A Recipe for an Elastic Mixture: One Mixture-of-Experts for Every Resource Budget

Chloe Chia [1]

Abstract

We present a single Mixture-of-Experts checkpoint that exposes a continuous memory–quality knob at serving time, letting researchers trade resident expert memory for quality without retraining or maintaining a family of distilled models. Starting from Qwen3-30B-A3B, we continue pretraining for a small fraction of the original compute budget under a fixed Bernoulli expert mask. The resulting model matches a no-mask control trained for the same compute at full inference, while degrading far more gracefully at mask rates it was never trained on — substantially lower loss across a wide range of held-out masking ratios, where the base model collapses. This means a single deployed checkpoint can be served at a range of expert-memory footprints, shrinking HBM residency on memory-pressured nodes, fitting larger batches or longer contexts in the freed capacity, and allowing the same weights to be co-located with other workloads on a shared accelerator. Looking at the router itself, the mechanism is not redundancy. Per-token routing actually becomes more peaky under masking; what changes is that different tokens prefer different fallbacks, so the batch-level load flattens, yielding a meaningful reduction in max-expert load and a corresponding theoretical throughput gain on top of the memory savings. Code [url].

1. Introduction

1.1. Motivation: Pretrained MoEs Are Inelastic

Foundation models are increasingly deployed across a spectrum of resource budgets (Kwon et al., 2023; Leviathan et al., 2023). The same Qwen3-30B-A3B model (Team, 2025) that runs on an 8-GPU server in a datacenter is desirable on a single GPU at the edge, in a high-throughput batched serving stack, and in latency-sensitive interactive applications. These deployments differ by orders of magnitude in available memory, compute, and latency targets. The standard answer in dense models is to ship multiple checkpoints at multiple sizes; for MoE this is doubly impractical because each training run requires substantial compute.

MoE models (Shazeer et al., 2017; Fedus et al., 2022; Jiang et al., 2024; Dai et al., 2024), despite using conditional computation per token, have never been framed as elastic at deployment time: there is no notion of a single trained MoE that supports varying memory budgets, compute budgets, or latency budgets at inference. The natural deployment-time control surface for an MoE is the number of experts loaded and made available to the router. We show empirically that pretrained MoEs collapse catastrophically under this control surface: dropping half the experts at inference inflates Qwen3-30B-A3B's validation loss from 1.45 to 2.75 nats, an effective perplexity inflation of more than $3.7\times$ (Table 1, top row). Continued pretraining with random expert masking eliminates this collapse at near-zero cost to full-inference quality (Table 1, bottom row).

Table 1. Validation loss across inference-time mask rates. Control (continued pretraining, no masking) gives a small uniform improvement over the released Baseline; Elastic (continued pretraining with random expert masking) Pareto-dominates both at every $\rho > 0$.

Model	ρ=0	ρ=0.5	ρ=0.7
Qwen3-30B Baseline	1.45	2.75	3.97
Control	1.29	2.51	3.75
Elastic	1.31	**1.78**	**2.28**
Elastic vs. Control	$+1.6\%$	-29.1%	-39.2%

1.2. Contributions

The core method is intentionally small: at every training step we sample a Bernoulli mask and zero out a fraction of the router's pre-softmax logits. Nothing else about the standard continued-pretraining setup (Parmar et al., 2024) changes. With less than 5×10^{19} FLOPs of extra compute,

Workshop on Adaptive Foundation Models (AdaptFM), ICML 2026, Seoul, South Korea. [1]UC Berkeley, Berkeley, CA, USA. Correspondence to: Chloe Chia <chloechia@berkeley.edu>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

this turns a pretrained MoE into a checkpoint that supports any mask rate at inference.

The interesting result is what happens at mask rates the model never saw. We trained at a fixed ρ=0.3, but the gap over a no-mask control widens as ρ climbs (39% at ρ=0.7, 43% at ρ=0.8). Whatever the router has learned, it is not specific to the training rate.

When we look at the router itself, the obvious hypothesis, "masking forces uniform spread", is wrong. Per-token routing is *more* concentrated under masking, not less; the diversity is across tokens. Different tokens have different fallback experts, so the batch-level load is balanced even though each token still commits to a small set. We call this *structured fallback routing*, and it is what gives Elastic both the loss win and the throughput win at once.

2. Background

2.1. MoE Compute Profile and Its Deployment Bottlenecks

MoE language models (Shazeer et al., 2017; Fedus et al., 2022) decouple parameter count from per-token compute by activating only top-k experts per token (Lepikhin et al., 2021; Zoph et al., 2022). For Qwen3-30B-A3B (Team, 2025), top-8 routing across 128 experts means ∼3.3B active parameters per token out of ∼30.5B total—roughly 9× less compute than an equivalent-capacity dense model. Under the standard $6ND$ FLOP estimate (Kaplan et al., 2020; Hoffmann et al., 2022), training and inference compute scale with active parameters, motivating production MoE deployments such as Mixtral (Jiang et al., 2024) and DeepSeek-V3 (DeepSeek-AI, 2024).

These savings come at three systems costs. **(1) Memory.** All experts must reside in HBM since the router can dispatch any token to any expert—∼61 GB for Qwen3-30B-A3B in bfloat16. MoEs reduce per-token compute but not per-deployment memory. **(2) Communication.** In expert-parallel deployment, tokens are routed via all-to-all collectives, often the dominant per-layer cost. **(3) Load imbalance.** Top-k routing concentrates tokens on a few "preferred" experts, and throughput is gated by the most-loaded one.

Why coarse-grained masking preserves GPU efficiency. A recurring concern with sparsity-style methods is that reducing non-zero weights does not always translate to wall-clock speedup (Hooker, 2021), because GPU kernels achieve peak FLOPs only on dense, regularly shaped tensors. The advantage of standard MoE is that all experts are *homogeneous*: an expert is a single fixed-shape MLP, identical across the layer, so a batched expert forward is a single high-utilization matmul. Methods that change expert dimensions individually (e.g. FH/FW/FHW-style fine-grained pruning) sacrifice this homogeneity and resemble unstructured pruning, which is hard to accelerate on contemporary GPUs. Our intervention is deliberately coarse: we mask whole experts, not weights within experts. Each expert is either fully available (and runs as the original dense matmul) or fully unselectable (and does no work). The hardware-friendliness of standard MoE is therefore preserved unchanged. The only systems-level effect is that the remaining expert set is smaller, which is exactly the opportunity: fewer experts means smaller HBM footprint and reduced all-to-all bandwidth without any new kernel work.

The deployed MoE is therefore over-provisioned for any single inference workload. A serving stack that could selectively unload experts at inference time could reclaim memory, reduce all-to-all bandwidth, and run on smaller pools—without changing the trained weights and without sacrificing kernel efficiency. We empirically show that pretrained MoEs cannot do this without retraining. Continued pretraining with random expert masking closes this gap.

2.2. The Qwen3-30B-A3B Architecture

Qwen3-30B-A3B is a 48-layer transformer decoder with 128 experts per MoE block and top-8 routing per token. Of ∼30.5B total parameters, ∼1.6B are dense (attention, layer norms, embeddings) and ∼28.9B are partitioned across 6,144 expert MLPs. Active parameters per token are ∼3.3B.

Our masking is applied on top of the 8-of-128 sparsity pattern: at training mask rate ρ, an additional ρ fraction of all 128 experts are rendered unselectable for the duration of each forward pass. At ρ=0.3, ∼38 experts are unavailable, leaving ∼90 candidates for top-8 selection. The masking is non-trivial: the model cannot route to its preferred experts and ignore the mask.

2.3. Related Work

Elastic inference for dense models. MatFormer (Devvrit et al., 2024) and Slimmable Networks (Yu et al., 2019) expose width as a control surface; Stochastic Depth (Huang et al., 2016) admits layer-wise dropout. Activation-sparsity methods (Liu et al., 2023; Song et al., 2024) predict which neurons or heads can be zeroed per input. None target pretrained MoEs.

MoE deployment-time adaptation. The closest prior work is inference-time expert pruning (Lu et al., 2024; Koishekenov et al., 2023), which permanently removes experts and so fixes a single operating point. We preserve all experts and adapt routing dynamically, supporting mask-rate selection from one checkpoint.

Routing diagnostics. We adopt the entropy and maxvio metrics from MoE pretraining (Fedus et al., 2022; Dai et al., 2024; Wang et al., 2024) and condition them on inference-

time mask rate. Domain-specialization and co-activation analyses (Muennighoff et al., 2024; Jiang et al., 2024) are complementary diagnostics that our structured-fallback view (Section 3.8) makes predictions about; we leave these for follow-up.

3. Experiments

3.1. Method: Random Expert Masking

Following the conditional-computation formulation of Shazeer et al. (2017) and the top-k routing scheme used in subsequent MoE work (Lepikhin et al., 2021; Fedus et al., 2022; Zoph et al., 2022), let $M_l \in \{0,1\}^E$ be the per-layer expert mask at training step t, where E=128 and l indexes the layer. We sample $M_l \sim \text{Bernoulli}(1-\rho)^E$ at each forward pass; the router's pre-softmax logits for masked experts are set to $-\infty$:

$$\mathbf{w} = \text{softmax}\big(\text{MLP}(x) + \log M_l\big), \quad (1)$$

$$\text{idx} = \text{top}_k(\mathbf{w}, k{=}8). \quad (2)$$

The same mask is applied to all tokens in a forward pass and is independent across layers. This mimics the deployment scenario in which the unavailable expert set is fixed for the duration of a serving period. We resample the mask at every gradient step. Figure 1 illustrates the training and inference configurations.

Under standard MoE training, the router's expected loss is computed assuming all experts are always available. Under random masking, the router minimizes loss across the distribution of expert availability sets, forcing it to learn routing strategies that work when an arbitrary ρ fraction of experts are missing. We use ρ=0.3 as a practical compromise: large enough to require non-trivial routing adaptation, small enough that per-token compute remains close to the pretrained operating point. We use the standard next-token cross-entropy loss with no auxiliary balance term beyond what is in the underlying Qwen3 implementation; masking is applied identically during evaluation.

3.2. Setup

Conditions. *Baseline*: the released Qwen3-30B-A3B checkpoint with no continued pretraining. *Control*: continued pretraining with ρ=0 (no masking). *Elastic* (ours): continued pretraining with fixed ρ=0.3 random expert masking. *B2* (ablation, Section 4): continued pretraining with a linear mask-rate schedule from ρ=0 to ρ=0.5.

Training. The Elastic training time was set by convergence: validation loss flattens well before step 1,100 and further training yields diminishing returns. This is roughly four orders of magnitude smaller than a from-scratch MoE pretraining budget (∼10T tokens, $\sim 2 \times 10^{23}$ FLOPs). Full hyperparameters in Table 7; FLOPs derivation in Section D.

Compute-matched comparison. To rule out the alternative that Elastic's gains are simply due to more training steps, we compare Elastic at step 500 against Control at step 543 (matched compute). Elastic still achieves a 33% reduction in validation loss at ρ=0.5 (1.957 vs. 2.910), and Control's loss at ρ=0.5 is stationary across its full training run (only 0.04 nats over 500 steps, 2.96 → 2.92), with extrapolation to Elastic's full budget still leaving a ∼1.0-nat gap. The elasticity gain is structural, not a function of training duration.

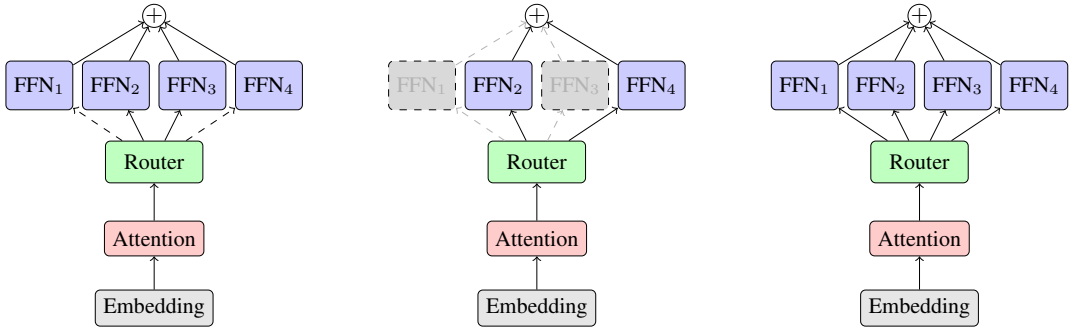
Evaluation. We evaluate on a held-out validation set of mixed web and scientific text drawn from the same distribution as our continued-pretraining corpus (full curation details in Section C). Mask-rate sweeps use 5 random Bernoulli mask samples per rate, averaging over samples to reduce single-sample variance. We report three metrics:

- *Mean negative log-likelihood (NLL)* on next-token prediction. Lower is better; perplexity is $\exp(\text{NLL})$.
- *Maxvio routing imbalance*: max-over-mean expert load minus 1, averaged across all 48 MoE layers. Lower is better; $(\text{maxvio} + 1)$ is the worst-case load multiplier on the most-loaded expert and bounds throughput in expert-parallel serving.
- *Routing entropy*: per-layer entropy of the normalized expert-load distribution (in nats), averaged across layers. Higher is better; bounded above by $\log(E_{\text{avail}})$. A drop indicates the router is collapsing onto a small set of preferred experts under masking.

Per-token tail analysis (Section 3.7) is on 32,768 validation token positions, scored on the same fixed batch under each (model, mask-rate) condition so per-token comparisons are exactly apples-to-apples.

3.3. Main Result: Elastic Trades Memory for Quality on a Single Checkpoint

Stochastic-mask training produces a single MoE checkpoint that runs across multiple memory budgets without retraining and at near-parity quality. The headline operating point is the training mask rate ρ=0.3: Elastic reaches validation loss 1.50, within 0.21 nats (16%) of Control's no-masking inference at ρ=0 (1.29), while loading only ∼70% of expert parameters during inference. Critically, choosing the elastic-trained checkpoint costs nothing when memory is unconstrained: the same Elastic checkpoint evaluated at full memory (ρ=0) reaches loss 1.31, within 2% of Control's 1.29 and inside typical pretraining variance. So the operator who has memory to load all experts loses no quality by deploying Elastic over the non-elastic checkpoint, and the



(a) Standard MoE **(b) Training: drop 30%** **(c) Inference: all experts**

Figure 1. **Random expert masking during continued pretraining.** (a) Standard MoE routes each token to top-k experts. (b) During continued pretraining we randomly mask 30% of experts per forward pass; the router must use a smaller available pool, and the remaining experts must absorb redirected tokens. (c) At inference, all experts are again available; the model has learned routing patterns that generalize across mask rates and produce more uniform expert utilization.

operator who is memory-constrained gains a $\sim$30% memory budget for a 16% quality penalty—and any operating point in between—selectable at inference time, on the same artifact.

Comparing Elastic and Control under matched masking (Section 3.4, Table 2) shows Elastic strictly wins at every $\rho \geq 0.3$, ruling out the alternative that the memory savings come from continued pretraining alone—mask exposure during training is what produces elasticity. The remainder of Section 3 expands this in four views: the full mask-rate sweep including extrapolation to rates unseen during training (Section 3.4), memory-quality tradeoff (Section 3.5), downstream task accuracy (Section 3.6), and per-token catastrophic-failure analysis (Section 3.7).

3.4. Mask-Rate Sweep: Elastic Generalizes to Unseen Rates

The per-rate comparisons in Section 3.3 establish that Elastic beats Control *at* the training mask rate (ρ=0.3). The sweep in Table 2 (visualized in Figure 2a) isolates a stronger claim: the routing strategy transfers to rates *outside* the training distribution. Elastic was trained only at ρ=0.3, so the four masked rates $\rho \in \{0.5, 0.6, 0.7, 0.8\}$ are all above the training rate and represent extrapolation. The Elastic-vs-Control gap not only persists under extrapolation, it *grows* with distance from the training rate: 0.13 nats in distribution at ρ=0.3, 0.73 at ρ=0.5, 1.47 at ρ=0.7, 2.27 at ρ=0.8 (-43% relative). This rules out overfitting to one mask rate and supports the deployment claim of Section 3.3 across the full operating range: a single training rate is sufficient to confer drop-tolerance across at least $\rho \in [0, 0.8]$, giving the operator a continuous memory–quality frontier to move along at inference time on the same artifact.

Table 2. Validation loss across the full inference-time mask-rate sweep, averaged over 5 random Bernoulli mask draws per rate. Qwen3-30B numbers come from our own inference runs on the released checkpoint under the same masking protocol.

ρ	Qwen3-30B	Control	Elastic
0.0	1.45	1.29	1.31
0.3	1.76	1.63	**1.50**
0.5	2.75	2.51	**1.78**
0.6	2.66	2.51	**1.96**
0.7	3.97	3.75	**2.28**
0.8	5.37	5.27	**3.00**

3.5. Memory–Quality Tradeoff

This section establishes the central elastic-inference claim: a single trained model whose deployment cost can be tuned across memory budgets without retraining. We focus on a few key metrics that together show quality does not degrade as experts are masked—validation loss, the max-load violation maxvio (which caps expert-parallel throughput as $1/(\text{maxvio} + 1)$), router entropy, and active-expert memory (the budget freed when masked experts are physically unloaded). Table 3 reports these jointly across mask rates. Elastic and Control are indistinguishable in routing imbalance near the unmasked operating point, but the gap widens monotonically with ρ: training-time mask exposure produces a routing structure that limits the load skew induced by capacity contraction, so memory savings translate to throughput gains rather than throughput penalties. The two highlighted rows mark the canonical operating points of the elastic-inference story—full-budget parity with the released checkpoint at one end, a halved-memory regime where the released model collapses but Elastic remains usable at the other.

Figure 2. Mask-rate sweep (mean of 5 random masks per rate). **(a) Loss:** Elastic, trained only at ρ=0.3, holds dense quality and degrades far slower than Control at higher ρ; at ρ=0.8 it cuts loss from 5.37 to 3.00. **(b) Max-load:** Elastic stays below Control, implying higher expert-parallel throughput. **(c) Entropy:** Elastic resists router collapse under masking.

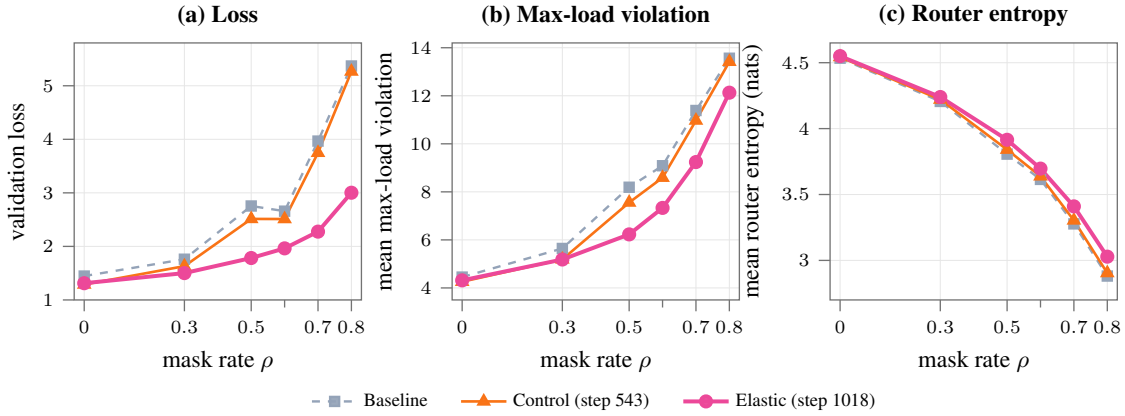


Table 3. Elastic Pareto-dominates baseline across memory budgets at fixed FLOPs/token F_0 (unchanged by masking). Red row: baseline collapses at half-memory; green row: Elastic matches baseline at full memory. Memory assumes masked experts are unloaded; K/E_{avail} is effective routing density.

Configuration	Memory (GB)	FLOPs/token	Maxvio	Loss	K/E_{avail}
Baseline @ ρ=0	61.0	F_0	4.26	1.45	6.25%
Baseline @ ρ=0.5	32.1	F_0	8.19	**2.75**	12.5%
Elastic @ ρ=0	61.0	F_0	4.32	1.31	6.25%
Elastic @ ρ=0.3	43.7	F_0	5.18	1.50	8.9%
Elastic @ ρ=0.5	32.1	F_0	6.50	**1.78**	12.5%
Elastic @ ρ=0.7	20.5	F_0	9.39	**2.28**	21.9%
Elastic @ ρ=0.8	14.8	F_0	11.81	**3.00**	31.3%

3.6. Minimal Downstream Validation

Table 4. HellaSwag (Zellers et al., 2019): Elastic at ρ=0.3 matches the released baseline at full inference within 0.1 pts—retained downstream accuracy at ∼30% less expert memory. Δ_ρ is the masking-induced drop (ρ=0→0.3); gap is Elastic − baseline.

Model	ρ=0	ρ=0.3	Δ_ρ
Qwen3-30B (released)	59.40%	54.50%	−4.90
Elastic (ours)	**62.20%**	**59.30%**	**−2.90**
gap	+2.80	+4.80	—

The +2.8 pt at full inference reflects continued-pretraining gains on our mixed-domain corpus, not the elastic property per se; the masking-induced robustness shows up in the smaller Δ_ρ (−2.90 vs. −4.90).

3.7. Catastrophic Tokens under Masking

Mean validation loss aggregates over many easy tokens and a few catastrophic ones; for elastic deployment we care more about the catastrophic ones, since they dominate user-visible generation errors when their preferred experts are masked. We compute per-token NLL at ρ=0.5 and ρ=0.0 for both Control and Elastic on a fixed batch of 32,768 positions, and define $\Delta_{\text{loss}}(t) = \text{NLL}(t \mid \rho{=}0.5) - \text{NLL}(t \mid \rho{=}0.0)$; large Δ_{loss} marks the expert-dependent tokens Elastic should rescue.

The decoded examples reveal a consistent pattern: highly specific scientific and technical names that the base model knows confidently at full inference (NLL near 0) but loses access to when its preferred experts are masked. Examples include "Leydig" cells in endocrinology, "ROCK" inhibitor in stem-cell biology, "Kaluza–Klein truncation" in theoretical physics, "Hitchin Systems" in mathematical physics, and "glucuronic" in biochemistry. For one such token (a unit "μ" after "(25"), Elastic reduces the masking-induced NLL inflation from 11.66 nats to 2.20 nats, a ∼13,000× reduction in per-token perplexity. Full top-15 list with contexts in Section A.

3.8. What the Routing Reveals: Structured Fallback

The intuitive guess is that random masking forces the router to spread weight evenly. *We find the opposite.* Under mask-

ing, Elastic's per-token routing becomes *more peaky* than Control's: top-k entropy falls more and max-weight share rises more. When preferred experts are masked the router does not hedge—it commits confidently to a learned alternative.

We call this *structured fallback routing*: each token has a preferred expert and an ordered list of fallbacks, and the router skips down that list rather than redistributing uniformly. This peaky-but-diverse pattern is consistent with the per-domain expert specialization reported for OLMoE and Mixtral (Muennighoff et al., 2024; Jiang et al., 2024) and with the fine-grained-specialization analysis of DeepSeekMoE (Dai et al., 2024), and is naturally encouraged by training under stochastic expert availability (Huang et al., 2016). Different tokens have different fallback orders, which reconciles the apparent tension with batch-level maxvio (Figure 2b): per-token routing concentrates on alternatives, but the alternatives differ across tokens, so aggregate load stays balanced. This contrasts with expert pruning, which permanently removes experts and gives the router no signal to adapt (Table 1, top row).

4. Ablations

4.1. B2 Schedule Underperforms Fixed Rate

Motivated by curriculum-learning evidence that easier-to-harder schedules can improve generalization (Bengio et al., 2009), we ablated the masking schedule with a B2 variant trained under a linear ramp on the mask rate, $\rho(t) = 0.5 \cdot t/T$, where T=3340 is the planned full training horizon (chosen to match a 7B-token continued-pretraining budget). At our reported checkpoint of step 300, this corresponds to an effective training mask rate of $\rho \approx 0.045$. Despite using identical compute and data, B2 underperforms the fixed-rate Elastic objective on every metric we measured (Table 5): the early low-ρ phase lets the model overspecialize on full routing, after which it cannot recover the routing flexibility that stationary high-rate masking would have produced.

Table 5. B2 vs. fixed-rate masking at matched step 300. B2 loses on validation loss (val@ρ) and routing imbalance (maxvio@ρ) at every ρ.

Schedule	val@0	val@0.5	maxvio@0.5
Elastic (fixed 0.30)	1.32	**2.00**	**7.27**
B2 (lin. $0 \to 0.5$)	1.29	3.05	8.84
Δ (B2 $-$ Elastic)	-0.03	$+1.05$	$+1.57$

5. Discussion

5.1. Limitations

Wall-clock measurement. Our throughput claim is analytical: maxvio proxies batched-serving throughput under expert-parallel sharding, and active memory is computed assuming masked experts can be unloaded. We do not measure end-to-end TTFT, ITL, or tokens/sec/GPU on production serving stacks such as vLLM (Kwon et al., 2023) or speculative-decoding pipelines (Leviathan et al., 2023); the maxvio improvement is necessary but not sufficient for wall-clock speedup, since real serving also depends on kernel choice, all-to-all implementation, and batching. Because we mask whole experts (not weights inside them), our intervention preserves expert homogeneity and adds no kernel-efficiency loss relative to standard MoE serving—unlike fine-grained pruning (Hooker, 2021).

Single architecture and single mask rate. We focused on Qwen3-30B-A3B at fixed ρ=0.3. Generalizing to other MoE architectures (Mixtral, DeepSeekMoE, Qwen1.5-MoE) and to uniform-rate sampling during training is left to future work.

Downstream evaluation. Our primary evaluation uses validation loss on a held-out pretraining mix, supplemented by HellaSwag (Section 3.6) for downstream confirmation under both full and constrained inference. We did not evaluate on broader benchmarks such as MMLU, BBH, or HumanEval; per-token tail analysis (Section 3.7) gives finer-grained evidence that Elastic reduces catastrophic-failure tokens, but a comprehensive downstream sweep under masking remains future work.

6. Conclusion

Random expert masking during continued pretraining is enough to retrofit elasticity into a pretrained MoE. The numbers we land on are 33% lower loss at ρ=0.5, 2.5$\times$ fewer catastrophic-failure tokens, and around 27% theoretical serving-throughput gain — but the more interesting takeaway is that the underlying mechanism isn't redundancy. The router learns a fallback structure: peaky per token, diverse across tokens, which is why we get the loss reduction and the load-balance improvement together rather than as a tradeoff.

A natural next step is real inference benchmarking (Kwon et al., 2023) to replace our analytical maxvio-based throughput multiplier with end-to-end measurements (TTFT, ITL, tokens/sec/GPU) under realistic batching and all-to-all kernels.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *International Conference on Machine Learning (ICML)*, 2009.

Dai, D., Deng, C., Zhao, C., et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv:2401.06066*, 2024.

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

Devvrit, Kudugunta, S., Kusupati, A., et al. MatFormer: Nested transformer for elastic inference. In *NeurIPS*, 2024.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformer: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Hoffmann, J., Borgeaud, S., Mensch, A., et al. Training compute-optimal large language models. In *NeurIPS*, 2022.

Hooker, S. The hardware lottery. *Communications of the ACM*, 64(12):58–65, 2021.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *ECCV*, 2016.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., et al. Mixtral of experts. *arXiv:2401.04088*, 2024.

Kaplan, J., McCandlish, S., Henighan, T., et al. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

Koishekenov, Y., Berard, A., and Nikoulina, V. Memory-efficient NLLB-200: Language-specific expert pruning of a massively multilingual machine translation model. *arXiv:2212.09811*, 2023.

Kwon, W., Li, Z., Zhuang, S., et al. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. GShard: Scaling giant models with conditional computation and automatic sharding. In *ICLR*, 2021.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Liu, Z., Wang, J., Dao, T., et al. Deja Vu: Contextual sparsity for efficient LLMs at inference time. In *ICML*, 2023.

Lu, X., Liu, Q., Xu, Y., et al. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv:2402.14800*, 2024.

Muennighoff, N., Soldaini, L., Groeneveld, D., et al. OLMoE: Open mixture-of-experts language models. *arXiv:2409.02060*, 2024.

Parmar, J., Satheesh, S., Patwary, M., Shoeybi, M., and Catanzaro, B. Reuse, don't retrain: A recipe for continued pretraining of language models. *arXiv:2407.07263*, 2024.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Song, Y., Mi, Z., Xie, H., and Chen, H. Powerinfer: Fast large language model serving with a consumer-grade GPU. *arXiv:2312.12456*, 2024.

Team, Q. Qwen3 technical report. *arXiv:2505.09388*, 2025.

Wang, L., Gao, H., Zhao, C., Sun, X., and Dai, D. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv:2408.15664*, 2024.

Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. Slimmable neural networks. In *ICLR*, 2019.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. ST-MoE: Designing stable and transferable sparse expert models. In *arXiv:2202.08906*, 2022.

A. Catastrophic-Tail Tokens (Top 15 under Control)

Table 6 lists the 15 tokens from Section 3.7 with the largest Δ_{loss} under Control. We show the preceding context (truncated to 30 tokens), the next token being predicted, and each model's Δ_{loss}.

Table 6. Top-15 catastrophic tokens under Control at ρ=0.5, with decoded context and Elastic comparison. "Elastic saves" is the Control NLL minus ElasticNLL at ρ=0.5 on this position.

#	context (last ∼30 tokens)	next	Control Δ	ElasticΔ	Elastic saves
1	...Jack polynomial, Macdonald polynomial, compact symmetric space...	MAT	+13.03	+8.03	3.04
2	...glucose, galactose and arabinose from Wampee [Clausena lansium...	Ske	+13.03	+11.22	1.44
3	...glucose, galactose and arabinose from Wampee [C	laus	+12.73	+11.87	0.61
4	...4-NP inhibits the secretion of testosterone by Ley	dig	+12.52	+9.50	3.02
5	...do not compete with hormones at the receptor level. 5.1. Mechanism 4 {#sec5	dot	+12.26	+8.10	4.16
6	...arXiv.org/abs/1009.3805. D. Cassani and P. Koerber, [Tri-S	as	+12.13	+9.93	2.14
7	...arXiv.org/abs/1203.0303. M. M.	Cald	+12.07	+7.48	4.55
8	...EpiSCs, like human ESCs, do not survive well as single cells... selective	ROCK	+11.71	+10.43	1.24
9	...epiblasts cultured with Activin, Fgf2, and XAV939 (25	μ	+11.66	+2.20	**9.45**
10	...arXiv.org/abs/1009.4210. I. B	ena	+11.58	+10.97	0.58
11	...PBDEs ... induction of hepatic enzymes involved in gluc	uron	+11.53	+5.33	**6.00**
12	...Decrease of adherence =}mspace{6	mu	+11.48	+9.65	1.81
13	...Gaiotto, G. W. Moore, A. Neitzke, [Wall-crossing, Hitchin	Systems	+11.45	+9.08	0.01
14	...Kim, Varela, Waldram, [Consistent supersymmetric Kaluza–Klein	trunc	+11.44	+6.69	1.97
15	...Gauntlett and O. Varela, [Consistent Kalu	za	+11.42	+7.00	**4.42**

B. Hardware and Hyperparameters

Table 7 lists the full training and evaluation configuration shared by Control, Elastic, and B2.

C. Validation Data Construction

Our validation set consists of approximately 1,793 sequences of 4,096 tokens each (∼7.3M tokens), held out from the continued-pretraining mixture. The pretraining mixture is a mixed-domain sample of web text, scientific papers (arXiv abstracts and PubMed Central biomedical articles), and code, weighted to match typical pretraining-data distributions used in recent open MoE work. Train and validation splits are deterministic by document hash, with no document overlap. Per-token tail analysis (Section 3.7) uses a fixed batch of 32,768 token positions (16 sequences × 2,048 positions each, drawn from the validation split with seed cfg.seed + 1) so that per-token comparisons across (model, mask-rate) conditions are exactly aligned. Mask-rate sweeps use 20 batches per evaluation, scored at 5 random Bernoulli mask draws per rate.

D. Compute Derivation

Training FLOPs follow the standard $6ND$ approximation, where N is the active parameter count and D is the number of training tokens. For Qwen3-30B-A3B with $N = 3.3 \times 10^9$ active per token and $D = 2.3 \times 10^9$ tokens (1,100 steps at 2.097×10^6 tokens per step), training compute is $6 \times 3.3{\times}10^9 \times 2.3{\times}10^9 = 4.55 \times 10^{19}$ FLOPs.

For a from-scratch MoE at 10T tokens, training compute is $6 \times 3.3{\times}10^9 \times 10^{13} \approx 2 \times 10^{23}$ FLOPs—approximately $4{\times}10^3$ times the compute of our continued-pretraining setup.

Inference memory: at mask rate ρ, active expert memory is $(1 - \rho) \times 28.9$ GB. Throughput multiplier is computed from maxvio: max-load is $(\mathrm{maxvio} + 1)$ times mean load, and serving throughput in expert-parallel deployment is gated by the most-loaded expert; the Elastic/Control throughput ratio is $(\mathrm{maxvio}_{\mathrm{Control}} + 1)/(\mathrm{maxvio}_{\mathrm{Elastic}} + 1)$.

Table 7. Training and evaluation configuration (shared by all continued-pretraining runs unless noted).

Hardware and Parallelism	
GPUs	8 × NVIDIA H100 80 GB
Sharding	FSDP2 (per-parameter)
Precision	bfloat16
Model and Data	
Base model	Qwen3-30B-A3B (released checkpoint)
Sequence length	4096
Micro-batch / GPU	2
Grad accumulation	32
Global batch tokens	$8 \times 2 \times 32 \times 4096 = 2{,}097{,}152$
Optimization	
Optimizer	AdamW
Peak learning rate	3×10^{-5}
Min learning rate	3×10^{-6}
LR schedule	Cosine decay
Warmup steps	100
β	$(0.9, 0.95)$
Weight decay	0.1
Gradient clip	1.0
Training Length and Checkpoints	
Elastic total steps	1,100
Control total steps	543
B2 reported step	300 (for ablation)
Checkpoint cadence	every 100 steps
Masking	
Control mask rate ρ	0 (no masking)
Elastic mask rate ρ	0.30 (fixed)
B2 mask rate ρ	linear $0 \rightarrow 0.5$ over 3340 steps
Mask granularity	per-layer i.i.d. Bernoulli, fixed per fwd pass
Evaluation	
Eval mask rates	$\{0.0, 0.3, 0.5, 0.6, 0.7, 0.8\}$
Mask samples / rate	5 (averaged)
Batches / rate	20
Per-token tail size	32,768 token positions

E. Reproduction Details

Code is released and checkpoints will be released upon acceptance. Training and evaluation scripts are available in the repository. Mask-rate sweeps and per-token tail analysis run on the same validation seed across all conditions, ensuring per-token comparisons are apples-to-apples. We compute per-token loss using standard causal cross-entropy with label shift; routing diagnostics use forward hooks on each `Qwen3MoeTopKRouter` module.