparse dispatch with Gated DeltaNet (Yang et al., 2025a) linear attention on their token subsets.

We denote configurations as $S{+}k\text{of}N$: S shared blocks plus top-k routed from $N-S$ candidates, with N total blocks (the pool size K of Section 2). For example, 1+3of15 has 1 shared softmax block and selects 3 from 14 routed DeltaNet blocks, activating 4 total.

Combination rule. The shared block does not pass through the router. The router and the CV^2 loss cover only the $N-S$ routed candidates. With S=1 the split stage becomes

$$\text{SplitStage}(x) = x + w_s \cdot \text{ThinBlock}_0(x) + \sum_{i \in \text{top-}k(N-1)} w'_i \cdot \text{ThinBlock}_i(x), \tag{3}$$

where w'_i is the renormalised top-k routing weight w_i of Equation (2), used as is or rescaled, and w_s is a fixed scalar, not produced by the router. Two settings of (w_s, w'_i) are used across the experiments. A routed DeltaNet block runs over its gathered tokens as one contiguous sequence, with no correction for the original inter-token gaps. Appendix E gives both settings and the details.

3.3 Why DeltaNet in Routed Blocks

With the shared block carrying the global softmax context, the routed blocks need no quadratic attention of their own. The d_{thin} sweep below also favours DeltaNet at d_{thin}=256, and DeltaNet is $O(T)$ rather than $O(T^2)$, faster than softmax in a thin block from $T \geq 2048$ (Appendix G).

Decomposing the DeltaNet advantage. On WikiText-103 at K=1 (single block, no routing, all tokens visible), DeltaNet outperforms softmax by 2.55 PPL (36.28 vs 38.83) at matched architecture (∼43M params). An iso-parameter softmax control (87.5M, d_{ff}=1133, 3 seeds: 32.09 ± 0.26) matches DeltaNet's extra parameters (gates, convolutions) with a wider FFN and closes 0.26 PPL (23%) of the 1.11 PPL gap in the 1+2of5 MoL configuration (Tables 2 and 12). The remaining 0.85 PPL (77%) persists at matched parameter count, 3.0× the pooled seed std (σ_{pooled}=0.29).

Scale dependence. A single-block (K=1) sweep across $d_{\text{thin}} \in \{128, 256, 512, 1024\}$ shows the DeltaNet–softmax gap shrinking monotonically, from 2.79 → 2.55 → 1.12 → −0.11 PPL. At d_{thin}=d_{model}=1024, softmax marginally wins (27.60 vs 27.70). DeltaNet's inductive biases (decay, delta rule) seem to matter most when attention capacity is tight. Each width is a single run.

4 EXPERIMENTAL SETUP

4.1 DATASETS

We use three datasets of increasing size: **WikiText-103** (Merity et al., 2017): 103M tokens, custom BPE 32K, T=255, multi-epoch (∼8 epochs); architecture ablations at 36–198M params. **Cosmopedia v2** (Ben Allal et al., 2024a;b): first 15B synthetic-textbook tokens of the corpus (streamed, untruncated order), GPT-2 vocab, T=2048, single-epoch budget; data-regime dependence at ∼104M params. **FineWeb-Edu** (Penedo et al., 2024): GPT-2 vocab, single-epoch. 80M topology sweep at T=2048 (∼3.9B tokens at 60K steps); billion-scale runs at T=4096 (20B tokens).

4.2 ARCHITECTURES

80M-scale ablations (WikiText-103, Cosmopedia): d_{model}=1024, 16 heads, 8 layers, d_{thin}=256, 4 heads per thin block (d_{head}=64; one RoPE table serves all widths). Training: 50K steps (WikiText) or 114K steps (Cosmopedia), lr = 3×10^{-4}, cosine decay, 1000-step warmup (WikiText) / 2000-step (Cosmopedia), AdamW (β_1=0.9, β_2=0.95, wd= 0.01), gradient clipping at 1.0, effective batch size 64. The WikiText-103 dense baselines also differ from the MoL runs in normalization, activation and learning-rate floor (Appendix K).

Billion-scale headline (FineWeb-Edu 20B tokens, 4× H200 NVL). Three architectures share d_{model}=2048, 24 layers, GPT-2 50,257-vocab, T=4096, batch 1.0M tok/step, 20K steps, AdamW lr = 2×10^{-4} (β_1=0.9, β_2=0.95, wd= 0.1 on decay parameters), cosine decay clipped at 0.1×, 1000-step warmup, gradient clipping at 1.0, bf16 with gradient checkpointing. **Dense Softmax 1.3B**: d_{ff}=8192, 16 heads (d_{head}=128). 1.31B total. **Dense Softmax 0.7B**: d_{ff}=2048 matched to a single MoL thin block, 32 heads (d_{head}=64 matched to thin-block heads). 0.71B total. **MoL Hybrid 1+3of15**: d_{thin}=512, 15 thin blocks (1 shared softmax + 14 routed DeltaNet, top-3 active = 4 active per token), $d_{\text{ff,thin}}$=2048. 2.08B total / 0.63B active.

Protocol notes. The three billion-scale runs add a logit z-loss, $10^{-4}\,\mathbb{E}[(\log\sum_v e^{z_v})^2]$, to the cross-entropy. The 80M-scale runs do not. Perplexities are each run's best validation checkpoint, usually the final step and elsewhere within 0.38 PPL of it. The MoE-FFN control of Section 5.1 is the one exception, quoted at its final step. The 80M FineWeb-Edu sweep was extended from a 38K-step schedule to 60K by resuming a mid-run checkpoint, which raises the learning rate and repeats data at the resume point, so Table 4 mixes extended and clean-schedule rows (Appendix J).

4.3 BASELINES

Dense baselines match MoL's total (iso-param, or iso-total) or active (iso-active) parameter count.

- **Dense Softmax**: standard transformer, iso-param to MoL conditions. At 85M: PPL 30.26 (WikiText). At 103.5M: PPL 6.65 (Cosmopedia). At 198M: PPL 26.89 (WikiText).
- **Dense DeltaNet**: DeltaNet attention, iso-param, which requires d_{ff}=588 against the softmax baseline's 1120 to offset DeltaNet's gate and convolution weights. At 85M: PPL 30.68 (WikiText). At 103.4M: PPL 6.64 (Cosmopedia). The per-run Cosmopedia numbers come from validation subsets of unequal size (200 sequences for the controls, 400 for the MoL and dense-softmax runs). Section 5.3 rescores every condition on one held-out set.
- **Rank-1 MoE baseline**: full-width MoE with 1,024 rank-1 experts per layer (one down and one up vector each), 84.7M params, PPL 33.93 (WikiText).
- **MoE-FFN controls**: full-width attention with routed full-rank FFN experts and no block routing. They test whether FFN routing alone reproduces MoL's effect (Sections 5.1, 5.3 and 5.4).

5 RESULTS

Sections 5.1 and 5.2 fix the block design on WikiText-103, where the key conditions have three seeds. Sections 5.3 and 5.4 test it on single-epoch corpora against dense and MoE-FFN baselines, and Sections 5.5 and 5.6 report the billion-scale comparison (Table 5) and its cost.

Table 1: Dense FFN thin blocks vs rank-1 MoE on WikiText-103, 50K steps. The MoL rows use d_{thin}=256; the rank-1 MoE baseline is a full-width (d_{model}=1024) rank-1 MoE. It scored 800 of the 845 validation sequences and the dense-FFN rows all 845; for the re-scored models the last 45 are harder (Appendix K); the baseline was not re-scored.

Config	FFN	Params	PPL	Δ vs rank-1 MoE
Rank-1 MoE baseline	MoE	84.7M	33.93	—
MoL K=7 split_all (rank-1)	MoE	85.6M	34.83	+0.90
MoL K=5 top-3 (dense)	Dense	85.3M	$\mathbf{30.95 \pm 0.11}$	$\mathbf{-2.98}$
MoL K=5 all-active (dense)	Dense	85.3M	31.94 ± 0.12	-1.99

Table 2: Hybrid attention on WikiText-103. Among the sparse rows, routed DeltaNet beats routed softmax at both pool sizes. S=shared (all tokens), R=routed (sparse). †=1 seed; ±=3 seeds. Effective batch 64 throughout, at micro-batch 16, 32 or 64, a split worth up to 0.3 PPL for MoL (Appendix C). *Re-scored on the full 845-sequence split (Appendix K).

Config	Attn	Active	Params	PPL
MoL K=5 top-3, full attn.	softmax	3/5	85.3M	30.95 ± 0.11
Sparse 3-of-5	softmax	3/5	85.3M	32.19 ± 0.15
Softmax 1+2of5 iso-param	softmax	1S+2R	87.5M	32.09 ± 0.26
DeltaNet 1+2of5	S:smx, R:Δ	1S+2R	87.5M	$\mathbf{31.24 \pm 0.31}$
Softmax 1+2of15†*	softmax	1S+2R	190.4M	37.17
DeltaNet 1+2of15†*	S:smx, R:Δ	1S+2R	198.2M	33.32
DeltaNet 1+3of15*	S:smx, R:Δ	1S+3R	198.2M	$\mathbf{30.53 \pm 0.06}$
Dense softmax 85M†	softmax	—	84.7M	30.26
Dense softmax 198M†	softmax	—	198.2M	26.89
Dense DeltaNet 85M†	DeltaNet	—	84.7M	30.68

5.1 Dense-FFN Thin Blocks on WikiText-103

Replacing the rank-1 MoE FFNs in the thin blocks with standard dense FFNs ($d_{\text{ff}} = 4 \times d_{\text{thin}}$) far exceeds the quality of rank-1 experts (Table 1, Appendix A), and the MoL runs in the rest of this section use dense-FFN thin blocks. The dense-FFN MoL (K=5, top-3, 85.3M params) achieves PPL 30.95 ± 0.11 (3 seeds), 2.98 better than the full-width rank-1 baseline (33.93). This gain reflects the recovery from rank-1 factorisation rather than superiority over sparse MoE in general, and part of it needs no dense FFN: rank-1 thin blocks with learned top-6-of-7 routing reach 32.49 (one seed). A full-rank MoE-FFN control on WikiText-103 (five FFN experts of width 256, top-3, 87.4M, shared full-width attention) ends at 31.73 at 50K steps (best 31.49), so dense-FFN thin blocks are 0.79 PPL better than routing full-rank FFN experts at this size, while the iso-parameter dense softmax model (30.44 at matched micro-batch) stays ahead of both. All three numbers are final-step values.

5.2 Hybrid Attention Results

In Table 2, DeltaNet 1+2of5 (87.5M, 31.24 ± 0.31, 3 seeds) beats both the iso-param Softmax 1+2of5 (32.09 ± 0.26) and the standard one (32.35 ± 0.14). At 15 blocks, DeltaNet 1+3of15 (198M total, 77M active) reaches PPL $\mathbf{30.53 \pm 0.06}$ (3 seeds) at 27% of blocks per token, 0.42 PPL better than the five-block dense-FFN MoL (Table 1).

Dense baselines. At iso-total parameters (198M), a dense softmax transformer reaches PPL 26.89, 3.64 PPL ahead of DeltaNet 1+3of15 on the same validation split. The gap mixes the price of sparse block routing with implementation differences between the two models (Appendix K). In exchange, MoL runs only 77M active parameters per token (39% of total), a 2.6× ratio of total to active parameters. That ratio is a nominal compute figure, not a measured FLOP count: sparse dispatch pads every routed block to the busiest block's load, and the routed and shared blocks use different attention mechanisms. Blocks can also be placed on separate devices (Appendix H).

Table 3: Cosmopedia v2 results (first 15B tokens of the corpus, ∼104M params). "Logged" uses a validation split cut from the training token stream and may overlap training data; "held-out" scores a later shard that no run reached and is the comparison we use. Resumed runs restarted their data order at different points, so exposure differs across conditions (Appendix J).

Condition	Params	PPL (logged)	PPL (held-out)	vs Dense
Dense softmax	103.5M	6.65	7.110	—
Dense DeltaNet	103.4M	6.64	7.014	−0.096
MoL K=5 top-3	104.0M	6.49	**6.936**	**−0.175**
Sparse MoL 1+2of5	106.2M	6.52	6.985	−0.126
MoE-FFN top-3-of-5	∼106M	6.89	7.295	+0.185

5.3 Data-Regime Scaling: Cosmopedia 15B

On WikiText-103 (103M tokens, ∼8 epochs), MoL trails the dense baseline: K=5 top-3 (85.3M, 30.95 ± 0.11, micro-batch 64) is 0.51 PPL behind dense softmax at the same micro-batch (30.44). A single-epoch budget over the first 15B tokens of Cosmopedia v2 flips this result (Table 3).

On the held-out shard MoL K=5 finishes 0.175 PPL below the dense baseline, a 0.69 PPL swing relative to WikiText-103, where the same comparison at matched micro-batch leaves MoL 0.51 PPL behind. On the logged curves it overtakes dense at step 40K. The sparse MoL variant (routed DeltaNet + shared softmax) stays within 0.05 PPL of the full-attention K=5 run on the held-out set and retains 72% of its gain under sparse dispatch.

Isolating the structural contribution. MoL K=5 top-3 runs softmax attention in every block, so none of its 0.175 PPL margin over dense softmax comes from the attention mechanism. The two implementations also differ in normalization, activation and FFN bias, and the margin carries those differences along with the structural change. The dense DeltaNet control is built from MoL's block implementation (with the FFN halved to hold parameters fixed) and is 0.096 PPL *ahead* of dense softmax, leaving MoL K=5 0.079 ahead of it on the one comparison that shares a block implementation. The two routes do not add up: the sparse variant pairing a shared softmax block with routed DeltaNet reaches 0.126, short of the full-attention K=5 run. The MoE-FFN control is 0.185 PPL *behind* dense softmax, so FFN-only routing does not reproduce the gain.

5.4 FineWeb-Edu 80M Ablations

To see whether the MoL advantage on Cosmopedia (Section 5.3) holds on web text, we ran an 80M sweep on FineWeb-Edu, 60K steps (T=2048, ∼3.9B tokens, one seed per MoL topology): three MoL Hybrid topologies (1+3of{5, 10, 15}), three Dense Softmax baselines of different active size and a full-rank sparse MoE-FFN built to MoL's total and active size (Table 4). The lead holds against dense models once schedules are matched, but not against a sparse MoE-FFN. On clean schedules and one common validation set, the widest topology, 1+3of15, finishes at 25.17, 0.77 PPL ahead of a standard-shaped iso-active dense model (25.95 ± 0.04 over three seeds; MoL's own seed spread was not measured on this corpus) and 0.62 ahead of the narrowest, 1+3of5, which is itself 0.16 ahead of that baseline. A full-rank MoE-FFN of the same total and active size finishes 0.81 PPL ahead of 1+3of15. The five-expert MoE-FFN controls matched to five-block MoL on WikiText-103 and Cosmopedia trail it instead (Sections 5.1 and 5.3): the ordering changes with pool size and corpus. The middle topology, 1+3of10, was not retrained (Appendix J).

5.5 Billion-Scale Results

At billion scale MoL keeps a small lead over an iso-active dense model but stays 3.21 PPL behind a dense model with twice its active parameters, at about twice the training time (Table 5). All three runs use FineWeb-Edu 20B tokens and one seed each. We read them as a scaling probe.

Iso-active comparison. On the closest active-parameter match and the matched validation protocol, MoL Hybrid (0.63B active) reaches 17.92 PPL against 18.09 for Dense Softmax 0.7B (0.71B active, d_{ff} matched to a single thin block), a 0.17 PPL gap (0.49 on the unmatched logged values). On

Table 4: FineWeb-Edu 80M sweep at 60K steps, every checkpoint scored on the same first 2,048 validation sequences (Appendix K). One seed per row except Dense 95M and MoE-FFN (three seeds each), given as mean ± sample standard deviation. MoL rows use shared softmax and routed Gated DeltaNet at d_{thin}=256. Δ is against the Dense 95M mean. Negative favours the row.

Condition	Total	Active	d_{ff}	Val PPL @ 60K	Δ PPL
Dense Softmax 152M (larger active budget)†	152M	152M	4096	**23.50**	−2.45
Dense Softmax 95M (iso-active)	95.3M	95.3M	3072	$\mathbf{25.95 \pm 0.04}$	–
Dense Softmax 98M (iso-active, narrow FFN)†	97.7M	97.7M	768	27.59	+1.64
MoE-FFN top-2 (full-rank, iso-total)	217M	97.8M	384¶	$\mathbf{24.36 \pm 0.07}$	−1.59
MoL Hybrid 1+3of15†	217M	95M	1024§	25.17	−0.77
MoL Hybrid 1+3of10	162M	95M	1024§	26.54	+0.60
MoL Hybrid 1+3of5†	106M	95M	1024§	25.79	−0.16

Dense 95M: d_{model}=768, d_{ff}=3072; Dense 98M: d_{model}=1024, 0.75× expansion. Every row sees 3.93B tokens at 32 sequences per step, as micro-batch 2×16 except 1+3of15 at 4×8. §Per thin block, 4 active. ¶Per expert, 21 experts, top-2. 1+3of10 carries the schedule extension of Appendix J; †clean-schedule retrains of sweep runs that read 24.01, 28.23, 25.42 and 26.67 on the same set.

Table 5: Billion-scale validation perplexity on FineWeb-Edu 20B tokens (4× H200 NVL, identical optimizer and token budget). Wall time is the full 20K-step run.

Model	Total	Active	d_{ff}	PPL (logged)	PPL (matched)	Wall time
Dense Softmax 1.3B	1.31B	1.31B	8192	15.03	**14.72**	53.3h
Dense Softmax 0.7B	0.71B	0.71B	2048	18.53	18.09	44.5h
MoL Hybrid 1+3of15	2.08B	0.63B	2048†	18.04	**17.92**	102.0h

† Per thin block; 4 active blocks per token. "Logged" is what each run reported at step 20K; because the runs validated with their own micro-batch and a fixed cap of 20 batches, those come from subsets of different sizes (320 sequences for the dense runs, 640 for MoL). "Matched" re-scores the three step-20K checkpoints on the same 2,560 validation sequences and is the comparison we use.

the per-run curves MoL overtakes Dense 0.7B at step 7K. The sign matches 80M, where on clean schedules the widest topology leads a standard-shaped iso-active dense model by 0.77 PPL (Section 5.4). The closer 80M analogue of Dense 0.7B is the narrow-FFN Dense 98M, which the same topology leads by 2.42 PPL, so the margin over this kind of baseline shrinks from 80M to billion scale. Context length, tokens per parameter and the z-loss also differ between the scales, so we do not put the shrinkage down to scale alone. The match is in active parameters, not in FFN width, and Appendix O argues that a better-shaped iso-active dense model could erase a 0.17 PPL margin. The comparison uses one seed per condition. For reference, the cross-seed standard deviation is 0.06 PPL for the same MoL topology on WikiText-103 and 0.04 for the iso-active dense model at 80M on FineWeb-Edu (Appendix I).

Gap to Dense 1.3B. Dense Softmax 1.3B finishes 3.21 PPL ahead of MoL's 2.08B total on the matched protocol (3.01 as logged). The gap stabilizes at step 11K (Figure 1). Candidate causes (active FFN capacity, within-MoL sparsity, sub-Chinchilla token budget) are discussed in Appendix P.

Zero-shot transfer. On an 8-task lm-eval-harness suite, MoL and iso-active Dense 0.7B fall within one per-model standard error on five of eight tasks and MoL leads on two of the other three, WinoGrande (+5.1) and HellaSwag (+1.0). Against Dense 1.3B it trails on the broad-knowledge tasks by 1.5–4.9 pts and leads on WinoGrande. Averaged over tasks, transfer lands between the two dense baselines, though not on every task (Appendix Q).

Training cost. MoL Hybrid takes 1.91× the wall time of Dense 1.3B (102.0h vs 53.3h) and 2.29× that of Dense 0.7B (44.5h) at identical token budget. Two costs compound: gradient checkpointing is mandatory at 2.08B total params on 141 GB H200s, and DeltaNet's chunked kernel is more expensive to replay in the backward pass than FlashAttention. The tax is not specific to this scale: at 80M without checkpointing, per-token throughput is 1.9–5.2× lower than the narrow-FFN iso-active

dense model on an RTX 3090 and 4–13× lower on an RTX 5090 (Appendix G), because small routed blocks make training launch-bound. At equal wall-clock the comparison is less favourable: in the 102.0h that MoL used, Dense 1.3B would process about 40B tokens and Dense 0.7B about 48B.

5.6 Inference Wall-Clock: Prefill and Decode

We benchmark prefill and decode at batch=1 in bf16 (post-warmup; FlashAttention-2 for softmax, FLA Triton kernels for DeltaNet). Prefill is timed as a single forward pass with no cache. The decode benchmark enables the KV cache for both Dense and MoL. The checkpoint state matters for MoL: with the padding of Section 2.3, prefill cost scales with the trained router's load imbalance, which a random-init model (routing near-uniformly) understates. The inputs are uniformly sampled token ids, not natural text, so the routing loads are those the trained router produces on random input.

Decode. With a KV cache, Dense decode latency grows linearly with context while MoL's stays flat but higher at every measured length. Appendix M has the tables.

Table 6: Prefill throughput of MoL Hybrid 2.08B relative to Dense Softmax 1.3B (ratio of tokens/s; batch=1, single GPU, bf16, trained step-20000 checkpoints, faster of the two MoL dispatch paths at each T). Values above 1 (bold) mark the crossover. "–" = not measured (RTX 3090 beyond 64K: memory; A100 at 256K: job time limit). Absolute numbers and path selection: Appendix F.

GPU	T=4K	32K	64K	128K	256K
RTX 3090 24GB	0.52	0.80	**1.13**	–	–
A100 SXM4 80GB	0.23	0.58	0.82	**1.20**	–
H100 SXM 80GB	0.17	0.47	0.65	0.98	**1.42**
H200 141GB	0.14	0.51	0.70	**1.05**	**1.54**

Prefill crossover across four GPUs. The crossover is a long-context effect on every GPU we measured, but its position differs: near T=64K on the RTX 3090, between 64K and 128K on A100 and H200, and between 128K and 256K on H100 SXM (Table 6). The ordering does not follow the 2.4× bandwidth range (2.04→4.8 TB/s) or the 3.2× compute range (312→989 TFLOPS bf16) of the datacenter parts. We do not isolate the cause and claim no scaling law.

5.7 Limitations

(1) Cosmopedia, the billion-scale runs and the 80M MoL topologies are single-seed: the 0.175, 0.17 and 0.77 PPL differences each rest on one run. Measured seed spreads are 0.06–0.31 PPL across the WikiText-103 conditions of Tables 1 and 2, 0.04 over three seeds of the iso-active dense model at 80M on FineWeb-Edu, and 0.07 over three seeds of MoE-FFN, all as sample standard deviations (Appendix I). **(2)** MoL trains slower than dense at every scale we measured (Section 5.5), because per-block sparse dispatch is launch-bound for small blocks. We did not build a fused multi-block kernel, which could reduce this. **(3)** The prefill advantage is single-GPU, batch=1 on FlashAttention-2 + FLA kernels, and long-context only, with crossover onset between T=32K and 256K depending on the GPU (Section 5.6). Multi-GPU parallelism, batched serving and newer kernels could shift it. All models were trained at T=4096 and degrade beyond it without length extension, already at 8K and steeply by 32K (Dense 1.3B 15.2 → 24.0 → 221, MoL 18.5 → 31.7 → 119; Table 13). The prefill figures are therefore throughput measurements only, and MoL decode is slower at every measured context (3.3–10.2× over 256–16K). **(4)** The budget gives 10.1 tokens per total parameter for MoL and 16.0 for Dense 1.3B, both below the ∼20 tokens per parameter of the dense compute-optimal heuristic, which is not established for sparse models. Larger budgets are needed to test whether the gap persists. **(5)** At 80M a full-rank sparse MoE-FFN of the same total and active size beats the widest MoL topology on clean schedules (24.36 over three seeds against 25.17). No such control exists at billion scale. One visible cost: with $d_{\text{thin}}=d_{\text{model}}/4$ and a 4× thin FFN, each block's down/up projections are as large as its FFN and take 38% of MoL's non-embedding active parameters at both 80M and 2.08B, while MoE-FFN and Dense 0.7B give 67–72% of theirs to full-width attention, 3.3× MoL's in absolute terms. The projection-to-FFN ratio is $d_{\text{model}}/(4d_{\text{thin}})$: wider blocks lower it, in line with the k_{active} sweep (Appendix N). **(6)** Dense 1.3B holds 0.63×

MoL's total parameters: a compute reference, not a parameter match. At billion scale we trained no 2.08B dense model, no standard-shaped iso-active dense model and no repeats, for lack of compute.

6 Related Work

MoE and layer modularity. Standard MoE (Shazeer et al., 2017; Fedus et al., 2022; Zoph et al., 2022) routes tokens to FFN experts within a full-width layer, and sequential hybrids (Qwen Team, 2025; Glorioso et al., 2024; Ren et al., 2025) alternate attention types across layers rather than inside one. In contrast, MoL routes over whole transformer blocks at reduced d_{model}. Layer-level modularity work modifies *which* layers execute (MoD, Raposo et al., 2024; MoLEx, Teo & Nguyen, 2025; MoR, Bae et al., 2025). Sub-network parallelism partitions weights (UoE, Yang et al., 2025b), the FFN only (FlashMHF, Zhang et al., 2025; LatentMoE, Elango et al., 2026; Kimi K3, Kimi Team, 2026) or sub-blocks (AltUp, Baykal et al., 2023). Hyper-connections (Zhu et al., 2025; Xie et al., 2026; DeepSeek-AI, 2026) widen the residual stream. MoM (Gong et al., 2024), MoUE (Chen et al., 2026) and MoEUT (Csordás et al., 2024a) share modules at full d_{model}. Inside attention, routing selects heads (MoA, Zhang et al., 2022; SwitchHead, Csordás et al., 2024b; MoH, Jin et al., 2025) or a token subset per head (MoSA, Piękos et al., 2026), which raises the coverage question of Section 3. MH-MoE (Wu et al., 2024) splits tokens into lower-dimensional sub-tokens for FFN experts. Jamba (Lenz et al., 2025) interleaves attention and Mamba layers with MoE FFNs, and the always-active shared block plays the role of DeepSeekMoE's shared experts (Dai et al., 2024). We are not aware of a prior combination of parallel blocks, reduced-d_{model} projections (Houlsby et al., 2019; Hu et al., 2022), top-k block routing and hybrid attention.

7 Conclusion

We introduced MoL, which splits each transformer layer into parallel thin blocks selected by top-k block routing, with one shared softmax block and Gated DeltaNet in the routed blocks (Sections 2 and 3). On single-epoch data MoL comes out ahead: with softmax in every block it leads a dense model of equal size on Cosmopedia v2, and it leads iso-active dense models on FineWeb-Edu at 80M and at 2.08B total / 0.63B active. It does not beat a full-rank MoE-FFN at 80M or Dense Softmax 1.3B, trains 1.9–2.3$\times$ slower, and is cheaper than dense only in long-context prefill. We make no deployment claim. The training and decode costs come from launching each routed block separately (Appendices G and M). The next engineering step is to fuse the routed blocks into shared kernels. Whether the iso-active margin survives better-shaped dense baselines, MoE-FFN controls and more seeds at scale remains open.

AI Use Statement

We used generative AI tools to assist with translation of draft text into English, and to edit the manuscript for readability. We also used them to suggest refactorings of the released code, and to provision and operate rented GPU infrastructure for launching and monitoring training runs from scripts authored by the researchers. We consulted AI tools as a sounding board on statistical methodology, and during development they were used to review experimental designs and diagnose problems in them, including identifying a shared-weight confound in the hybrid-attention comparison and an initialisation bug in the DeltaNet blocks, and to help interpret results while drafting. During revision we also used them to implement, run and draft the description of the supplementary control experiments and diagnostics: the 80M training-throughput measurements, the perplexity evaluation beyond the training context, the re-scoring of the WikiText-103, Cosmopedia and billion-scale checkpoints on common validation sets, and the additional 80M control runs, clean-schedule retrains and extra seeds. In all of these sessions the authors set the direction and rejected suggestions that diverged from the intended design. We did not use generative AI tools to generate synthetic data, develop theoretical models, formulate or prove mathematical claims, originate the research questions of this work, or design or implement the MoL architecture and the main training runs it is evaluated with (the WikiText-103, Cosmopedia, FineWeb-Edu 80M sweep, and billion-scale experiments). The architecture, the questions it was built to answer, and the conclusions drawn from the results are the authors' own. AI-edited passages were checked against the underlying experimental records,

and AI-suggested code changes were tested by the authors before use. We take responsibility for the final content of this work, including all material produced with the aid of generative AI.

Ethics Statement

This work trains language models on three public corpora (WikiText-103, Cosmopedia v2 and FineWeb-Edu), involves no human subjects or private data, and releases code, logs and checkpoints under an MIT licence. We foresee no ethical concerns beyond those common to language-model pretraining.

Reproducibility Statement

The architecture and training setup needed to reproduce our results are specified in the paper. Section 4 lists the three datasets with their tokenizers, sequence lengths, and token budgets, and gives the full hyperparameters for both the 80M-scale ablations and the three billion-scale runs: layer counts, widths, head counts, d_{thin}, optimizer settings (β values, weight decay, gradient clipping), learning-rate schedule, warmup, batch size, step counts, and hardware. The thin-block construction and block routing are defined in Sections 2 and 3, and Section 2.3 reports the numerical equivalence between sparse dispatch and dense restricted attention.

All three datasets (WikiText-103, Cosmopedia v2, FineWeb-Edu) are publicly available. The multi-seed protocol and cross-seed standard deviation are given in Appendix I, and the zero-shot evaluation records its lm-eval-harness version, precision, batch size, context length, and tokenizer in Appendix Q. The inference benchmark protocol (batch size, warmup, kernels, KV-cache handling, and per-GPU specifications) is described in Section 5.6 and Appendix F, with the full decode tables in Appendix M, and the candidate causes of the gap to Dense 1.3B are analyzed in Appendix P. The common validation sets behind every matched comparison are specified in Appendix K, and the interrupted, resumed and extended training schedules, with estimates of their effect, in Appendix J. Settings not listed in Section 4, such as the optimizer settings of the FineWeb-Edu 80M sweep, are recorded in the released scripts and logs. An anonymized copy of the model code, training scripts, and raw training logs is included in the supplementary material. The trained checkpoints (the WikiText-103 MoL ablations, the Cosmopedia runs, the FineWeb-Edu 80M sweep and its control runs, and the final step-20000 weights of the three billion-scale models) are available from an anonymous repository at https://huggingface.co/iclr-mol/mol-checkpoints.

References

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 *NeurIPS*, 2025.

Cenk Baykal, Dylan Cutler, Nishanth Dikkala, Nikhil Ghosh, Rina Panigrahy, and Xin Wang. Alternating updates for efficient transformers. In *NeurIPS*, 2023.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. SmolLM-Corpus. Hugging Face dataset, https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus, 2024a. Contains Cosmopedia v2.

Loubna Ben Allal, Anton Lozhkov, and Daniel van Strien. Cosmopedia: How to create large-scale synthetic data for pre-training large language models. Hugging Face Blog, https://huggingface.co/blog/cosmopedia, 2024b.

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.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. MoEUT: Mixture-of-experts universal transformers. In *NeurIPS*, 2024a.

Róbert Csordás, Piotr Piękos, Kazuki Irie, and Jürgen Schmidhuber. SwitchHead: Accelerating transformers with mixture-of-experts attention. In *NeurIPS*, 2024b.

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

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

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models. In *ICLR*, 2025.

Venmugil Elango, Nidhi Bhatia, Roger Waleffe, Rasoul Shafipour, Tomer Asida, Abhinav Khattar, Nave Assaf, Maximilian Golub, Joseph Guman, Tiyasa Mitra, Ritchie Zhao, Ritika Borkar, Ran Zilberstein, Mostofa Patwary, Mohammad Shoeybi, and Bita Darvish Rouhani. Revisiting efficiency–accuracy scaling in mixture-of-experts architectures. In *ICML*, 2026. Preprint title: LatentMoE: Toward Optimal Accuracy per FLOP and Parameter in Mixture of Experts.

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.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7B SSM hybrid model. *arXiv preprint arXiv:2405.16712*, 2024.

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 *EMNLP*, 2024.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. MoH: Multi-head attention as mixture-of-head attention. In *ICML*, 2025.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Barak Lenz, Opher Lieber, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, et al. Jamba: Hybrid Transformer-Mamba language models. In *ICLR*, 2025.

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 *ICLR*, 2021.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters using Megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2021.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *NeurIPS Datasets and Benchmarks Track*, 2024.

Piotr Piękos, Róbert Csordás, Firas Laakom, Li Nanbo, and Jürgen Schmidhuber. Mixture of sparse attention: Content-based learnable sparse attention via expert-choice routing. *Transactions on Machine Learning Research*, 2026.

Qwen Team. Qwen3-Next-80B-A3B-Instruct. Hugging Face model card, https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct, 2025.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *ICLR*, 2025.

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 *ICLR*, 2017.

Rachel Teo and Tan Minh Nguyen. MoLEx: Mixture of layer experts for fine-tuning with sparse upcycling. In *ICLR*, 2025.

Xun Wu, Shaohan Huang, Wenhui Wang, Shuming Ma, Li Dong, and Furu Wei. Multi-head mixture-of-experts. In *NeurIPS*, 2024.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Mingyu Xu, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mHC: Manifold-constrained hyper-connections. In *ICML*, 2026.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *ICLR*, 2025a.

Yujiao Yang, Jing Lian, and Linhui Li. Union of experts: Adapting hierarchical routing to equivalently decomposed transformer. *arXiv preprint arXiv:2503.02495*, 2025b.

Minshen Zhang, Xiang Hu, Jianguo Li, Wei Wu, and Kewei Tu. Flash multi-head feed-forward network. *arXiv preprint arXiv:2512.06989*, 2025.

Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. Mixture of attention heads: Selecting attention heads per token. In *EMNLP*, 2022.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *ICLR*, 2025.

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.

Table 7: Rank-1 mirror on WikiText-103 (T=255), trained for 8 epochs (25.3K steps); the baseline row is the 50K-step run of Table 1.

Model	N/layer	Rank	Params	PPL
Rank-1 mirror	65,536	1	322.7M	47.51
Rank-1 MoE baseline	1,024	1	84.7M	33.93

A Rank-1 Mirror Experiment

Before decomposing the layer structure, we tried scaling rank-1 experts further. A rank-1 model with 65,536 experts per layer and 1,024 active per token (322.7M) was built to mirror the expert parameters of a 256-expert rank-256 model from an earlier pipeline. We do not compare it with that model, whose training and validation setup differ.

The mirror reaches 47.51, well behind the 84.7M rank-1 MoE baseline despite 3.8× its parameters, with about the same number of training tokens (826M against 816M). The two models differ in depth (4 against 8 layers), width (512 against 1024), router and schedule, so this compares two rank-1 designs; it does not measure a rank effect. Neither a much larger rank-1 pool nor more active rank-1 experts (Appendix B) closes the gap to dense models, so we move on to dense-FFN thin blocks (Section 5.1).

B Active Compute Scaling

At a fixed pool of N=4096 experts and 138M parameters, scaling from K=12 to K=128 active rank-1 experts (10× the expert FLOPs, though attention and the vocabulary projection dominate the model's total active compute, which rises far less) moves PPL from 33.49 to 33.22, closing 8% of the 3.23-PPL gap to the dense baseline (30.26) despite the 10× larger expert computation. The two runs also differ in routing temperature (τ=30 against τ=15) and in validation coverage (800 against 845 sequences), so the comparison does not isolate active-expert count. The result is consistent with quality being limited more by *per-expert expressiveness* than by active-expert count.

C Activation-Level Gating

Squeeze-and-Excitation (SE) gating (Hu et al., 2018) applied to expert activations (not d_{model}-dimensional vectors) achieves PPL 31.82, the best iso-parameter rank-1 result. A small MLP ($K \to 4 \to K$, 896 extra parameters total) was meant to modulate routing weights from expert activation magnitudes. With the zero initialization we used, only its output bias receives gradient, so in practice it learned a fixed gate per selected-expert slot rather than an activation-dependent one. SE closes 46% of the dense gap (33.93 → 31.82, target 29.36), though the three figures cover different validation subsets: SE validated at micro-batch 8 under the 50-batch cap and so scored 400 sequences, against 800 for the rank-1 baseline and the target. That target is the iso-parameter dense baseline trained at micro-batch 8 with 8 accumulation steps. The same architecture reads 30.26 at micro-batch 16 and 30.44 at micro-batch 64, all at the same effective batch of 64. The three runs differ in more than the split: the micro-batch-16 run used seed 137 rather than 42, and the micro-batch-8 run scored 800 of the 845 validation sequences rather than all of them. The 0.18 PPL step from 64 to 16 is within seed noise. We report the further 0.90 PPL at micro-batch 8 as an observed difference between confounded runs, not as a micro-batch effect: no matched-seed, matched-exposure, matched-validation experiment was run, and the seed-137 run also resumed from a checkpoint. The 29.36 figure is not comparable with the rest of the WikiText-103 results, which use micro-batch 16 to 64. Section 5.3 compares runs at matched micro-batch. Table 2 mixes 16, 32 and 64, a range worth 0.21–0.31 PPL for the dense-FFN MoL at a fixed seed. Active compute scaling and activation gating both leave part of the gap, so we turn to MoL's structural decomposition.

Table 8: MoL granularity sweep on WikiText-103 (50K steps). All with rank-1 MoE FFN.

Config	d_{thin}	K	Params	PPL	Speed (steps/s)
Rank-1 MoE baseline	1024	—	84.7M	33.93	—
K=7 d=256 split_all	256	7	85.6M	**34.83**	1.51
K=7 d=256 split_mid6	256	7	85.4M	35.05	1.85
K=18 d=128	128	18	86.1M	36.67	1.08
K=40 d=64	64	40	84.7M	38.65	0.69
K=1 d=256	256	1	51.4M	39.71	4.87

D MoL Granularity Sweep

With K=7 at d_{thin}=256 and all layers split, MoL reaches PPL 34.83, only 0.90 PPL worse than the baseline. At iso-parameters, d_{thin}=256 (K=7) outperforms d_{thin}=128 (K=18) by 1.84 PPL, though the two runs also differ in which layers are split and in micro-batch, so the figure confounds width with both. Comparing the two six-layer-split rows narrows it to 1.61 PPL. Full-width anchor layers appear unnecessary: split_all (34.83) outperforms split_mid6 (35.05).

E Sparse Dispatch Details

Balance-loss reduction. The CV^2 reduction over layers is not uniform: WikiText-103 and the billion-scale runs average the per-layer terms before applying α, while Cosmopedia and the 80M FineWeb-Edu sweep sum the weighted terms, eight times the effective strength at eight layers. The WikiText-103 MoE-FFN control of Section 5.1 also sums, so its balance loss is eight times that of the MoL runs it is compared with. We did not sweep this.

Shared-block weighting. The fixed shared weight w_s of Equation (3) takes two values across the experiments. The WikiText-103 runs (Table 2) and the billion-scale runs (Section 5.5) use w_s=$1/N$ with w'_i=w_i. The Cosmopedia and FineWeb-Edu 80M runs use the balanced form w_s=$1/(S{+}k)$ with w'_i=$w_i \cdot k/(S{+}k)$, so the shared block and the k routed blocks together carry unit weight. Neither setting scales the sum by $1/(k{+}1)$.

Forward-pass speedup vs sparsity. On the forward pass of 8-layer models (B=16, T=512), gather/scatter dispatch is 1.53× faster than restricted attention at 57% active blocks (top-4-of-7, 106M), 2.85× at 20% (top-2-of-10, 138M) and 4.34× at 10% (top-2-of-20, 243M). With `torch.compile(dynamic=True)` the last figure rises to 4.94×. The three points use different pool sizes, so they are not one model measured at three sparsities, and all three are freshly initialised models, whose near-uniform routing is the best case for padding to the busiest block.

DeltaNet on gathered tokens. A routed DeltaNet block receives its tokens as a compact tensor in their original order. The short causal convolution and the gated delta recurrence then run over this compacted sequence with unit steps: the decay is not rescaled by the original inter-token gap, the state is not reset across gaps, and DeltaNet uses no RoPE, so no positional index is carried into the block. A routed block therefore treats its tokens as an adjacent stream, and any dependence on absolute position comes from the shared softmax block, which sees every token with RoPE. The numerical-equivalence check of Section 2.3 covers the softmax path only. For DeltaNet the sparse path is the definition of the model, not an approximation of a dense one.

Projection overhead. The projection cost (W_{down}, W_{up}) sets a floor on d_{thin}: for a dense-FFN wrapper at d_{thin}=256, projections account for 40% of wrapper parameters, rising to 57% at d_{thin}=128 and 73% at d_{thin}=64. The rank-1 wrapper of the granularity sweep carries a steeper 56%, 70% and 81%. Empirically, d_{thin}=256 outperforms d_{thin}=128 at iso-parameters (∼85M) by 1.84 PPL, or 1.61 PPL between the two runs that split the same layers (Appendix D), so width outweighs count.

F PREFILL THROUGHPUT: FULL PER-GPU TABLES

Why the crossover appears only at long context. MoL's only quadratic term is its shared softmax block, 8 heads of width 64 against 16 of width 128 in each of Dense 1.3B's 24 layers, a quarter of the dense $O(T^2)$ attention work, while the routed DeltaNet blocks are $O(T)$. The saving grows with T until it outweighs MoL's dispatch overhead, consistent with a crossover only at long context.

Implementation. MoL has two dispatch paths and each cell reports the faster one. At long context that is usually `sparse`, because the stacked path exceeds memory (at T=128K on A100, for instance, `batched_sparse` runs out of memory and the reported figure is the `sparse` one). The batched path, `attention_mode="batched_sparse"`, fuses the N=14 routed FLA calls into one `chunk_gated_delta_rule` call with $N \times H$ heads and stacks per-block projections via `bmm`. The naive (sparse) path is 2.8× slower on 3090 (T=256) and 2.77× on H200 (T=4K). The two paths agree in quality but not bit for bit: validation PPL on the trained billion-scale MoL checkpoint shifts by -0.11% (18.48 → 18.46), while end-to-end logits differ by up to 2.4 (0.16 on average) in bf16. These timings predate the RoPE-cache correction of Appendix L. Re-timed on the RTX 3090 with the old and the corrected cache, MoL's prefill throughput is unchanged (8,730 against 8,739 tokens/s at T=32K, 8,052 against 8,061 at 64K), as is its routed-block load imbalance, although the outputs differ.

The summary ratios of Table 6 are computed from the absolute measurements below.

G DELTANET SPEED CROSSOVER AND TRAINING THROUGHPUT

DeltaNet crosses over softmax at $T \approx 2048$ and is 7.8× faster at T=16K.

End-to-end training throughput at 80M. The layer-level advantage above does not carry over to end-to-end training of the routed architecture. Table 11 reports per-token training throughput (bf16 autocast, T=2048, micro-batch 2 with 16 accumulation steps, no gradient checkpointing) for the conditions of the 80M FineWeb-Edu sweep: RTX 3090 values are steady-state averages from the sweep's training logs. RTX 5090 values come from a controlled 8-step microbenchmark of the same model code. Per-block sparse dispatch issues one kernel sequence per routed block per layer (14 blocks × 8 layers for 1+3of15), which is launch-bound at this block size, and the penalty grows with the number of routed blocks on both GPUs. The two columns differ in more than the GPU. The 3090 values come from trained models consuming corpus batches under the training script's fused optimizer. The 5090 microbenchmark builds fresh models, feeds uniformly sampled token ids and steps a default AdamW without the scaler and clipping path. Since dispatch cost tracks routing imbalance and a near-uniform router is cheaper, the 5090 column describes that microbenchmark rather than steady-state corpus training. The two stacks also differ in PyTorch build. Only ratios within a column should be compared.

H BLOCK-PARALLEL MODEL SHARDING (PROTOTYPE)

MoL's thin blocks have no within-layer cross-block data dependencies, a property that suits expert-parallel placement. We measured a prototype that splits the 15 thin blocks of a 1+2of15 model with routed softmax attention (190.4M) across 2 GPUs over PCIe. Each device owns a disjoint subset of the routed blocks and replicates the shared block. The split stage is summed with one all-reduce. Relative to the same model on one GPU it cuts per-GPU parameters by 39% and VRAM by 23%, at 2.60 versus 3.94 steps/s, i.e. 1.52× the wall-clock per step. The prototype synchronises gradients only for the parameters outside the thin blocks (embedding, head, norms and routers), not for the replicated shared block, whose copies therefore drift apart across devices. It measures step time under the intended communication pattern and is not a training-equivalent implementation. Training equivalence would also need the shared block's gradients and the gradient flowing back into each split stage summed across devices, plus a loss-curve match against the single-GPU run. We did none of these. A three-device run and an all-to-all token-dispatch variant (implemented in the released code) were not benchmarked. On PCIe the collective dominates, and NVLink should reduce the penalty. Compared to Megatron-style tensor parallelism, which incurs two all-reduces

Table 9: Prefill throughput (tokens/s, batch=1, single GPU; mean of 15 timed runs after 15 warmup runs on the datacenter GPUs, and of 10 after 8 warmups for the RTX 3090 cells, which were re-measured with a separate script; per-run timings were not retained, so we report no dispersion). **All cells use the trained step-20000 checkpoints.** MoL Hybrid value is the faster of `batched_sparse` and `sparse` at each T. *Best per row in bold* marks the crossover between Dense and MoL Hybrid.

SeqLen	Dense Softmax 1.3B	MoL Hybrid 2.08B
RTX 3090 (consumer; 936 GB/s GDDR6X, 71 TFLOPS bf16)		
4,096	**17,548**	9,148
32,768	**10,666**	8,579
65,536	6,942	**7,873** (sparse)
A100 SXM4 80GB (2.04 TB/s HBM, 312 TFLOPS bf16)		
4,096	**43,585**	9,998
32,768	**28,472**	16,650
65,536	**19,495**	15,968
131,072	11,779	**14,115** (sparse)
H100 SXM 80GB (3.35 TB/s HBM, 989 TFLOPS bf16)		
4,096	**98,476**	16,459
32,768	**71,925**	33,579
65,536	**51,731**	33,559
131,072	**31,299**	30,819 ($\approx$0.98$\times$)
262,144	17,495	**24,841** (sparse)
H200 (4.8 TB/s HBM, 989 TFLOPS bf16)		
4,096	**102,460**	14,174
32,768	**71,912**	36,911
65,536	**51,860**	36,292
131,072	31,348	**33,008** (sparse)
262,144	16,973	**26,105** (sparse)

Path-selection protocol: `batched_sparse` stacks all N=14 routed FLA calls into one kernel (peak $\sim$85 GB at 128K) and dominates short context where dispatch overhead matters; `sparse` dispatches per block (peak $\sim$32 GB at 128K) and dominates long context where the stacked footprint OOMs. We report the faster path at each T. Because MoL's dispatch cost scales with routing load imbalance, only the trained router gives a representative prefill number; a random-init MoL routes near-uniformly and runs $\sim$2.2$\times$ faster on 3090, so it is not a valid speed proxy. **Bench limits**: on A100 80GB the sweep stops at 128K (the T=256K job hit its time limit before producing a measurement, so neither model has an A100 number there); Dense Softmax 1.3B OOMs at T=1M on H200; the H200 sweep at T=524K hit its job time limit while still benchmarking a dense model, so we have no MoL measurement above T=262K and do not claim a ceiling for it; FlashAttention-3, fp8 KV-cache, or multi-GPU serving could move all ceilings.

Table 10: DeltaNet vs softmax speed in one full thin block (d=256, 4 heads, d_{ff}=512, batch 2, forward+backward).

SeqLen	Softmax (ms)	DeltaNet (ms)	Ratio
512	1.07	3.12	0.34$\times$
1024	1.89	3.18	0.59$\times$
2048	**4.04**	**3.33**	**1.21$\times$**
4096	11.06	4.81	2.30$\times$
8192	36.08	8.93	4.04$\times$
16384	131.81	16.88	7.81$\times$

per transformer layer in the forward pass (Narayanan et al., 2021), and to standard expert-parallel MoE, where dispatch/combine collectives per MoE layer require dedicated pipeline engineering to hide (e.g., DualPipe in DeepSeek-V3 (DeepSeek-AI, 2024)), MoL's block-parallel design keeps attention local to each block: a routed block attends only within its own token subset, so no sequence-parallel communication is needed inside a block. The measured prototype needs one all-reduce per split stage. The all-to-all variant needs four collectives per stage (split sizes, token dispatch, metadata, and the return exchange). Neither uses an overlap scheme, and whether one is needed at

Table 11: End-to-end training throughput at 80M on FineWeb-Edu (ktok/s, higher is better) and ratio to the iso-active dense model. RTX 3090: training-log averages at micro-batch 2×16 from the trained runs; RTX 5090: 8-step microbenchmark on freshly initialised models with synthetic inputs, same shapes.

Condition	Total	RTX 3090	ratio	RTX 5090	ratio
Dense 98M (iso-active, d_{ff}=768)	97.7M	43.9	1.00	112.5	1.00
Dense 152M (d_{ff}=4096)	152.2M	38.0	0.87	91.7	0.82
MoL Hybrid 1+3of5	106.2M	23.0	0.52	28.2	0.25
MoL Hybrid 1+3of10	161.5M	12.8	0.29	13.3	0.12
MoL Hybrid 1+3of15	216.9M	8.5	0.19	8.7	0.08

scale is untested. A larger-scale strategy would combine block placement (primary axis), pipeline depth, and context parallelism inside long routed sub-sequences. We leave a measured throughput study to future work.

I Multi-Seed Validation

Table 12: Multi-seed validation (seeds 42, 137, 256) on WikiText-103, 50K steps. The DeltaNet 1+3of15 column is re-scored on the full 845-sequence validation split; its training-time values, capped at 800 sequences, were 29.96, 29.94 and 30.08 (Appendix K).

Seed	1+3of15 (198M)	1+2of5 (87.5M)	Softmax 1+2of5 (85.3M)	Softmax 1+2of5 iso-param (87.5M)	Sparse 3/5 (85.3M)	Full top-3 (85.3M)	Full all (85.3M)
42	30.51	30.95	32.46	31.82	32.04	30.85	31.80
137	30.48	31.57	32.41	32.11	32.33	30.93	32.03
256	30.60	31.20	32.19	32.34	32.21	31.06	31.98
Mean±std	30.53 ± 0.06	31.24 ± 0.31	32.35 ± 0.14	32.09 ± 0.26	32.19 ± 0.15	30.95 ± 0.11	31.94 ± 0.12

Shared-block ablation. On WikiText-103, removing the shared block at an equal number of active blocks does not cost quality: DeltaNet 3-of-5 without a shared block reaches 30.89 ± 0.35 (3 seeds) against 31.24 ± 0.31 for 1+2of5, and at fifteen blocks 32.10 against 33.32 (one seed each, full split). The softmax pairs move the same way (32.19 against 32.35, and 35.34 against 37.17). At T=255 the hybrid's quality gain therefore comes from DeltaNet in the routed blocks. The shared block is kept by design, to give every token full softmax attention at every layer, which we expect to matter beyond these short sequences; we did not train a model without it at long context.

FineWeb-Edu 80M (Table 4). Two baselines of the 80M sweep were repeated on the clean 60K schedule. On the common validation set of Table 4, the iso-active Dense 95M finishes at 26.00, 25.92 and 25.93 for seeds 42, 137 and 256 (25.95 ± 0.04), and MoE-FFN top-2 at 24.39, 24.28 and 24.41 (24.36 ± 0.07). Seeds 137 and 256 ran uninterrupted. Seed 42 of MoE-FFN was resumed once (Appendix J). No MoL topology was repeated on this corpus. On WikiText-103 the MoL seed spreads range from 0.06 to 0.31 PPL (Table 12).

J Training Schedules and Data Order

Runs on the rented and local GPUs were interrupted and resumed, and the 80M FineWeb-Edu sweep was additionally extended from its original 38K-step schedule to 60K by resuming a mid-run checkpoint. Resuming restores the model, optimizer and step counter but not the loader's position, and the loader is rebuilt with `shuffle=True` under the run's seed, so it replays the same permutation from its first element. How far a resumed run reaches into its intended data order is therefore bounded by its longest uninterrupted segment, not by its step count.

Where this matters depends on how many passes a run was making. WikiText-103 spends about eight epochs on 103M tokens, so a restart changes the pattern of repetition rather than the coverage. We have no control that measures what the interruptions there cost. The Cosmopedia runs spend 14.94B tokens on the 15B-token slice they stream, so a restart removes the tail: dense DeltaNet

(resumed at 60K and 85K of 114K), sparse MoL (60K) and MoE-FFN (55K and 75K) each reach about half of the slice, while dense softmax and MoL K=5 top-3, the pair that sets the headline margin of Table 3, ran uninterrupted. The 80M sweep spends 3.93B tokens against a 24.8B-token split, so it repeats part of its slice without exhausting the data. Its longest uninterrupted segment is 30K of 60K steps for most conditions and shorter for 1+3of15. The three runs of Table 5 ran uninterrupted on a single schedule.

Schedule extension. Extending the sweep re-created the cosine schedule with the longer horizon, so the learning rate jumps upward at the resume point: Dense 98M sits at 3.5×10^{-5} at step 30K under the 38K schedule and at 1.6×10^{-4} at step 30,050 under the 60K one. In Table 4, only 1+3of10 still carries the extension. Every other row, including the retrained Dense 152M, Dense 98M, 1+3of5 and 1+3of15, was trained on a single clean 60K schedule. The seed-42 MoE-FFN run was resumed once, at 10K steps, on that same schedule, so its learning rate is continuous but its data order replays the first 10K steps. Its uninterrupted seeds finished 0.11 PPL lower and 0.02 higher, so the interrupted run lies within the range of the uninterrupted seeds. The extra seeds and the clean 1+3of15 and Dense 152M runs draw their data order from a restart-safe sampler, a different permutation from the loader's at the same seed. None of them was interrupted.

We measured what the difference is worth on two architectures. Retraining the narrow-FFN dense baseline on a clean 60K schedule at the same seed takes it from 28.23 to 27.59, and retraining MoL 1+3of5 the same way takes it from 26.67 to 25.79, both on the common validation set. Each pair shares architecture, data pipeline and seed and differs in schedule and data order, but both pairs also ran on different GPU models under different PyTorch builds, and none of the four runs is repeated. The 0.63 and 0.88 PPL are single-trial estimates, not isolated measurements. Retraining 1+3of15 moves it by less, from 25.42 to 25.17, but its sweep run had also seen 17% more tokens (micro-batch 4×16 for its first 10K steps), so it does not measure the schedule term. Dense 152M moves from 24.01 to 23.50. The 80M comparisons we draw, MoL against Dense 95M and against MoE-FFN, are all between clean-schedule runs. Only 1+3of10 is reported as extended.

K Evaluation Protocols for the Matched Comparisons

Several of this paper's tables report perplexities that the training runs did not produce, mostly because each run validated on a subset of its own choosing. This appendix records how those numbers were obtained.

Cosmopedia held-out set (Table 3). The five Cosmopedia checkpoints were trained on the first 15B tokens of cosmopedia-v2, which the corpus stores as 104 shards in order. We tokenised the last shard, `train-00103-of-00104`, with the same GPT-2 BPE, packed it into 4,000 sequences of 2,048 tokens with no padding, and scored each condition's step-114K checkpoint on it at batch size 4 under bf16 autocast. No training run reached that shard.

FineWeb-Edu matched subset (Table 5). The three billion-scale checkpoints were re-scored on the first 2,560 sequences of the held-out FineWeb-Edu validation split at batch size 2 with shuffling disabled, so every model sees the same sequences in the same order. Both a batch-mean and a token-weighted mean were computed. The two agree to within 0.0002 PPL for every model.

FineWeb-Edu 80M common set (Tables 4 and 15). The 80M validator scores the first 50 batches of an unshuffled loader at twice the training micro-batch, so runs trained at micro-batch 2 scored the first 200 validation sequences and the micro-batch-4 run of 1+3of15 scored 400. We re-scored every 80M checkpoint on the first 2,048 sequences at batch 8 under bf16 autocast, token-weighted, with the validator's loss definition. Replaying each run's own capped protocol reproduces its logged value to within 0.003 PPL. Values on the common set are 0.9–1.4 PPL lower than the logged ones. Two pairs swap order: the clean 1+3of15 run, which the harder second 200 sequences had penalised, now reads below its sweep run, and 1+2of5 reads 0.05 below 1+1of5.

WikiText-103 full validation split (Tables 2 and 12). The 15-block conditions finish training at micro-batch 16 and validate under a 50-batch cap, so they scored the first 800 of the 845 validation sequences, while most other WikiText rows validate at a larger batch and reach all 845. The exceptions, which we did not re-score, are the rank-1 MoE baseline and the $K{=}12$ run of Appendix B (800

sequences) and the SE-gating run of Appendix C (400). We re-scored the seven affected checkpoints on the full split at micro-batch 16, with the same masking and bf16 autocast. Replaying the 50-batch cap reproduces each published value to within 0.011 PPL, which confirms the reconstruction. Under the cap the three Table 2 rows had read 36.51 (Softmax 1+2of15), 32.75 (DeltaNet 1+2of15) and 29.99 ± 0.07 (DeltaNet 1+3of15). On the full split every one of them is 0.52–0.66 PPL worse, so the last 45 sequences are the harder ones, and the ordering among them is unchanged. The tables report the full-split values. Like the training validators, the re-scoring averages batch losses, so the final partial batch weighs differently at batch 16, 32 and 64; for one checkpoint the batch mean is 30.51 and the token-weighted mean 30.54.

Contiguous long-context stream (Appendix L). The processed validation split keeps every 200th chunk of the training stream, so two adjacent 4,096-token rows lie about 815K original tokens apart and a window longer than one row joins unrelated excerpts. For the length sweep we therefore tokenised a late shard of the same `sample-100BT` subset, which the 25B-token preparation never reached, and packed 2,097,152 tokens into one uninterrupted stream exactly as training packed its data: documents concatenated with no separator and no padding.

Baseline implementations. The WikiText-103 dense baselines come from a separate script that differs from the MoL runs in normalization (LayerNorm against RMSNorm), activation (SiLU against GELU) and learning-rate floor (cosine clipped at $0.1\times$ against decay to 0). The Cosmopedia dense softmax baseline differs from MoL in normalization, activation and FFN bias (Section 5.3), while the Cosmopedia dense DeltaNet control and every MoL run share one block implementation. Comparisons between MoL and these dense baselines therefore mix the architectural effect with these implementation differences. The 198M dense row of Table 2 is the micro-batch-16 run, matching the 15-block MoL runs. A micro-batch-64 run of the same model at another seed reads 27.09.

Loss definition. The training-time validators pass `ignore_index=0` to the cross-entropy, intended to skip padding. Neither corpus is padded in the region we score: FineWeb-Edu packs documents end to end and reserves id 50256 for padding, and the Cosmopedia held-out sequences are fully packed. GPT-2 id 0 is the ordinary token `!`, which covers 0.37% of the Cosmopedia held-out targets and 0.06% of the contiguous FineWeb-Edu stream, so the flag drops a few real tokens rather than padding. The held-out column of Table 3 and the tables of Appendix L score every position. Table 5 keeps the flag so that its two columns share a definition. Every condition inside a given table uses one definition, so each table compares models on the same metric; margins under full-token scoring were not computed for the tables that keep the flag.

L Modelling Quality Beyond the Training Context

The prefill benchmarks of Section 5.6 run to $T{=}256$K, but all three billion-scale models were trained at $T{=}4096$. Table 13 reports perplexity on windows of increasing length cut from one uninterrupted held-out FineWeb-Edu stream, evaluated without any length-extension technique (262K tokens per length, every position scored; Appendix K gives the construction). Two earlier versions of this measurement were wrong in ways worth recording. One cut its windows from the processed validation split, which keeps every 200th chunk of the training stream and so spliced unrelated excerpts at every 4,096-token boundary. The other, and larger, defect was in the RoPE cache: when a window exceeded the 4,096 positions built at initialisation, the cache was rebuilt under bf16 autocast, where position indices above 4,096 collapse onto multiples of 32 (64 above 8,192) and the whole table loses precision. That inflated every $T{\geq}8$K entry, and it inflated the dense models more than MoL, since they carry RoPE in every block while MoL carries it only in its shared softmax block. Under the defect MoL appeared best already at 8K. The table below is measured with the cache built in float32 outside autocast. Absolute values at 2K–4K differ slightly from the training-time validation numbers because of the different windowing and the different held-out text.

All three models degrade once the window exceeds the training length, as expected for RoPE softmax attention extrapolated 2–$8\times$ without adjustment, though far less sharply than the defective measurement suggested. At 8K the ordering follows model quality, with Dense 1.3B best and MoL worst. MoL's relative advantage appears only from 16K. Over the full range MoL degrades the least: measured as the increase in mean cross-entropy from 4K to 32K, it loses 1.86 nats against 2.68 for Dense 1.3B and 2.98 for Dense 0.7B. This is not an artefact of starting from a weaker

Table 13: Perplexity vs context window on a contiguous held-out stream (T tokens per window, no length extension). Right block: perplexity on positions ≥ 4096 only.

	All positions					Positions $\geq$ 4K		
Model	2K	4K	8K	16K	32K	8K	16K	32K
Dense Softmax 1.3B	15.51	15.16	24.0	86.8	221	37.7	154	326
Dense Softmax 0.7B	19.19	18.77	27.1	120	371	38.6	219	572
MoL Hybrid 2.08B	18.86	18.49	31.7	66.9	119	53.9	102	156

model. Dense 0.7B begins within 0.29 PPL of MoL at 4K and degrades the *most* of the three, so a higher starting perplexity does not by itself buy a flatter curve. At matched starting quality MoL keeps 1.12 nats more. A plausible reading is that three of MoL's four active blocks per layer are DeltaNet blocks carrying no positional index, so only the shared softmax block extrapolates RoPE. We did not isolate this, and the model is not usable at these lengths either. The prefill crossover of Section 5.6 is therefore a throughput result. Obtaining long-context quality would require length extension or long-context training, which we did not do.

M Decode Latency: 3090 KV-cache Tables

Decode (3090 KV-cache). With a KV cache, Dense Softmax decode latency grows roughly linearly with context (6.2 ms/tok at 256, 19.6 at 16K, 26.5 at 24K), while MoL stays near 63–65 ms over the range where both were measured on the same implementation (256–16K). A per-token dispatch floor of $\sim$2,160 Python op launches dominates it and keeps MoL slower than Dense at every measured context.

Section 5.6 gives a one-line summary; the full 3090 table is reproduced here. Dense Softmax incurs $O(T)$ KV-cache memory traffic in the attention of every layer, whereas MoL splits attention across a shared softmax block (also $O(T)$) and routed DeltaNet blocks ($O(1)$ recurrent state). With 1 shared and 3 routed-active blocks per token in 1+3of15, only one of the four active blocks per token runs $O(T)$ attention. The growth is still linear in T, with a smaller coefficient sitting on top of a large constant dispatch cost, which is why the measured curve looks flat over this range.

Table 14: Decode latency per token (ms, batch=1, RTX 3090, KV cache enabled). MoL figures are from the T=1 decode fast path, benchmarked up to 16K; the 24,576 cell is blank because only the earlier, slower implementation was measured there. Dense grows with context while MoL stays near 63–65 ms, held there by a per-block Python dispatch floor.

Context	Dense Softmax 1.3B (ms/tok)	MoL Hybrid 2.08B (ms/tok)	Dense growth
256	**6.2**	63.2	—
1,024	**6.9**	64.2	+12%
4,096	**9.6**	63.3	+55%
8,192	**12.9**	62.5	+108%
16,384	**19.6**	65.2	+216%
24,576	**26.5**	—	+328%

Over the range where both were measured with the same implementation (256–16K), Dense's decode latency grows $3.2\times$ while MoL's grows 3%. Dense was also measured at 24K; MoL was not on that implementation. KV cache alone gives $2.74\times$ / $1.92\times$ / $4.81\times$ uplift on Dense 1.3B / Dense 0.7B / MoL at short context. The decode fast path adds a further 29% for MoL. These are latency measurements only: the cached MoL path routes a batch by its first sequence, and for routed blocks that receive fewer tokens than the busiest one it keeps recurrent state computed over padded positions, so it is not a verified-exact generator. The $\sim$60 ms floor would shrink under a per-block-fused kernel: a partial fix (FLA-only fusion) gave $+8\%$ throughput. Full fusion is left to future work. The decode bench was not re-run on H200. Prefill and batch-1 decode are bound by different resources (compute for prefill, memory bandwidth and launch overhead for decode), and MoL did not overtake Dense in decode at any measured context on the 3090. We make no claim about a decode crossover on Hopper.

N k_{ACTIVE} SWEEP AT 80M (FIXED POOL N=5)

To probe the effect of k_{active} at fixed pool size, we ran two additional 60K conditions on FineWeb-Edu alongside the existing 1+3of5 baseline, varying k_{active} while applying the MoL design rule $k_{\text{active}} \times d_{\text{thin}} \approx d_{\text{model}}$ and $k_{\text{active}} \times d_{\text{ff,thin}} \approx d_{\text{ff,dense}}$ (active FFN width held at ∼4096):

Table 15: k_{active} sweep at 80M, N=5, FineWeb-Edu 60K steps. All conditions iso-active-FFN-width ∼4096. Single seed each. Values on the common set of Table 4. Δ is against the sweep's own narrow-FFN Dense 98M run (28.23), which shares these rows' schedule extension, rather than against the clean-schedule retrain of that condition in Table 4 or the standard-shaped Dense 95M.

Condition	k	d_{thin}	$d_{\text{ff,thin}}$	Total	Active	PPL @ 60K	Δ vs Dense 98M
1+1of5 (k=2)	2	512	2048	228M	121M	26.34	−1.88
1+2of5 (k=3)	3	320	1344	132M	99M	**26.29**	−1.93
1+3of5 (k=4, baseline)	4	256	1024	106M	95M	26.67	−1.55

At k=2 and k=3 the results are within 0.05 PPL of each other; k=4 trails by ∼0.35 PPL. The sweep is single-seed. On this corpus the seed spread is about 0.04 PPL for Dense 95M and 0.07 for MoE-FFN (Appendix I), so the ∼0.35 PPL deficit of k=4 exceeds those models' spread and the 0.05 between k=2 and k=3 does not, though MoL's own spread on this corpus was not measured. All three rows also carry the schedule extension, which cost 1+3of5 0.88 PPL (Appendix J) and was not measured for the other two. Total parameters are not held fixed across the sweep (228M at k=2 vs 106M at k=4), so block granularity is confounded with capacity. Separating the two would need N scaled inversely with d_{thin}. With that caveat, the direction is opposite to MoE granularity findings (Dai et al., 2024) that favor fine-grained routing. One possible reason is scope: in MoL, each block includes attention plus FFN (not FFN alone), so per-block capacity may matter more than block count. We do not claim it generalizes beyond the 80M / N=5 regime.

O SHAPING THE ISO-ACTIVE DENSE BASELINE

At iso-active parameters the answer depends on how the dense baseline is shaped. Against the sweep's original baseline, which holds d_{model}=1024 and shrinks the FFN to d_{ff}=768 to match active parameters, all three MoL topologies finish 1.55–2.81 PPL ahead on the common validation set. That baseline is handicapped: a 0.75× FFN expansion is far from the standard 4×, and a dense model that keeps the standard shape and instead narrows the residual stream (d_{model}=768, d_{ff}=3072, 95.3M) reaches 25.95 ± 0.04 over three seeds at the same active budget. Retraining the narrow-FFN baseline on the same clean 60K schedule brings it from 28.23 to 27.59, so of the 2.28 PPL originally separating the two shapes, roughly 0.63 went with the schedule and replay and 1.64 with the change of shape (residual width, heads and FFN width together). Both figures rest on a single run of the narrow-FFN model (Appendix J). Against that baseline, on the sweep's own runs, only the widest topology is ahead, by 0.53 PPL, while 1+3of10 and 1+3of5 fall 0.60 and 0.72 PPL behind. That reading changes on matched schedules. Retrained on the same clean 60K cosine as the baseline, 1+3of5 moves from 26.67 to 25.79, 0.16 PPL ahead of the Dense 95M mean, and 1+3of15 finishes at 25.17, 0.77 PPL ahead of it; 1+3of10 was not retrained (Appendix J). What we can claim at 80M is that on matched schedules the widest MoL topology beats a well-shaped dense model of the same active size, the narrowest is slightly ahead of it, and the FFN-shape effect on the dense side is 1.64 PPL. At billion scale the same question is open. Dense Softmax 0.7B keeps d_{model}=2048 with d_{ff}=2048, a 1× expansion, whereas MoL activates four thin blocks whose FFNs sum to width 8192, the same as Dense 1.3B. The match is in active parameters, not FFN width. Since moving to a standard 4× shape was worth 1.64 PPL at 80M on clean schedules, far more than the 0.17 PPL MoL leads by at billion scale, a well-shaped iso-active dense model at that scale could erase the margin. That control was not trained.

With a larger active budget (Dense 152M, 1.6× MoL's active parameters), Dense finishes 1.67 PPL ahead of the best MoL on the common validation set. MoL trades total parameters for fewer active parameters, as MoE does, but at the block level. Appendix N varies k_{active} at a fixed pool of N=5.

P GAP TO DENSE 1.3B: CANDIDATE CAUSES

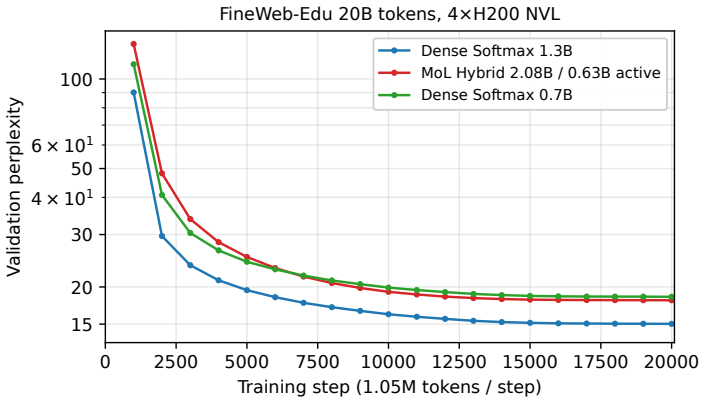


Figure 1: Validation perplexity (log scale) vs training step, three billion-scale runs on FineWeb-Edu 20B tokens. Dense Softmax 1.3B (blue) leads from step 1K and its gap to MoL Hybrid (red) stabilizes at step 11K. Dense Softmax 0.7B (green) starts below MoL and is overtaken at step 7K.

The 3.21 PPL gap between Dense Softmax 1.3B and MoL 2.08B-total at 20K steps does not reproduce the Cosmopedia crossover at ∼104M. Three candidate causes are consistent with the missed crossover, but cannot be separated from a single trio of runs:

Active per-layer FFN capacity. At billion scale, MoL's active capacity (4 active blocks at $d_{\text{ff,thin}}$=2048) totals 8192, matched to Dense's d_{ff}=8192. On Cosmopedia at ∼104M, by contrast, MoL's active capacity ($3 \times 1024 = 3072$) was 2.74× Dense's d_{ff}=1120, leaving a much larger margin to amortize routing overhead.

Within-MoL sparsity. Higher at billion scale: 4 of 15 blocks active vs 3 of 5 on Cosmopedia 1+2of5, so the routed pool to specialise across is far larger for barely more active blocks.

Total-parameter undertraining. On a total-parameter basis, MoL sees only 10.1 tokens/param, well below the ∼20 of the dense Chinchilla heuristic, which may not transfer to a sparse 2.08B-total model. Dense 1.3B sees 16.0 tokens/param.

Telling whether the gap is structural, reflects undertraining, or stems from optimizer, router or auxiliary-loss tuning specific to this scale would take a multi-seed budget sweep that we did not run.

Q ZERO-SHOT EVALUATION: PER-TASK DETAILS

Run config: lm-eval-harness 0.4.7, bf16, batch size 64 (47.5 GB peak on H100 NVL), max length 2048, GPT-2 tokenizer (matches training). Each model is loaded via a small `lm_eval.api.model.LM` subclass that reads the architecture cfg from the checkpoint and supports batched length-bucketed loglikelihood. No few-shot override was passed, so every task ran at its harness default. For this suite, including MMLU, that is 0-shot. The MMLU figures are therefore not comparable to the 5-shot numbers usually quoted for that benchmark.

Task-by-task reading. On an 8-task lm-eval-harness suite (full per-task table with stderr in Table 16), MoL and iso-active Dense 0.7B are within one per-model standard error of each other on ARC-Challenge (+0.6, SE 1.3), PIQA (+0.1, SE 1.1), ARC-Easy (−0.7, SE 1.0), and TruthfulQA mc1/mc2 (−1.2/−1.3, SE 1.5, a documented inverse-scaling regime). Two tasks separate by more than one such standard error in MoL's favour, WinoGrande (+5.1, SE 1.4) and HellaSwag (+1.0, SE 0.5), and one against it, MMLU (−0.9, SE 0.4), where all three models sit at or below the 25% chance level. These are per-model standard errors reported by the harness, not standard errors of the paired difference, so they give the noise scale without testing the difference. Against Dense 1.3B, MoL trails on the broad-knowledge tasks (HellaSwag, ARC-Easy, PIQA) by 1.5–4.9 pts, consistent

with the 3.21 PPL gap, and leads on WinoGrande (54.4 vs 51.6). Chance-level MMLU is expected at billion scale with 20B tokens.

Table 16 reports per-task accuracy for all three billion-scale models, adding TruthfulQA mc1/mc2 and the raw `acc` rows for HellaSwag/ARC/PIQA (whose headline metric is `acc_norm`), each with per-cell stderr. The values quoted in Section 5.5 are the `acc_norm` rows for HellaSwag, ARC and PIQA, and `acc` for MMLU, WinoGrande and TruthfulQA.

Table 16: Zero-shot accuracy with stderr (%). Standard errors are the harness's analytic per-model values (sample SD over examples divided by $\sqrt{n}$; MMLU aggregates its subtasks). They are not standard errors of the between-model difference.

Task	Metric	Dense 1.3B	Dense 0.7B	MoL 0.63B/2.08B
MMLU	acc	24.31 ± 0.4	23.98 ± 0.4	23.07 ± 0.4
HellaSwag	acc	33.49 ± 0.5	30.17 ± 0.5	30.53 ± 0.5
HellaSwag	acc_norm	39.71 ± 0.5	34.10 ± 0.5	35.10 ± 0.5
ARC-Easy	acc	59.05 ± 1.0	54.97 ± 1.0	54.00 ± 1.0
ARC-Easy	acc_norm	53.07 ± 1.0	48.86 ± 1.0	48.15 ± 1.0
ARC-Challenge	acc	24.74 ± 1.3	21.33 ± 1.2	23.04 ± 1.2
ARC-Challenge	acc_norm	26.88 ± 1.3	25.43 ± 1.3	26.02 ± 1.3
PIQA	acc	65.94 ± 1.1	63.60 ± 1.1	65.51 ± 1.1
PIQA	acc_norm	65.56 ± 1.1	63.98 ± 1.1	64.09 ± 1.1
WinoGrande	acc	51.62 ± 1.4	49.33 ± 1.4	54.38 ± 1.4
TruthfulQA mc1	acc	21.18 ± 1.4	23.13 ± 1.5	21.91 ± 1.4
TruthfulQA mc2	acc	37.83 ± 1.4	40.89 ± 1.5	39.55 ± 1.4

Two patterns in the per-task results extend the main-body summary:

Inverse scaling on TruthfulQA. Dense 0.7B leads on both mc1 and mc2, a pattern documented at small scale, where the smaller model knows fewer plausible but false alternatives (Lin et al., 2022). MoL sits between the two dense baselines, consistent with its capacity placement.

WinoGrande. MoL 0.63B-active wins this coreference task by 2.76 pts vs Dense 1.3B and 5.05 pts vs iso-active Dense 0.7B, though all three models sit within 4.4 pts of its 50% chance level. Apart from TruthfulQA, where the smaller models lead, it is the only task on which MoL beats Dense 1.3B. We do not attribute this to a mechanism.