WHAT DOES EXPERT SPARSITY BUY? A TWO-SIDED MATCHED-BUDGET STUDY OF MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Expert sparsity exchanges active computation for stored capacity, but its value depends on which resource a comparison holds fixed. We introduce MOE-TAX, a two-sided evaluation framework that compares the same sparse model with both a parameter-matched and an active-compute-matched dense reference. Controlled language-model experiments reveal a clear exchange: at approximately 96M parameters, the sparse configurations in our main sweep incur a 3.8–38.2% perplexity increase at matched parameters, whereas matching active FFN compute yields 2.2–6.1% lower perplexity at a 24.6–42.2% parameter premium. A protocol-controlled scale comparison preserves the parameter-matched gap from 96M to 203M. Expert-count, activation, routing, and balancing-loss ablations identify where the tax arises, and replicate seeds reproduce the configuration ordering. We further isolate a batch-size effect large enough to reverse a compute-matched comparison. Device profiling completes the accounting by measuring latency and peak memory separately from FLOPs. Together, the configuration generator, comparison audit, and released run records make the resource exchange explicit, reproducible, and useful for architecture selection.

1 INTRODUCTION

Mixture-of-Experts (MoE) expands a language model's capacity by routing each token to a subset of expert feed-forward networks. This separation between stored and active parameters underlies sparse scaling from early gated layers (Shazeer et al., 2017) to GShard (Lepikhin et al., 2021), Switch Transformer (Fedus et al., 2022b), and recent large language models (Jiang et al., 2024; DeepSeek-AI, 2024a;b; Qwen Team, 2025). The appeal is straightforward: a token can access a larger collection of weights without evaluating all of them. The evaluation question is less straightforward: *what does sparsity buy when the available budget is held fixed?*

The answer changes with the budget. At fixed total parameters, an MoE layer divides the feed-forward capacity among experts and activates only a fraction per token. At fixed active computation, the same layer can retain more total parameters than a dense alternative. The first comparison measures the quality cost of reducing active computation at a given model size; the second measures the quality benefit of additional stored capacity at a given compute budget. These are complementary measurements of one exchange. Treating either as a complete verdict on MoE obscures what the other resource is paying for.

Scaling-law studies formalize parameters and compute as separate axes (Kaplan et al., 2020; Hoffmann et al., 2022; Clark et al., 2022; Ludziejewski et al., 2024), and matched-compute experiments demonstrate the potential of sparse capacity (Artetxe et al., 2022). We build on this perspective by pairing *each sparse model with two dense references*. One matches total parameters; the other matches active feed-forward FLOPs. Data, tokenizer, token budget, optimizer, schedule, and batch size remain fixed within each comparison. Reading the pair together exposes the parameter premium behind a compute-matched gain and the compute reduction behind a parameter-matched penalty.

Figure 1 previews the result. In the 96M-parameter sweep, a four-expert model reaches 25.20 perplexity, compared with 24.29 for its parameter-matched dense reference and 25.78 for its compute-matched reference. The *same trained MoE* therefore ranks below one dense model and above the

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.

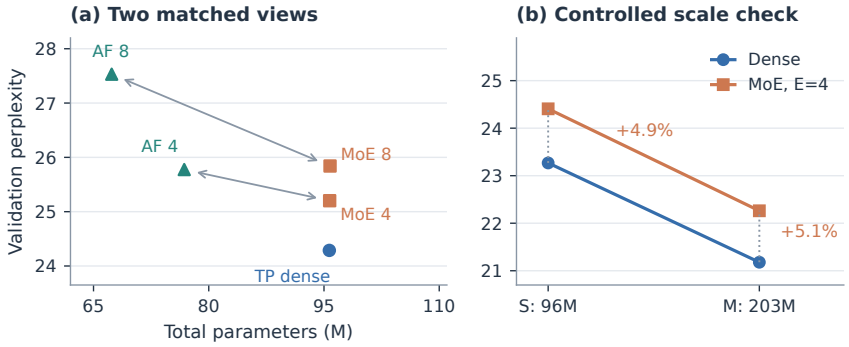


Figure 1: **One sparse model, two budget comparisons.** (a) S-scale runs at batch 32. Blue is the parameter-matched (TP) dense reference; orange denotes MoE; teal denotes active-FFN-compute-matched (AF) dense references. Connecting arrows identify AF pairs; the expert counts are 4 and 8. (b) The common four-expert configuration retains a positive perplexity tax in the batch-16 scale series. Comparisons are made within each panel's protocol.

other. With eight experts, its compute-matched improvement grows to 6.1%, while the parameter premium grows to 42.2%. This two-sided view replaces an ambiguous "MoE versus dense" comparison with an explicit resource trade.

Our study also separates nominal compute savings from measured systems cost. Sparse execution introduces routing, token dispatch, and smaller matrix operations, so a FLOP reduction does not by itself specify latency or peak memory. We profile these quantities directly on a commodity GPU and report the implementation alongside the measurements. This connects controlled quality comparisons to the resources that a deployment actually constrains.

We make three contributions:

- **Two-sided matched accounting.** We provide explicit matching rules, a configuration generator, and parameter/compute audits. The paired references quantify both the 3.8–38.2% parameter-matched tax in the main S-scale sweep and the 2.2–6.1% compute-matched gain with its parameter premium.
- **A controlled decomposition of the trade.** Expert-count, activation, routing, and auxiliary-loss ablations distinguish active capacity from stored capacity. A 96M–203M scale check and five replicate-seed comparisons strengthen the empirical account, while forward profiling measures the systems side through approximately 531M parameters.
- **An executable protocol safeguard.** Changing batch size alone shifts the dense reference by 4.4% perplexity and can reverse an AF comparison. We make comparison groups explicit and audit their batch sizes, parameter budgets, and active FFN widths before generating paper tables and figures.

2 Related Work

Sparse architectures and routing. Sparsely gated layers established conditional expert computation for language models (Shazeer et al., 2017). GShard and Switch Transformer developed large-scale top-2 and top-1 routing with auxiliary balancing (Lepikhin et al., 2021; Fedus et al., 2022b;a), while ST-MoE studied training stability (Zoph et al., 2022). Alternative assignment mechanisms include expert-choice routing (Zhou et al., 2022), balanced assignment in BASE (Lewis et al., 2021), and static hash layers (Roller et al., 2021). Representation collapse and expert specialization (Chi et al., 2022; Dai et al., 2024), vision MoE (Riquelme et al., 2021), sparse upcycling (Komatsuzaki et al., 2023), and instruction tuning (Shen et al., 2024) broaden the design space. Our experiments hold the decoder and training recipe fixed while varying four choices within that space: expert count, activation count, routing rule, and balancing strength.

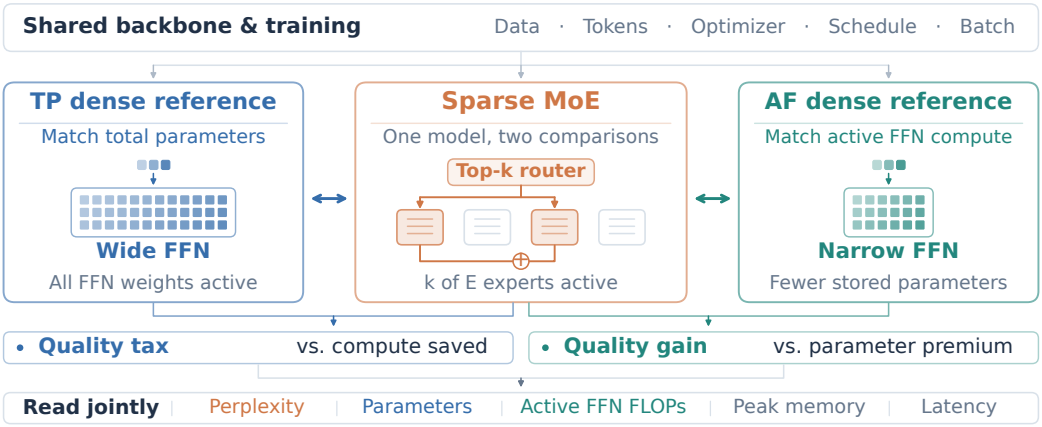


Figure 2: **The MoE-Tax framework.** The same sparse model is compared with a wide dense FFN at matched total parameters and a narrower dense FFN at matched active FFN compute. Colored expert blocks indicate the selected subset; the expert bank is schematic. The decoder and training protocol are shared. Quality and all four resource axes are read together.

Scaling and matched budgets. GLaM and recent sparse language models demonstrate the practical value of conditional capacity (Du et al., 2022; Jiang et al., 2024; DeepSeek-AI, 2024a;b; Qwen Team, 2025). Clark et al. (2022) model routed scaling using separate parameter and compute axes, and Ludziejewski et al. (2024) extend the analysis to fine-grained experts. Artetxe et al. (2022) compare dense and sparse language models under matched compute. These studies motivate our central distinction: compute matching and parameter matching answer different questions. We couple the two around the same MoE configuration and add a common training protocol, batch-size checks, and measured systems costs.

Systems support for sparsity. MegaBlocks (Gale et al., 2023), Tutel (Hwang et al., 2023), and DeepSpeed-MoE (Rajbhandari et al., 2022) improve sparse execution through specialized kernels, scheduling, and parallelism. Our measurements use an explicit expert-loop implementation. They characterize that execution path and show why active FLOPs and latency should be reported separately. The budget matching rules remain applicable when the kernel implementation changes.

Reliable comparisons and sparsity dividends. Benchmarking work emphasizes that initialization, data order, and training choices can materially affect model comparisons (Bouthillier et al., 2021; Dodge et al., 2020). We examine one such choice directly: batch size at a fixed token budget. The same concern motivates our replicate-seed check and explicit comparison groups. Related studies of category-conditioned routing (Wang & Lei, 2026) and activity–energy tradeoffs in spiking networks (Wang, 2026) also distinguish sparsity from its realized value. Our contribution is a quantitative accounting of that distinction for controlled language-model training.

3 Method

MoE-Tax uses a shared GPT decoder and two dense references to expose the resource exchange in a sparse model (Figure 2). We denote parameter matching by TP and active feed-forward compute matching by AF. Each comparison holds the training protocol fixed; the matching rule determines which resource is equalized and which is allowed to change.

3.1 Two complementary matching rules

Let H be the model width, L the number of Transformer layers, h_d the dense FFN hidden width, h_m the width of each expert, E the expert count, and k the number of selected experts. The dominant weight terms in one dense FFN and one learned-router MoE layer are $2Hh_d$ and $2EHh_m + HE$, respectively. The shared attention and embedding weights cancel when matching the two models.

Parameter matching (TP). Equating these terms yields

$$h_m^{\mathrm{TP}} = \frac{2h_d - E}{2E}. \tag{1}$$

The generator aligns the expert width to a multiple of 16; we then report the realized parameter count and check its deviation from the dense reference. For $H = 768$, $h_d = 3072$, and $E = 8$, the aligned width is $h_m = 384$. With top-2 routing, active expert FFN compute is one quarter of dense FFN compute. For $E = 4$, the aligned width is 768 and the fraction is one half.

Active-compute matching (AF). Forward FFN FLOPs per token are $4Hh_d$ for dense and $4kHh_m$ for MoE. Keeping the TP-derived MoE fixed, its AF dense reference therefore uses

$$h_d^{\mathrm{AF}} = kh_m^{\mathrm{TP}}. \tag{2}$$

For $E = 8, k = 2$, this gives a dense width of 768 and 67.4M total parameters, compared with 95.8M for MoE. The shared decoder remains unchanged; only its FFN width differs. AF denotes *active FFN* matching throughout: routing and dispatch overhead are captured by device measurements, and FFN savings are not presented as equal-sized reductions in whole-model FLOPs.

We express quality differences using the reference named by the comparison:

$$\tau_{\mathrm{TP}} = 100\left(\frac{\mathrm{PPL}_{\mathrm{MoE}}}{\mathrm{PPL}_{\mathrm{TP}}} - 1\right), \qquad g_{\mathrm{AF}} = 100\left(1 - \frac{\mathrm{PPL}_{\mathrm{MoE}}}{\mathrm{PPL}_{\mathrm{AF}}}\right). \tag{3}$$

A positive τ_{TP} is a parameter-matched tax; a positive g_{AF} is a compute-matched gain. The latter is reported with the parameter premium $100(P_{\mathrm{MoE}}/P_{\mathrm{AF}} - 1)$ rather than in isolation.

3.2 Shared models and training protocol

All models use a pre-norm GPT decoder with LayerNorm, causal self-attention, learned positional embeddings, tied input/output embeddings, and GELU FFNs. The MoE uses a linear router, selects the top-k logits, and normalizes the selected logits with softmax before combining expert outputs. We compare learned routing with an auxiliary load-balancing objective, learned routing without that objective, and static position-based hash routing. The reference auxiliary coefficient is $\lambda = 0.01$; one ablation uses 0.001.

Training uses WikiText-103 (Merity et al., 2017), the GPT-2 tokenizer, sequence length 512, and three epochs over 229,926 sequences, approximately 353M tokens. AdamW uses a peak learning rate of 3×10^{-4} and weight decay 0.1, with 200 warmup steps followed by cosine decay. We fix data order, optimizer settings, schedule rule, and seed within each paired comparison. Batch 32 is used for the S-scale sweep and its AF references; batch 16 is used for the controlled scale series and two-expert ablations. Replicate experiments change only the seed. Full settings are in Appendix G.

The controlled quality study covers S ($H = 768, L = 8$, approximately 96M parameters) and M ($H = 1024, L = 12$, approximately 203M). Device profiling also includes L ($H = 1536, L = 16$, approximately 531M). Validation perplexity is evaluated on the same held-out sequences; the reported value is the best recorded validation perplexity. For expert loads, we report the coefficient of variation (CV), with lower values indicating more uniform assignments.

3.3 Configuration generation and resource accounting

The generator implements Equations equation 1–equation 2 and emits matched model widths. The publication pipeline reads saved JSON records, checks the batch size and parameter tolerance of TP groups, and checks $h_d = kh_m$ for AF pairs. Tables, charts, and numeric summaries use those same records. A 2% TP tolerance accommodates width alignment and small implementation terms; the reported S-scale configurations differ by much less than this threshold.

We keep four resource quantities separate: total parameters, active FFN FLOPs, peak device memory, and forward latency. TP fixes the first approximately and allows the second to vary; AF fixes the second and allows the first to vary. Peak memory and latency are measured outcomes in both cases. Parameter matching does not assert identical activation memory, and active FLOP matching does not assert identical runtime. This separation is the basis of the two-sided account.

Table 1: **Parameter-matched comparison at S scale.** Batch 32, three epochs, approximately 353M tokens. Δ PPL is relative to dense; CV measures expert-load variation. Table 5 contains the complete MoE sweep.

Configuration	Params (M)	PPL	Δ PPL	Load CV
Dense reference	95.7	**24.29**	—	—
E=4, k=2, λ=0.01	95.7	25.20	+3.8%	0.21
E=8, k=2, λ=0.01	95.8	25.84	+6.4%	0.24
E=8, k=2, hash	95.8	33.56	+38.2%	1.15

Table 2: **Active-FFN-compute-matched comparison.** Batch 32, top-2 routing. Each dense reference has the same active FFN FLOPs as its MoE counterpart. Parameter counts are in millions; the premium is relative to AF dense.

Experts	MoE PPL	Dense PPL	PPL gain	Params: MoE / dense	Premium
E=4	25.20	25.78	2.2%	95.7 / 76.8	+24.6%
E=8	25.84	27.53	6.1%	95.8 / 67.4	+42.2%

4 EXPERIMENTS

We first quantify both sides of the budget exchange, then examine protocol, scale, architectural choices, reproducibility, and device performance. Every quality comparison identifies its reference and batch size. The run inventory and training curves appear in Appendices I and H.

4.1 PARAMETER MATCHING EXPOSES THE SPARSITY TAX

Table 1 summarizes the main S-scale comparison at batch 32. Dense reaches 24.29 perplexity with 95.7M parameters. The four-expert MoE reaches 25.20, a 3.8% tax; the eight-expert reference configuration reaches 25.84, a 6.4% tax. Static hash routing increases the tax to 38.2%. All seven sparse configurations in the full sweep have higher perplexity than parameter-matched dense (Appendix C).

This comparison isolates the cost of reducing the active fraction at a fixed stored-parameter budget. It does not deny that conditional capacity can improve quality at fixed compute; that is the question addressed by the AF reference. Instead, TP establishes the baseline against which a compute saving must be valued. In these runs, distributing the same parameter budget across more experts does not preserve the quality of a fully active dense FFN.

4.2 COMPUTE MATCHING REVEALS THE PARAMETER PREMIUM

The same MoE models outperform their AF dense references (Table 2). At $E = 4$, perplexity falls from 25.78 to 25.20, a 2.2% gain, while parameters rise from 76.8M to 95.7M, a 24.6% premium. At $E = 8$, the gain is 6.1%, from 27.53 to 25.84, with a 42.2% premium. These are the same sparse-model results used in TP; changing the dense reference exposes the other side of the exchange without retraining MoE.

Reading both tables together is more informative than either alone. Narrowing the dense FFN from its TP width to its AF width saves 19.7% of the full model's parameters for the four-expert pair and 29.6% for the eight-expert pair, while increasing perplexity by 6.1% and 13.4%, respectively. MoE uses the additional stored capacity to recover part of that quality at the same active FFN compute. The result is an explicit price for the sparse gain: a 24.6–42.2% parameter premium buys a 2.2–6.1% perplexity reduction against the narrower dense model.

The practical reading depends on the constrained resource. A parameter budget selects the TP comparison; an active FFN compute budget selects AF and must also account for the parameter premium. A latency budget requires direct profiling. Our framework keeps these decisions separate while making their connection visible in one set of models.

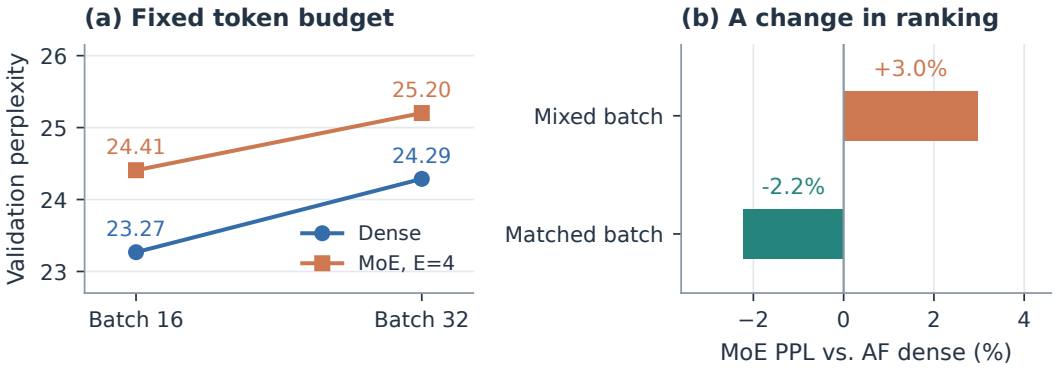


Figure 3: **A protocol choice can reverse the apparent architecture result.** (a) Batch 16 versus 32 at the same token budget. (b) The four-expert MoE at batch 32 appears 3.0% worse than AF dense at batch 16, but 2.2% better when both use batch 32. Positive values mean higher MoE perplexity. The mixed-batch case is shown only as a diagnostic.

Table 3: **Scale comparison with batch 16 and 353M tokens fixed.** Every MoE is compared with dense at the same scale. The four-expert top-2 configuration is shared across S and M. Two-expert controls are reported separately in Figure 5.

Scale	Configuration	Params (M)	PPL	Δ PPL
S	Dense	95.7	23.27	—
S	MoE E=4	95.7	24.41	+4.9%
M	Dense	203.1	21.18	—
M	MoE E=4	203.2	22.26	+5.1%
M	MoE E=8	203.3	22.62	+6.8%

4.3 Batch control changes the comparison

At fixed tokens, batch size changes both the number of optimizer updates and the gradient statistics. Figure 3 measures this effect using the same architecture and data. The S-scale dense reference reaches 23.27 perplexity at batch 16 and 24.29 at batch 32, a 4.4% change. The four-expert MoE moves from 24.41 to 25.20. These shifts matter when the architecture-level gains themselves are a few percent.

The AF comparison gives a concrete sign reversal. Comparing batch-32 MoE (25.20) with an earlier batch-16 AF dense run (24.48) produces an apparent 3.0% MoE penalty. Holding batch at 32 instead gives the 2.2% gain in Table 2. The earlier record is retained as a diagnostic, not used as a matched reference. This example shows why a fixed token budget alone is not a sufficient training-protocol specification.

We construct named comparison groups and require a common batch size, epoch count, tokenizer, and schedule rule within each group. The same mechanism checks that TP groups have one dense reference and AF pairs have equal active FFN widths. The comparison audit is part of figure generation, so a mixed-batch group cannot silently enter the resulting matched table or chart. Reporting these controls makes small quality differences interpretable and repeatable.

4.4 Scale under a fixed protocol

Table 3 holds batch at 16 and training at three epochs while increasing model size from approximately 96M to 203M parameters. For the common four-expert, top-2 configuration, the tax is 4.9% at S and 5.1% at M. Dense perplexities are 23.27 and 21.18, and MoE perplexities are 24.41 and 22.26. Both architectures improve with scale while the relative TP gap persists in the controlled pair. The eight-expert M-scale model pays a 6.8% tax.

Table 4: **Public-model context on the same evaluation split.** These models are not budget-matched; the controlled comparisons are in Tables 1–3. Active parameter counts are model-level counts, not the FFN-FLOP metric used for AF matching.

Model	Total parameters	Active/token	PPL
Qwen3-30B-A3B	30B	∼3.3B	**5.02**
Qwen3-8B	8B	8B	6.22

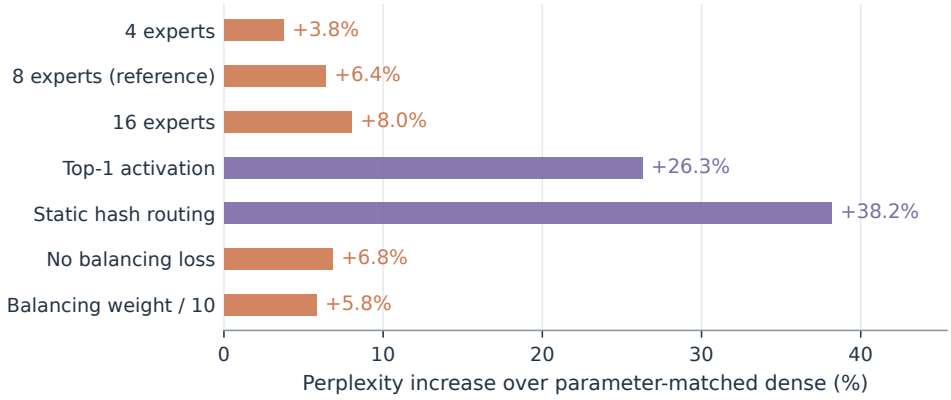


Figure 4: **Decomposing the parameter-matched tax.** S scale, batch 32. The reference MoE uses $E = 8$, $k = 2$, learned routing, and $\lambda = 0.01$; each row changes the named choice. All values are relative to the same 95.7M-parameter dense reference. Lower is better.

This separates a scale effect from a protocol change: the 4.4% batch effect measured above is comparable to the tax itself, whereas the controlled scale comparison preserves batch size and data order. Figure 1(b) shows the common configuration directly; Appendix C includes all available configurations without filling in unmeasured scale/configuration pairs.

Context at public-model scale. We also evaluate Qwen3-30B-A3B and Qwen3-8B on the same held-out WikiText-103 sequences. The sparse model reaches 5.02 perplexity versus 6.22 for dense, approximately 19.3% lower, while activating about 3.3B versus 8B parameters per token (Table 4). Its 30B total parameters represent a 3.75-fold stored-capacity increase. This public-model comparison illustrates why both active and total capacity matter. It is reported as context, with architecture and pretraining differences kept distinct from our controlled TP/AF results.

4.5 Where the tax comes from

Figure 4 varies four design choices within the batch-32 TP sweep. Unless changed explicitly, the reference is eight experts, top-2 routing, and auxiliary coefficient 0.01. Full values, including load CV, appear in Table 5.

Expert count and active capacity. With total parameters and top-k fixed, the tax rises from 3.8% at four experts to 6.4% at eight and 8.0% at sixteen. More experts make each expert narrower and reduce the active fraction. This experiment identifies the cost of that joint change under TP matching, complementing studies that vary expert granularity under other budget rules (Dai et al., 2024; Ludziejewski et al., 2024). Reducing eight-expert activation from top-2 to top-1 increases the tax to 26.3%. In this regime, retaining more active feed-forward capacity is the strongest of the tested design choices.

Learned routing contributes substantial quality. Replacing the learned router with static position hashing raises perplexity from 25.84 to 33.56 at the same stored-parameter count. Learned routing therefore reduces perplexity by 23.0% relative to this hash baseline; the hash model's 38.2%

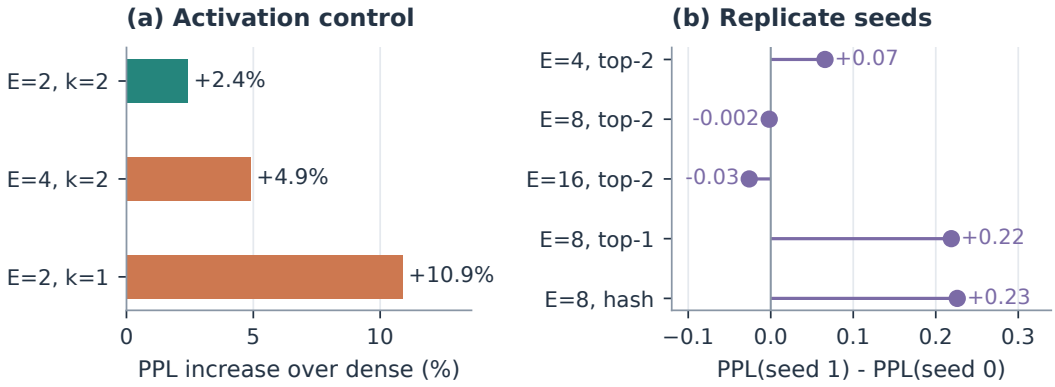


Figure 5: **Controls supporting the decomposition.** (a) Batch-16 S-scale activation controls, each compared with the same dense reference. $E = k = 2$ activates the entire expert bank. (b) Five batch-32 MoE configurations repeated with seed 1. Differences are seed-1 minus seed-0 perplexity, not confidence intervals; the maximum absolute difference is 0.23. Tables 6 and 7 give exact values.

tax is measured against *dense*, a different denominator. This distinction separates the value of learning the assignment rule from the overall parameter-matched sparsity penalty.

Balancing and quality need separate objectives. Removing the auxiliary objective increases load CV from 0.24 to 0.80, while perplexity changes from 25.84 to 25.95, approximately 0.4%. Reducing its weight tenfold gives 25.70 perplexity and CV 0.27. Load uniformity and language quality therefore need not move together: the best balance score does not identify the best perplexity in this sweep. These results motivate treating balancing strength as a quality/dispatch tradeoff rather than selecting it from load statistics alone.

4.6 Activation controls and replicate seeds

Two additional experiments make the decomposition more precise (Figure 5). First, the batch-16 two-expert control holds the expert bank fixed and changes the active count. Activating both experts reaches 23.83 perplexity, a 2.4% tax over the 23.27 dense reference. Activating one reaches 25.80, a 10.9% tax. The four-expert top-2 configuration lies between them at 24.41 and 4.9%. Activating all experts retains input-dependent weighted combination but removes sparse expert selection; the smaller gap distinguishes the gated expert structure from the larger cost of restricting active capacity.

Second, five batch-32 MoE configurations reproduce the same quality ordering under a second seed: four-expert top-2, eight-expert top-2, sixteen-expert top-2, eight-expert top-1, and static hash routing. Their absolute changes range from less than 0.01 to 0.23 perplexity. The routing and activation differences are substantially larger than these replicate changes. These checks support the ordering among the tested configurations and complement the protocol controls; they do not substitute batch matching with seed averaging.

4.7 Measured systems cost

We profile the forward pass on a single RTX 5090 using sequence length 512 and batch 32, averaging ten timed iterations after warmup. The implementation dispatches tokens through an explicit expert loop, and the benchmark synchronizes the device around timing. Figure 6 reports latency; full parameter, FLOP, and peak-memory accounting appears in Table 8.

At S scale, TP dense takes 183.8 ms, compared with 505.9 ms for four experts and 856.9 ms for eight: 2.8 and 4.7 times dense latency. Their active FFN compute is respectively one half and one quarter of TP dense. The AF dense references take 154.0 and 139.0 ms. The nominal FFN saving does not become a latency saving on this execution path at S scale. Peak allocated memory is approximately 3.82–3.83 GB for the TP models and 3.71–3.75 GB for AF dense.

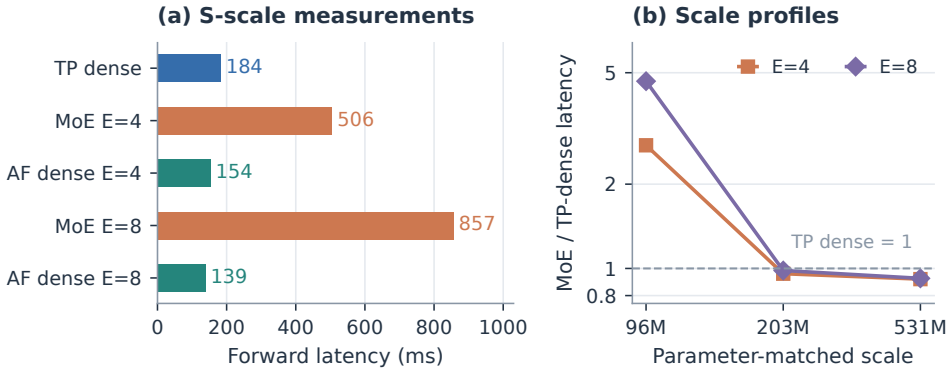


Figure 6: **FLOPs and runtime are different budget axes.** (a) Recorded S-scale forward latency for TP dense, MoE, and AF dense references. (b) MoE latency normalized to TP dense at each scale; the dashed line denotes equal latency. These measurements characterize the unfused expert-loop implementation. Table 8 contains the full recorded values.

The recorded larger-scale profiles show a different balance. At M, four- and eight-expert forwards take 160.0 and 164.2 ms versus 167.1 ms for TP dense. At L, they take 194.9 and 196.2 ms versus 212.8 ms. In both profiles MoE is close to or faster than TP dense, while narrower AF dense references remain faster. Joint accounting retains this distinction: the same sparse model can be favorable against one resource-matched reference and unfavorable against another. Against AF dense, MoE remains slower in the recorded profiles: 3.3–6.2$\times$ at S and 1.2–1.4$\times$ at M/L. Thus a favorable TP latency does not imply a favorable AF latency; the matched reference remains essential to interpreting a systems result.

Routing, dispatch, matrix shape, and kernel choice affect how FFN savings translate to runtime. Specialized systems such as MegaBlocks (Gale et al., 2023), Tutel (Hwang et al., 2023), and DeepSpeed-MoE (Rajbhandari et al., 2022) target these costs. Our profiles motivate measuring the implementation actually used rather than assigning a latency benefit from the active-parameter fraction. Changing kernels can change the systems exchange without changing the TP and AF definitions.

5 Discussion

The useful output of a matched study is the *triplet*: TP dense, a specific MoE configuration, and its AF dense reference, trained with the same protocol. TP answers a stored-capacity question; AF answers an active-compute question; device profiles answer the systems question. Reporting all three models makes the parameter premium, compute saving, and quality difference reconstructible. The scale, activation, and seed controls support the observed trade, while the batch diagnostic prevents a moving reference from being mistaken for an architecture effect. The same accounting can evaluate new routing methods and sparse kernels using the released generator and comparison audit. A compute-matched gain should accompany its parameter premium; a parameter-matched result should specify the active computation. This reporting rule separates capacity efficiency from execution efficiency without prescribing one architecture for every deployment.

6 Conclusion

MoE-Tax makes expert sparsity's resource exchange explicit. The same sparse models pay a quality tax at matched parameters and gain quality at matched active FFN compute, with a measurable parameter premium. Architecture ablations, activation controls, replicate seeds, and a fixed-protocol scale series explain and support this result. The batch-size experiment and device profiles show why protocol and runtime must be measured alongside nominal compute. Reporting both matched references turns an architecture ranking into a reproducible, budget-specific decision.

Reproducibility Statement

The released configuration generator implements the TP and AF width rules in Equations equation 1–equation 2. Training records contain model settings, batch size, seed, realized parameter count, and validation curves. The publication script reconstructs each comparison from these records, asserts consistent batch sizes and parameter budgets, verifies AF width equality, and generates tables, plots, and a numeric audit together. The audit lists every source record. Appendices A–G specify accounting and training; Appendices D and I provide replicate results and the run inventory. Historical mixed-batch records are labeled as diagnostics and excluded from matched quality tables. No training result is inferred from a device-only profile.

Ethics Statement

This study evaluates language-model architectures using public datasets and publicly released model checkpoints, in compliance with the licenses and usage terms of all datasets and models involved. It involves no human subjects and no private or personally identifiable user data. All generated outputs are confined to academic evaluation and do not involve harmful content. The results concern resource allocation and evaluation methodology; controlled quality findings are attached to the stated model scales, dataset, and training budget, while public-model context and device profiling are identified separately so their evidence can be interpreted with the appropriate scope. The authors declare no competing interests.

AI Use Statement

Generative AI tools were used solely for language polishing and grammar checking of the manuscript. The study design, implementation, experiments, data analysis, and the writing of all technical content were conducted entirely by the authors, who reviewed and verified all text, figures, code, data, and conclusions and take full responsibility for the integrity of the work.

References

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giridharan Anantharaman, Xian Li, Shuohui Chen, Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Brian O'Horo, Jeffrey Wang, Luke Zettlemoyer, Mona Diab, Zornitsa Kozareva, and Veselin Stoyanov. Efficient large scale language modeling with mixtures of experts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11699–11732. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.804. URL https://aclanthology.org/2022.emnlp-main.804/.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Naz Sepah, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Dmitriy Serdyuk, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems*, volume 3, pp. 747–769, 2021. URL https://proceedings.mlsys.org/paper_files/paper/2021/hash/0184b0cd3cfb185989f858a1d9f5c1eb-Abstract.html.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, Heyan Huang, and Furu Wei. On the representation collapse of sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 35, pp. 34600–34613, 2022. doi: 10.52202/068431-2507. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/df4f371f1f89ec8ba5014b3310578048-Abstract.html.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den

Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4057–4086. PMLR, 2022. URL https://proceedings.mlr.press/v162/clark22a.html.

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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.70. URL https://aclanthology.org/2024.acl-long.70/.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a. doi: 10.48550/arXiv.2405.04434. URL https://arxiv.org/abs/2405.04434.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b. doi: 10.48550/arXiv.2412.19437. URL https://arxiv.org/abs/2412.19437.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020. doi: 10.48550/arXiv.2002.06305. URL https://arxiv.org/abs/2002.06305.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 2022. URL https://proceedings.mlr.press/v162/du22c.html.

William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022a. doi: 10.48550/arXiv.2209.01667. URL https://arxiv.org/abs/2209.01667.

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, 2022b. URL https://www.jmlr.org/papers/v23/21-0998.html.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems*, volume 5, pp. 288–304, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/5a54f79333768effe7e8927bcccffe40-Abstract-mlsys2023.html.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030, 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, Joe 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, pp. 269–287, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/5616d34cf8ff73942cfd5aa922842556-Abstract-mlsys2023.html.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. doi: 10.48550/arXiv.2401.04088. URL https://arxiv.org/abs/2401.04088.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. doi: 10.48550/arXiv.2001.08361. URL https://arxiv.org/abs/2001.08361.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme, 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. URL https://openreview.net/forum?id=T5nUQDrM4u.

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. URL https://openreview.net/forum?id=qrwe7XHmYb.

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*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6265–6274. PMLR, 2021. URL https://proceedings.mlr.press/v139/lewis21a.html.

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 *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33270–33288. PMLR, 2024. URL https://proceedings.mlr.press/v235/ludziejewski24a.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18332–18346. PMLR, 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8583–8595, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17555–17566, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/92bf5e6240737e0326ea59846a83e076-Abstract.html.

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. URL https://openreview.net/forum?id=B1ckMDqlg.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. Mixture-of-experts meets instruction tuning: A winning combination for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6mLjDwYte5.

Boyao Wang and Zhihan Lei. SpecDrop: Parameter-free category-conditioned routing for modular specialization. *arXiv preprint arXiv:2608.04084*, 2026. doi: 10.48550/arXiv.2608.04084. URL https://arxiv.org/abs/2608.04084.

Zeyu Wang. The sparsity ceiling: Where spiking networks can and cannot trade activity for energy. *arXiv preprint arXiv:2607.26648*, 2026. doi: 10.48550/arXiv.2607.26648. URL https://arxiv.org/abs/2607.26648.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, pp. 7103–7114, 2022. doi: 10.52202/068431-0515. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/2f00ecd787b432c1d36f3de9800728eb-Abstract-Conference.html.

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. doi: 10.48550/arXiv.2202.08906. URL https://arxiv.org/abs/2202.08906.

A PROTOCOL AND COMPARISON GROUPS

All controlled quality runs use WikiText-103 and GPT-2 tokenization. The S-scale TP sweep and both AF pairs use batch 32, three epochs, and seed 0. The S/M scale series and two-expert controls use batch 16, three epochs, and seed 0. The seed check repeats five S-scale batch-32 MoE configurations with seed 1. These groups answer different questions and are never merged into a cross-batch architecture ranking. The batch-16 AF dense record is retained only for the explicit mixed-batch diagnostic in Figure 3.

Total parameters count stored model weights; active FFN FLOPs count selected expert feed-forward work per token. Peak device memory includes allocations arising during execution and need not equal weight memory. Latency includes the actual forward execution path. TP matching equalizes parameters approximately; AF matching equalizes the stated FFN compute term. Neither rule constrains measured runtime or peak memory to be equal.

B DERIVATION AND IMPLEMENTATION OF MATCHING

Let P_0 collect shared attention and embedding terms. The dominant parameter counts are $P_D = P_0 + 2LHh_d$ and $P_M = P_0 + L(2EHh_m + HE)$. Solving $P_D = P_M$ gives Equation equation 1. The generator rounds h_m to a multiple of 16 and the publication audit verifies realized parameter counts. Biases and normalization parameters are included in the realized count rather than hidden inside the approximate width equation.

Each FFN contains two matrix multiplications. Counting multiply and add as two operations gives $4Hh_d$ forward FLOPs for dense and $4kHh_m$ for the selected experts. Equating them yields Equation equation 2. The matched reference shares H, L, vocabulary, sequence length, and attention with MoE. Active FFN ratios are relative to TP dense: 1 for TP dense, 0.5 for the four-expert top-2 pair, and 0.25 for the eight-expert top-2 pair. Router and dispatch costs remain part of measured runtime.

C COMPLETE PARAMETER-MATCHED RESULTS

Table 5 contains every configuration in the batch-32 S-scale sweep. Figure 4 presents the same values as deltas against dense. M-scale results and available batch-16 configuration/scale pairs appear in Figures 7 and 8.

Table 5: Complete S-scale TP sweep, batch 32. Params are realized model weights; load CV is the last recorded coefficient of variation. All quality differences use the same dense reference.

Configuration	Params (M)	PPL	Δ PPL	Load CV
Dense reference	95.7	**24.29**	—	—
E=4, k=2, λ=0.01	95.7	25.20	+3.8%	0.21
E=8, k=2, λ=0.01	95.8	25.84	+6.4%	0.24
E=16, k=2, λ=0.01	95.9	26.24	+8.0%	0.26
E=8, k=1, λ=0.01	95.8	30.67	+26.3%	0.01
E=8, k=2, hash	95.8	33.56	+38.2%	1.15
E=8, k=2, no auxiliary loss	95.8	25.95	+6.8%	0.80
E=8, k=2, λ=0.001	95.8	25.70	+5.8%	0.27

D ACTIVATION AND SEED CONTROLS IN FULL

Tables 6 and 7 give the exact values underlying the activation and replicate-seed experiments in the main text. With $E = k = 2$, all experts execute, but the learned gate still forms an input-dependent weighted sum. This is an all-experts-active control, not an algebraic identity with the reference dense FFN.

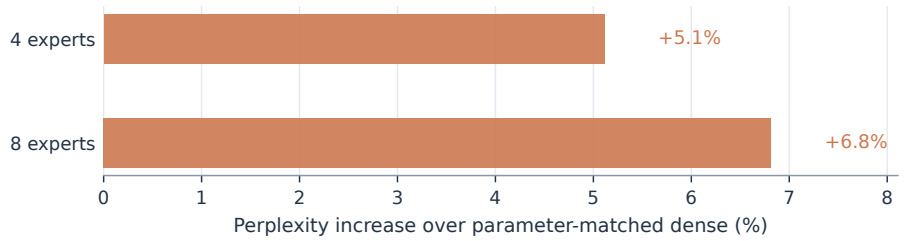


Figure 7: M-scale parameter-matched tax at batch 16, top-2 routing, and auxiliary coefficient 0.01. The reference has 203.1M parameters and 21.18 perplexity.

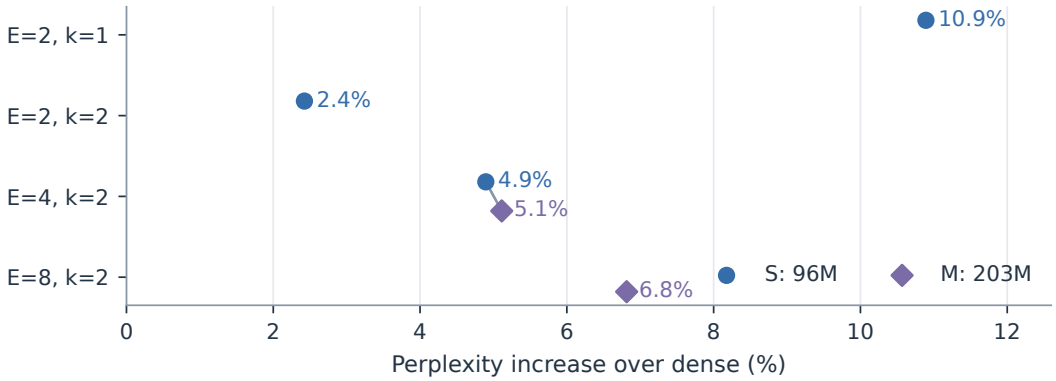


Figure 8: Available batch-16 scale/configuration pairs. Each point uses the dense reference at its own scale. Only $E = 4, k = 2$ is available at both S and M; the short connector identifies that common configuration. Missing pairs are not interpolated or represented by zeros.

E Systems measurements and implementation

The benchmark uses an explicit per-expert loop and standard dense operations, with three warmup forwards and ten timed forwards under inference mode. Device synchronization bounds the timing interval, and peak allocated device memory is reset before the timed region. Table 8 preserves the recorded profiles across all three scales. The L-scale profile is a systems measurement; it does not create a matched L-scale quality comparison.

MegaBlocks (Gale et al., 2023) uses block-sparse computation to organize expert work, Tutel (Hwang et al., 2023) optimizes MoE execution and parallelism, and DeepSpeed-MoE (Rajbhandari et al., 2022) develops efficient sparse inference. These systems offer different execution paths for the same budget accounting. Our profiles are measurements of the included implementation, rather than predictions of those optimized systems.

F Public-model evaluation

The public checkpoints are Qwen3-30B-A3B and Qwen3-8B. Their saved evaluation records report perplexities of 5.0193 and 6.2222 on the same WikiText-103 validation split with sequence length 512. The sparse model's total and active parameter counts are approximately 30B and 3.3B; dense has 8B active parameters. Table 4 presents these as context for parameter/compute accounting. Differences in depth, width, and pretraining distinguish this comparison from the controlled from-scratch experiments.

Table 6: S-scale activation control, batch 16, three epochs. The $E = 2, k = 1$ and $E = 2, k = 2$ models share the same expert bank size.

Configuration	PPL	Δ PPL
Dense	23.27	—
E=2, k=2, λ=0.01	23.83	+2.4%
E=4, k=2, λ=0.01	24.41	+4.9%
E=2, k=1, λ=0.01	25.80	+10.9%

Table 7: Replicate MoE seeds at S scale, batch 32. The last column is an absolute PPL difference. The largest change is 0.23; the ordering among the five configurations is unchanged.

Configuration	Seed 0	Seed 1	$\lvert\Delta\rvert$
E=4, top-2	25.20	25.27	0.07
E=8, top-2	25.84	25.84	0.00
E=16, top-2	26.24	26.21	0.03
E=8, top-1	30.67	30.89	0.22
E=8, hash	33.56	33.78	0.23

G Training settings and budget check

AdamW uses $\beta = (0.9, 0.95)$, weight decay 0.1, and peak learning rate 3×10^{-4}. Linear warmup lasts 200 steps, followed by cosine decay; gradient norm is clipped at 1.0. Sequences contain 512 tokens, and validation perplexity is computed on 400 held-out sequences. The three-epoch budget contains approximately 353M tokens. Batch size is part of the comparison group; seed 0 is used except for the explicit seed-1 replicates.

For the recorded numerical-equivalence check, non-reentrant gradient checkpointing preserves reported language loss, auxiliary loss, and validation perplexity (7.724, 0.0808, and 2485.31 in the short diagnostic). This check concerns numerical execution, not a trained quality result. The extended-budget dense run reaches 21.10 perplexity after six epochs versus 23.27 after three at S scale and batch 16. It is listed as a budget check, separately from three-epoch matched comparisons.

H Supplementary quality views and learning curves

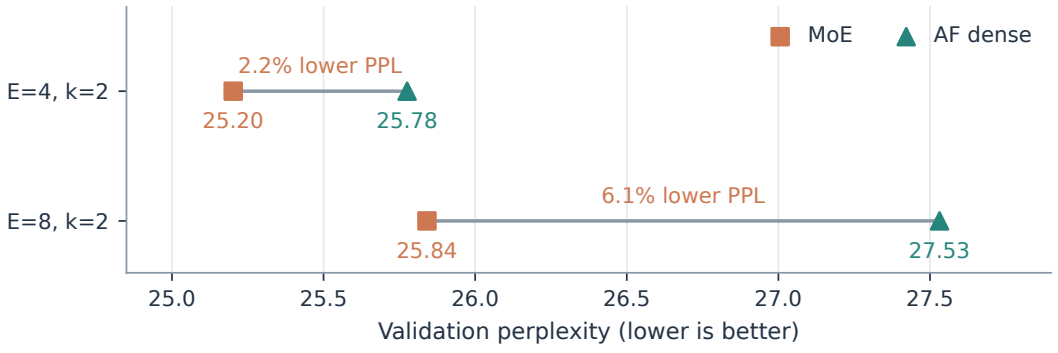


Figure 9: AF quality pairs at batch 32. Each connector joins a MoE and its active-FFN-compute-matched dense reference. Both favor MoE on perplexity; Table 2 reports the accompanying parameter premiums.

Table 8: Full recorded forward profiles. Batch 32, sequence length 512, RTX 5090. FFN FLOPs are normalized to TP dense at the same scale. Latency is the mean of ten timed forwards, and MB is decimal megabytes.

Scale	Configuration	Params (M)	FFN FLOPs	Latency (ms)	Peak MB
S	TP dense	95.7	1.00	183.8	3821
S	MoE E=4	95.7	0.50	505.9	3822
S	AF dense E=4	76.8	0.50	154.0	3746
S	MoE E=8	95.8	0.25	856.9	3831
S	AF dense E=8	67.4	0.25	139.0	3708
M	TP dense	203.1	1.00	167.1	4288
M	MoE E=4	203.2	0.50	160.0	4288
M	AF dense E=4	152.8	0.50	135.9	4086
M	MoE E=8	203.3	0.25	164.2	4288
M	AF dense E=8	127.6	0.25	120.3	3986
L	TP dense	531.3	1.00	212.8	3940
L	MoE E=4	531.5	0.50	194.9	3941
L	AF dense E=4	380.2	0.50	161.8	3336
L	MoE E=8	531.7	0.25	196.2	3941
L	AF dense E=8	304.7	0.25	137.2	3034

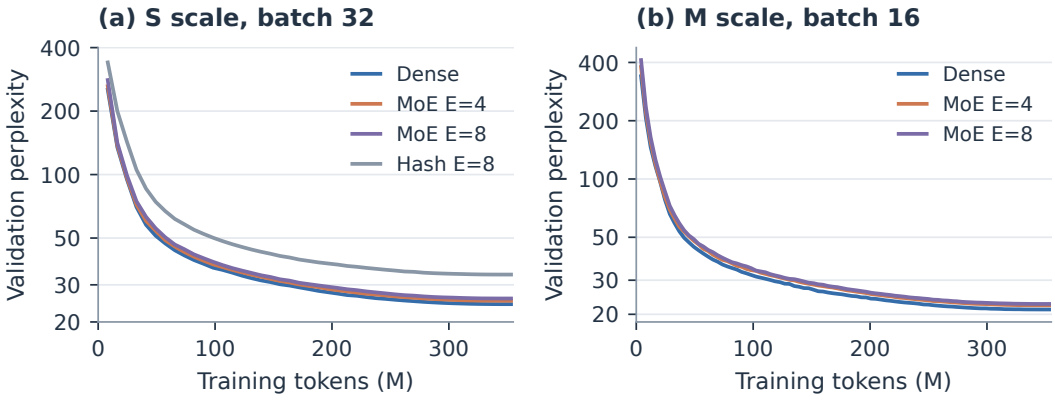


Figure 10: Validation trajectories from the saved training records. The horizontal axis converts optimizer steps to processed tokens using each run's batch size and sequence length. Panels retain their own batch protocols; the vertical axis is logarithmic.

I Complete run inventory

Table 9 lists the training records used by the analysis, including replicate seeds, the longer-budget dense check, the available L-scale dense reference, and the historical mixed-batch diagnostic. "Comparison" identifies records used in controlled groups; it does not imply that a record appears in only one table. The same MoE is intentionally reused across TP and AF views. The source-file mapping accompanies the machine-readable numeric audit.

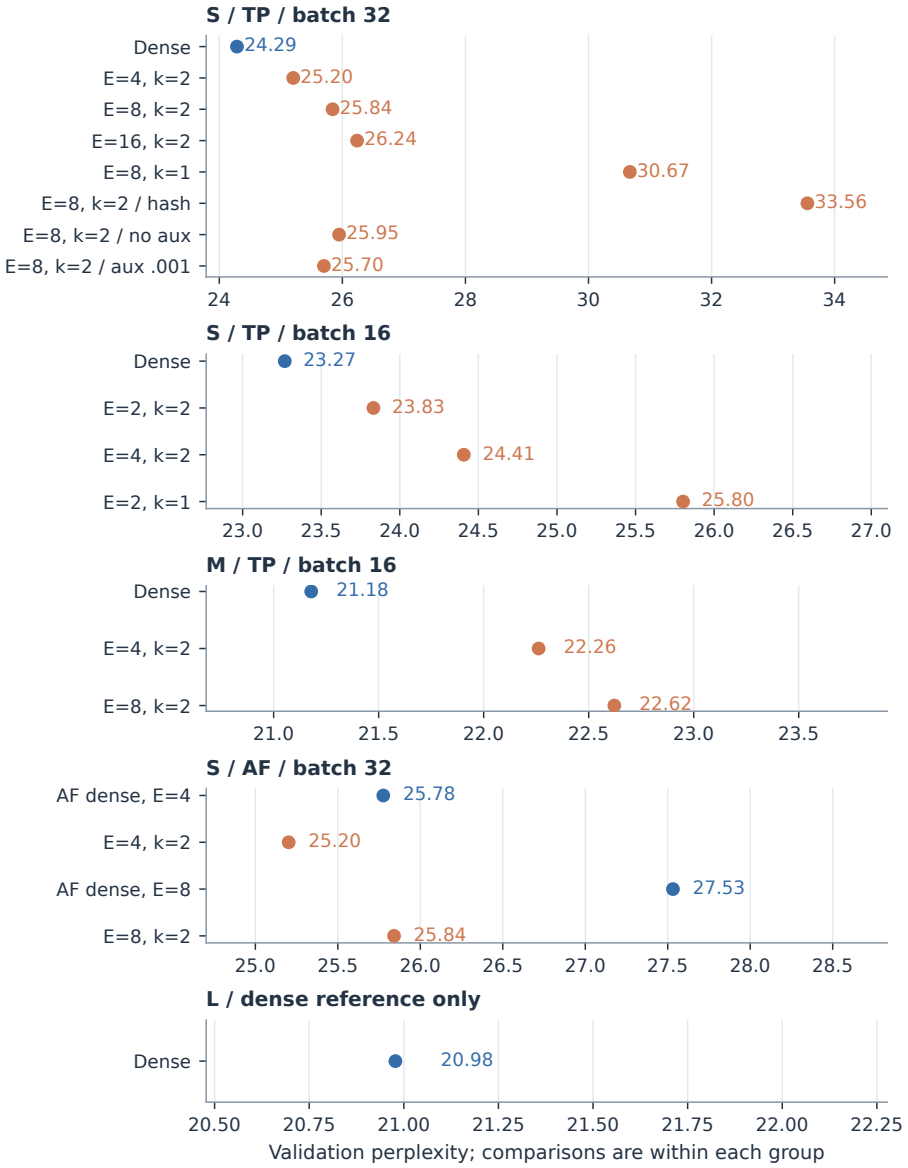


Figure 11: Quality records organized by comparison protocol. Each panel has its own reference group and horizontal range; this is an inventory, not a ranking across scales or budgets. AF panels repeat the paired MoE records intentionally. The L panel contains a dense reference only.

Table 9: Run inventory from JSON records. Standard comparison groups use three epochs. Diagnostic and budget-check rows have the roles stated in the final column and are not substituted for matched references.

Scale	Configuration	Batch	Epochs	Seed	Params (M)	PPL	Use
L	Dense	16	3	0	531.3	20.98	reference
M	Dense	16	3	0	203.1	21.18	comparison
M	E=4, k=2	16	3	0	203.2	22.26	comparison
M	E=8, k=2	16	3	0	203.3	22.62	comparison
S	AF dense, E=4	16	3	0	76.8	24.48	diagnostic
S	Dense	16	3	0	95.7	23.27	comparison
S	E=2, k=1	16	3	0	95.7	25.80	comparison
S	E=2, k=2	16	3	0	95.7	23.83	comparison
S	E=4, k=2	16	3	0	95.7	24.41	comparison
S	Dense	16	6	0	95.7	21.10	budget check
S	AF dense, E=8	32	3	0	67.4	27.53	comparison
S	AF dense, E=4	32	3	0	76.8	25.78	comparison
S	Dense	32	3	0	95.7	24.29	comparison
S	E=4, k=2	32	3	0	95.7	25.20	comparison
S	E=4, k=2	32	3	1	95.7	25.27	comparison
S	E=8, k=1	32	3	0	95.8	30.67	comparison
S	E=8, k=2 / hash	32	3	0	95.8	33.56	comparison
S	E=8, k=2 / aux .001	32	3	0	95.8	25.70	comparison
S	E=8, k=2	32	3	0	95.8	25.84	comparison
S	E=8, k=2 / no aux	32	3	0	95.8	25.95	comparison
S	E=8, k=1	32	3	1	95.8	30.89	comparison
S	E=8, k=2 / hash	32	3	1	95.8	33.78	comparison
S	E=8, k=2	32	3	1	95.8	25.84	comparison
S	E=16, k=2	32	3	0	95.9	26.24	comparison
S	E=16, k=2	32	3	1	95.9	26.21	comparison