PRUNE GRADUALLY, ADAPT SPARSELY: EFFICIENT PRUNING OF LARGE LANGUAGE MODELS AT EXTREME SPARSITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Weight pruning can reduce the storage and inference cost of large language models (LLMs), but preserving quality at high sparsity levels remains challenging. Existing methods face a quality–cost trade-off in this regime. One-shot pruning is cheap but degrades sharply as sparsity increases, even when followed by post-pruning recovery. A growing line of work instead optimizes the mask directly from the dense model under the end-to-end objective; among these methods, ELSA (Lee et al., 2026), the recent state of the art at extreme sparsity, reaches high quality but at the cost of model-sized optimizer state and auxiliary variables. To address this gap, we propose ETAPS, which prunes gradually and adapts sparsely: it tightens a global sparsity constraint in progressive stages while training a sparse delta with a small, fixed budget. Our key observation is that gradual weight removal keeps each pruning step small enough for sparse adaptation to compensate, allowing the adapted model to provide end-to-end gradients that guide the next pruning decision without model-sized optimizer state. Across six models from 125M to 13B parameters, ETAPS achieves lower perplexity than ELSA at every evaluated sparsity from 50% to 90%, with perplexity reductions of up to 42%. ETAPS is also far cheaper: at 90% sparsity, it prunes LLaMA-2-7B on a single H200, where ELSA uses four, with $5.0\times$ smaller total persistent state and 49% fewer GPU-hours. Together, our findings point to a promising direction for achieving high quality in demanding regimes such as extreme sparsity at substantially lower cost.

1 INTRODUCTION

Large language models (LLMs) are expensive to deploy because they require moving billions of parameters through the memory hierarchy during inference. This is especially true during autoregressive decoding, where each token requires another pass over the weights, making execution memory-bandwidth bound at typical deployment batch sizes (Xia et al., 2023; Macko & Boža, 2026). Weight sparsity offers a direct remedy, as it reduces parameter storage and weight traffic. When paired with a suitable representation and kernel, these reductions can translate into lower inference cost (Xia et al., 2023; Okanovic et al., 2025): with the MACKO SpMV kernel (Macko & Boža, 2026), a 90%-sparse LLaMA-2-7B runs with $2.5\times$ lower latency and $4.6\times$ less memory on an RTX 3090 (Lee et al., 2026). Since the savings grow with the fraction of weights removed, higher sparsity is particularly desirable. The difficulty is that pruning becomes least forgiving exactly where it is most rewarding, as conventional methods degrade sharply at high compression (90%–99%) (Frantar & Alistarh, 2023; Sun et al., 2024; Lee et al., 2026). Our work targets this extreme sparsity regime, where recovering model quality is most challenging and existing methods are pushed to their limits.

One-shot pruning provides the most economical route into sparsity. Methods such as Wanda (Sun et al., 2024), SparseGPT (Frantar & Alistarh, 2023), and Pruner-Zero (Dong et al., 2024) construct sparse models directly from pretrained checkpoints using a small calibration set, with pruning decisions made from a cheap criterion without retraining. Although one-shot pruning remains robust at moderate compression (50%–60%), its performance degrades significantly at higher sparsity levels (Lee et al., 2026). One common explanation is that most one-shot pruning criteria favor locally optimal solutions that are misaligned with the end-to-end objective, causing errors from individual pruning decisions to accumulate as sparsity increases (Lee et al., 2026; Mozaffari et al., 2026).

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.

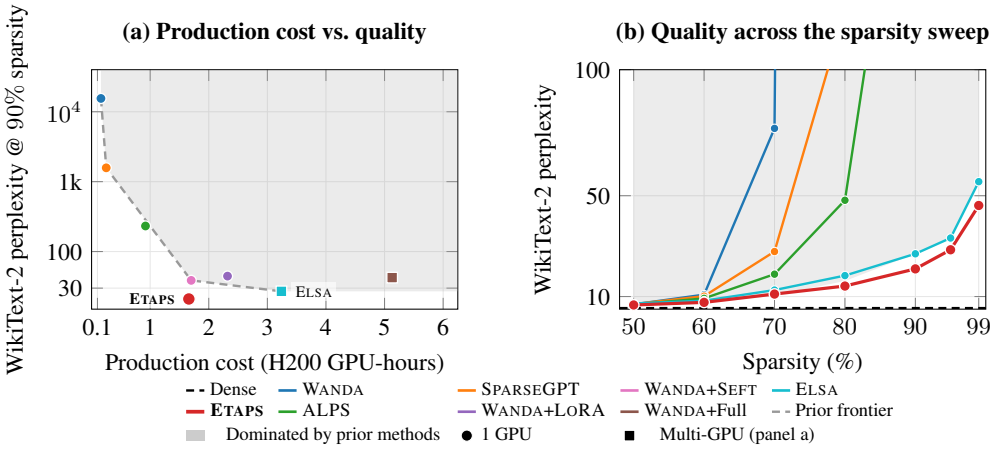

Figure 1: Cost–quality trade-off for sparse LLaMA-2-7B. Both panels report WikiText-2 perplexity.

A natural fix is to recover the model after pruning, allowing the surviving parameters to compensate for pruning-induced errors via end-to-end loss gradients (Hu et al., 2022; Lu et al., 2024; Munoz et al., 2024; Zimmer et al., 2023; Xiao et al., 2025). While most recovery methods update only the remaining weights, SEFT (Xiao et al., 2025) further allows the sparse topology to evolve, steering the model toward sparse configurations better aligned with the fine-tuning objective. However, at extreme sparsity levels (90%–99%), we observe that recovered models degrade sharply as sparsity grows (Table 2). Recovery also remains sensitive to its starting point: even SEFT, despite being able to adjust its topology, reaches noticeably different quality depending on the pruning method used to initialize it. This indicates that although post-pruning recovery can improve a sparse model, its outcome is heavily constrained by the quality of the initial sparse state.

A complementary line of work constructs the mask directly from the dense model, using the language-modeling objective to capture the global effect of weight removal (Lee et al., 2026; Fang et al., 2024; Mozaffari et al., 2026; Hourri et al., 2025). Among these, ELSA (Lee et al., 2026), our primary comparator, achieves strong quality in the extreme-sparsity regime by jointly optimizing the weights and mask under the end-to-end loss using ADMM. However, it requires model-sized auxiliary variables on top of optimizer state, exceeding the persistent memory of dense fine-tuning at matched precision (Table 4). As a result, ELSA illustrates the other end of the quality–cost trade-off, where strong quality at extreme sparsity comes at the expense of substantially higher memory cost.

This leaves an open question: *can extreme-sparsity pruning be guided by gradients from the end-to-end loss throughout its trajectory without paying for model-sized optimizer state?* Existing approaches reveal a trade-off in this regime: inexpensive methods select the sparse support with limited or no gradient information, whereas strong model-level optimization can require dense-model state overhead. ETAPS addresses this gap by bringing together two ingredients for extreme-sparsity LLM pruning: (i) a gradual sparsification trajectory (Zhu & Gupta, 2018), which avoids committing to an extreme mask in a single step, and (ii) a sparse delta (Ansell et al., 2024; Xiao et al., 2025) that carries small trainable and optimizer state, making end-to-end adaptation affordable throughout pruning. Our key observation is that these two ingredients support each other at extreme sparsity. The gradual trajectory makes a small adaptation budget feasible: a model with few trainable parameters struggles to recover from losing most of its weights at once, but recovers better when they are removed a fraction at a time. Adaptation, in turn, informs pruning: gradient statistics gathered at each stage guide the next pruning step, so each decision is built on a model already adapted to the preceding sparsity level. Our ablations show that both advantages grow with sparsity, as progressive pruning outperforms direct projection and gradient-weighted scoring outperforms magnitude alone. Together, they let ETAPS traverse the path to extreme sparsity at a fraction of full-model optimizer cost while achieving strong quality in this regime (Figure 1).

We empirically demonstrate the scalability and efficiency of ETAPS across LLM families and model sizes, from 125M to 70B parameters. Across six models, ETAPS achieves lower perplexity than ELSA at every tested sparsity from 50% to 90%. On LLaMA-2-7B, ETAPS achieves the lowest

perplexity of all evaluated methods across 50%–99% sparsity, and notably, in the extreme regime (90%–99%) it reduces perplexity relative to ELSA by 14.0%–25.0%. Moreover, as the small delta carries only sparse optimizer state, ETAPS also requires far less memory: it prunes LLaMA-2-7B on a single H200 with $5.0\times$ less persistent state and 49% fewer GPU-hours than ELSA on four equivalent GPUs. Finally, ETAPS scales to 67B–70B models and extends naturally to $N{:}M$ sparsity.

Our contributions are threefold:

- **Progressive pruning with increasing adaptation intervals.** ETAPS reaches the target sparsity gradually, using longer adaptation intervals at later pruning stages. Controlled comparisons with direct projection and uniform step allocation show that both the progressive trajectory and the adaptation schedule affect quality at extreme sparsity.
- **Parameter-efficient model-level adaptation.** ETAPS adapts the model using a sparse delta trained on the end-to-end loss. Its persistent trainable parameters and optimizer state scale with a fixed budget, far smaller than the model ($d_\phi \ll d_\theta$). It uses gradient statistics to select the delta's coordinates, requiring only one dense, moment-free accumulator.
- **Quality, efficiency, and scale.** ETAPS demonstrates that extreme-sparsity pruning can achieve competitive quality and a favorable quality–cost trade-off at LLM scale. It remains strong at 90% sparsity on models up to 70B parameters, extends to 95%–99% on LLaMA-2-7B, and requires substantially less persistent memory than ELSA.

2 PROBLEM STATEMENT

Let $\theta \in \mathbb{R}^{d_\theta}$ denote the prunable parameters of a pretrained Transformer, and let $\mathcal{L}_\mathcal{D}(w)$ denote the end-to-end language-modeling objective (next-token cross-entropy) on a dataset $\mathcal{D}$. For a target $S_{\max}$, we define the maximum number of surviving parameters K and the corresponding feasible set $\mathcal{C}_K$ as:

$$K = \lfloor (1 - S_{\max}) d_\theta \rfloor, \qquad \mathcal{C}_K = \left\{ w \in \mathbb{R}^{d_\theta} : \|w\|_0 \le K \right\}. \tag{1}$$

An ideal sparse model satisfies the constrained optimization problem:

$$w_K^\star \in \arg\min_{w \in \mathcal{C}_K} \mathcal{L}_\mathcal{D}(w). \tag{2}$$

Directly optimizing this objective is intractable at LLM scale because support selection is combinatorial and jointly optimizing the surviving weights requires model-scale computation.

Rather than following standard one-shot pruning methods that project the dense model directly into $\mathcal{C}_K$, we decompose the optimization into a sequence of nested constraints:

$$\mathcal{C}_{K_0} \supset \mathcal{C}_{K_1} \supset \cdots \supset \mathcal{C}_{K_P}, \qquad d_\theta = K_0 > K_1 > \cdots > K_P = K. \tag{3}$$

Between consecutive constraints, the model is adapted to the current support; this adaptation compensates for the previous pruning step and supplies gradients from the language-modeling objective that determine the next pruning decision. For $N{:}M$, the same principle is applied to group-wise feasible sets. ETAPS realizes this by freezing the pretrained weights, adapting a sparse delta with a model-dependent budget d_ϕ fixed throughout pruning, and progressively prune using the collected gradient statistics.

3 METHOD: ETAPS

3.1 OVERVIEW AND PROGRESSIVE PRUNING

ETAPS constructs sparse models progressively over P stages. At stage $i \in \{1, \ldots, P\}$, the effective weights are given by

$$w_i = M_i \odot (\theta + \delta_i), \tag{4}$$

where pretrained weights $\theta \in \mathbb{R}^{d_\theta}$ remain frozen during all stages. $M_i \in \{0,1\}^{d_\theta}$ is a binary mask maintaining target cardinality $K_i = \lfloor (1 - S_i) d_\theta \rfloor$, with S_i the sparsity ratio at stage i, and $\delta_i \in \mathbb{R}^d_\theta$ is a sparse, trainable delta.

The masks form a strictly nested sequence ($\mathrm{supp}(M_{i+1}) \subseteq \mathrm{supp}(M_i)$), preventing already pruned parameters from re-entering. The support Φ_i of the sparse delta is restricted strictly to the surviving coordinates ($\mathrm{supp}(\delta_i) \subseteq \Phi_i \subseteq \mathrm{supp}(M_i)$), with at most d_ϕ entries, where d_ϕ is fixed from the dense-model budget. We index the L prunable weight matrices by ℓ and refer to them as modules. Within each pruning step, ETAPS performs restricted adaptation of the sparse delta under the current support, corresponding to the objective:

$$\min_{\delta:\mathrm{supp}(\delta)\subseteq\Phi_i} \mathcal{L}_{\mathcal{D}}\left(M_i \odot (\theta + \delta)\right). \tag{5}$$

At initialization, ETAPS processes C calibration sequences in micro-batches to estimate the gradient statistic used to construct M_1 and initialize the delta support Φ_1. Sparse-delta adaptation at stage i compensates for past pruning perturbations while accumulating the gradient statistic used for the next pruning-mask selection in the same backward passes.

3.2 Pruning Schedules and Adaptation Intervals

To schedule the pruning steps, ETAPS uses the cubic gradual-pruning profile of Zhu & Gupta (2018):

$$S_i = S_{\max} + (S_0 - S_{\max})\left(1 - \frac{i}{P}\right)^3, \qquad i = 1, \ldots, P, \tag{6}$$

with $S_0 = 0$ as the starting point of the schedule and $S_P = S_{\max}$, which defines exactly P pruning boundaries.

Separately from the sparsity profile, we allocate adaptation time non-uniformly: intervals k_i between boundaries grow linearly with the stage index, assigning progressively longer adaptation intervals to later, higher-sparsity stages. This increasing allocation is an ETAPS scheduling choice and is evaluated against a uniform allocation in Section 4.4.2. Over a total optimizer-step budget T, the cumulative step boundaries b_i and interval durations k_i are assigned as:

$$b_i = \mathrm{round}\left[T\frac{i(i+1)}{P(P+1)}\right], \quad b_0 = 0, \qquad k_i = b_i - b_{i-1}, \quad i = 1, \ldots, P. \tag{7}$$

Pruning boundary i is applied at optimizer step b_{i-1} and followed by k_i adaptation steps. With default $P = 32$ and $T = 4096$, the interval grows from $k_1 = 8$ to $k_P = 248$; the final interval is therefore a fixed-mask recovery phase occupying 6.1% of the optimization budget and does not accumulate statistics for another boundary.

3.3 Gradient-Weighted Global Selection

At each stage, ETAPS accumulates a gradient statistic from the same backward passes used for sparse-delta adaptation. Let $\mathcal{B}_{i,r}$ denote the r-th micro-batch processed during interval i, and let $\mathcal{L}_{\mathcal{B}_{i,r}}$ denote its mean token loss. We accumulate

$$\bar{g}_{i,j} = \frac{1}{n_i}\sum_{r=1}^{n_i} \left|\nabla_{w_{i,j}} \mathcal{L}_{\mathcal{B}_{i,r}}(w_{i,r})\right|, \tag{8}$$

where n_i counts micro-batches (equivalently, backward passes), not sequences or optimizer steps. If each optimizer step accumulates gradients over A micro-batches, then $n_i = k_i A$. The element-wise absolute value is taken on each micro-batch gradient before optimizer-gradient accumulation, so Equation 8 is the mean of absolute micro-batch gradients rather than the absolute value of an accumulated gradient. The same backward pass supplies the active-delta update and this saliency statistic, so no additional forward/backward pass is required; ETAPS nevertheless retains the dense backward computation and incurs the element-wise accumulation and dense saliency-buffer cost. For the initial dense state, the C calibration sequences are processed in micro-batches and n_0 is the resulting number of calibration backward passes.

At boundary $i+1$, the accumulated statistic is combined with the current effective-weight magnitude in a gradient-weighted score motivated by gradient-based pruning criteria (LeCun et al., 1989; Lee et al., 2019; Sanh et al., 2020; Zhang et al., 2022a):

$$s_{i,j} = \left|M_{i,j}(\theta_j + \delta_{i,j})\right| \bar{g}_{i,j}, \qquad j \in \mathrm{supp}(M_i). \tag{9}$$

Adaptation therefore changes both the boundary-time effective magnitudes and the interval-aggregated language-modeling gradients used to select the next pruning support. Because the magnitude factor is evaluated at the pruning boundary whereas $\bar{g}_{i,j}$ averages absolute gradients over the preceding interval, Equation 9 is an interval-aggregated saliency score rather than a single-state Taylor estimate. At boundary $i+1$, ETAPS retains the globally highest-scoring K_{i+1} surviving coordinates. We implement this selection with the four-pass streaming histogram and residual-ranking procedure in Algorithm 2, without retaining a second dense score buffer, while enforcing the requested global cardinality.

3.4 SPARSE-DELTA SUPPORT REALLOCATION

While the pruning support is strictly monotone ($\mathrm{supp}(M_{i+1}) \subseteq \mathrm{supp}(M_i)$), the delta support Φ_i is relocated at pruning boundaries. Surviving delta entries retain their state, and vacated slots are reassigned within each module to surviving coordinates with the largest accumulated gradient magnitude $\bar{g}_j$, following the gradient-guided growth principle used in dynamic sparse training (Evci et al., 2020).

Module delta budgets B_ℓ are fixed at initialization. If a module's survivor count falls below B_ℓ, its active delta count contracts rather than reallocating the unused budget to other modules. Thus, the active delta count remains bounded by d_ϕ: the pruning support monotonically removes coordinates, while the delta support redistributes only among surviving coordinates within each module.

Relation to SEFT. SEFT (Xiao et al., 2025) also uses a fixed-budget sparse delta whose support is periodically refreshed with gradient information from its fine-tuning objective. Its role is different: SEFT starts from a model already pruned to the target sparsity, allows delta locations to move onto previously pruned coordinates, and then updates the pruning mask to restore the target sparsity. ETAPS instead starts from the dense checkpoint and couples delta-support reallocation to a monotone pruning trajectory: reallocated delta entries are restricted to currently surviving coordinates, and the adapted effective model directly determines the next pruning support.

4 EXPERIMENTS

We evaluate ETAPS on five model families: OPT-125M/1.3B (Zhang et al., 2022b), Gemma-2-2B/27B (Gemma Team et al., 2024), LLaMA-3.2-3B (Grattafiori et al., 2024), LLaMA-2-7B/13B/70B (Touvron et al., 2023), and DeepSeek-67B (DeepSeek-AI, 2024). We report WikiText-2/C4 perplexity (Merity et al., 2017; Raffel et al., 2020) and zero-shot accuracy over seven downstream tasks using the Language Model Evaluation Harness (Gao et al., 2024). Comparators are MAGNITUDE, WANDA (Sun et al., 2024), SPARSEGPT (Frantar & Alistarh, 2023), ALPS (Meng et al., 2024), and ELSA/ELSA-L (Lee et al., 2026); post-pruning recovery uses full fine-tuning, LORA (Hu et al., 2022), and SEFT (Xiao et al., 2025); and semi-structured $N{:}M$ uses MaskLLM (Fang et al., 2024). Trained methods use exactly 32,768 length-2048 C4 sequence presentations; one-shot methods use 128 calibration sequences. Unless stated otherwise, we set the sparse delta budget d_ϕ to the number of trainable parameters that a rank-32 LORA would add over the same prunable module. Full protocol, baseline provenance, and reproduction details are given in Appendix C.

4.1 MAIN RESULTS

Table 1 summarizes unstructured pruning on LLaMA-2-7B and LLaMA-2-13B at 70%–90% sparsity. Across both models and all three sparsity levels, ETAPS achieves the lowest WikiText-2/C4 perplexity and the highest mean zero-shot accuracy among the evaluated methods. Results at moderate sparsity and on additional model families are reported in Appendix F.

4.2 EXTREME SPARSITY AND POST-PRUNING RECOVERY

Table 2 reports LLaMA-2-7B at 90%, 95%, and 99% sparsity to examine behavior in the extreme-sparsity regime. In addition to ETAPS and ELSA, we evaluate one-shot pruning with WANDA and SPARSEGPT and post-pruning recovery applied after the target sparse mask has already been constructed. For SEFT, we use both WANDA and SPARSEGPT initializations, providing a direct test of

Table 1: LLaMA-2-7B and LLaMA-2-13B at 70%–90% sparsity: perplexity (↓) and average zero-shot accuracy (↑, %). The ELSA 13-B results are taken from (Lee et al., 2026).

		LLaMA-2-7B			LLaMA-2-13B		
Sparsity	**Method**	**Wiki ↓**	**C4 ↓**	**Avg. ↑**	**Wiki ↓**	**C4 ↓**	**Avg. ↑**
0%	DENSE	5.47	7.26	60.21	4.88	6.72	63.37
70%	MAGNITUDE	5.0e4	2.8e4	33.55	218.1	196.4	33.91
	WANDA	77.08	81.51	35.29	45.19	55.31	38.44
	SPARSEGPT	27.77	31.37	41.70	19.65	23.25	45.94
	ALPS	18.88	20.18	45.38	14.13	16.35	49.44
	ELSA	12.42	13.69	46.73	11.14	12.20	50.30
	ETAPS	**10.99**	**10.87**	**49.19**	**9.37**	**9.29**	**56.29**
80%	MAGNITUDE	1.1e5	1.1e5	33.13	3.9e4	4.8e4	32.89
	WANDA	2226	2010	32.25	1127	877.2	32.44
	SPARSEGPT	119.4	100.7	32.54	94.64	78.75	34.65
	ALPS	48.20	47.21	36.85	38.48	36.99	38.55
	ELSA	17.99	17.34	41.77	17.21	16.60	43.27
	ETAPS	**14.17**	**13.09**	**44.50**	**11.93**	**11.10**	**50.20**
90%	MAGNITUDE	4.8e4	5.4e4	32.84	7.4e4	6.4e4	32.12
	WANDA	6477	5099	32.62	2.0e4	1.1e4	32.67
	SPARSEGPT	1369	1033	32.24	1294	911.9	36.05
	ALPS	232.2	174.5	32.20	241.2	149.8	32.92
	ELSA	27.98	23.20	38.41	30.19	25.07	37.37
	ETAPS	**20.98**	**17.39**	**38.84**	**17.50**	**14.89**	**41.94**

how strongly recovery depends on the starting sparse checkpoint. Post-pruning recovery improves

Table 2: Extreme sparsity perplexity on LLaMA-2-7B.

Model	**Method**	**90%**		**95%**		**99%**	
		Wiki	C4	Wiki	C4	Wiki	C4
LLaMA-2-7B *Dense: 5.47 / 7.26*	WANDA	6477	5099	9558	7951	1.24e5	1.44e5
	SPARSEGPT	1369	1033	5075	5628	1.51e4	9158
	WANDA + LORA	43.67	34.33	75.52	48.41	249.6	91.25
	WANDA + Full	31.27	24.30	69.45	39.50	195.75	68.62
	WANDA + SEFT	38.71	30.73	79.49	47.37	160.3	64.85
	SPARSEGPT + SEFT	30.67	26.24	50.81	37.04	92.45	56.46
	ELSA	27.98	23.20	33.22	28.14	55.58	39.98
	ETAPS	**20.98**	**17.39**	**28.56**	**22.12**	**46.13**	**33.22**

one-shot models but remains sensitive to the initial sparse support. ETAPS instead constructs the support during adaptation and achieves lower WikiText-2/C4 perplexity than all evaluated recovery methods and ELSA at 90%–99% sparsity, with 14.0%–25.0% lower perplexity than ELSA across the six dataset–sparsity comparisons. The matched progressive-versus-direct control in Section 4.4.1 separately isolates the effect of the pruning trajectory under the same adaptation budget.

4.3 EXECUTION AT LARGE MODEL SCALE

We evaluate ETAPS on models up to 70B parameters at 90% sparsity. As shown in Table 3, ETAPS achieves 16.85/18.44 WikiText-2/C4 perplexity on Gemma-2-27B, compared with 28.47/29.27 for ELSA-L baseline. We further apply ETAPS to DeepSeek-67B and LLaMA-2-70B, obtaining 13.00/14.65 and 11.69/11.00 perplexity, respectively. These results demonstrate that the same pruning procedure can be applied to models up to 70B parameters without changing its core formulation.

We next examine pruning cost on LLaMA-2-7B at 90% sparsity. Table 4 reports wall-clock time, aggregate GPU-hours, persistent pruning state, and final perplexity on NVIDIA H200 NVL GPUs.

Table 3: Perplexity (↓) at 90% unstructured sparsity across 27B–70B parameter models. ELSA-L utilizes quantized optimizer and auxiliary state and is reproduced on Gemma-2-27B (its maximum reported scale); 67B and 70B settings are compared against scalable one-shot baselines.

	Gemma-2-27B		DeepSeek-67B		LLaMA-2-70B	
Method	Wiki	C4	Wiki	C4	Wiki	C4
DENSE	5.68	9.26	7.11	11.06	3.32	5.71
Magnitude	4.6e12	4.2e12	5.1e5	5.0e5	1.5e5	9.0e4
WANDA	1.5e4	1.1e4	1.3e6	1.2e6	5319.8	4375.0
SPARSEGPT	404.6	337.1	245.6	305.0	268.4	143.4
ELSA-L	28.47	29.27	–	–	–	–
ETAPS	**16.85**	**18.44**	**13.00**	**14.65**	**11.69**	**11.00**

One-shot methods are substantially cheaper but degrade sharply at this sparsity, while post-pruning recovery improves quality at additional compute cost.

Table 4: Resource overhead and perplexity (↓) for 90%-sparse LLaMA-2-7B on NVIDIA H200 NVL GPUs. Persistent state denotes aggregate batch-independent model, optimizer, auxiliary, and saliency memory (excluding activations).

	Perplexity (↓)		Resource Overhead			Memory (GiB)
Method	**Wiki**	**C4**	**GPUs**	**Wall-clock (h)**	**GPU-h**	**Persistent State**
Magnitude	4.8e4	5.4e4	1	<**0.01**	<**0.01**	12.55
WANDA	6477	5099	1	0.01	0.01	12.55
SPARSEGPT	1369	1033	1	0.06	0.06	12.55
ALPS	232.2	174.5	1	0.92	0.92	**4.04**
WANDA + SEFT	38.71	30.73	1	1.70	1.70	14.43
WANDA + LORA	43.67	34.33	1	2.32	2.32	**12.91**
WANDA + Full FT	**31.27**	**24.30**	2	2.56	5.12	103.73
ELSA	27.98	23.20	4	0.81	3.24	132.29
ETAPS	**20.98**	**17.39**	1	1.66	1.66	**26.40**

ETAPS reaches 20.98/17.39 WikiText-2/C4 perplexity in 1.66 GPU-hours on a single H200. In contrast, ELSA reproduction reaches 27.98/23.20 using 3.24 GPU-hours across four H200s, although its parallel execution gives a lower wall-clock time. ETAPS therefore uses 49% fewer aggregate GPU-hours while achieving lower perplexity in this setting. It also requires 26.40 GiB of aggregate persistent pruning state, compared with 132.29 GiB for ELSA, a 5.0× reduction. The reported persistent-state measurements include model, optimizer, auxiliary, and saliency state but exclude activations and other transient buffers.

4.4 ABLATION STUDY

We examine four design choices using LLaMA-2-7B: gradual versus direct pruning, adaptation-interval schedules, saliency score formulations, and sparse-delta capacity. Additional model scale extensions (LLaMA-3.2-3B and OPT-1.3B) and results can be found in Appendix E.

4.4.1 ONE STEP OR GRADUALLY?

At moderate sparsity, direct one-shot pruning with sparse-delta repair performs similarly to progressive ETAPS. However, the gap emerges at high sparsity (Table 5). On LLaMA-2-7B at 90% sparsity, direct projection degrades WikiText-2/C4 perplexity to 34.17/23.14, compared to 20.98/17.39 for ETAPS. We attribute this to the size of the single pruning step: removing 90% of the weights at once leaves a perturbation that the fixed-budget sparse delta cannot fully compensate, whereas progressive interleaving lets the delta adapt at each intermediate mask before the next pruning decision. Increasing from $P{=}1$ (direct) to $P{=}4$ recovers most of the degradation (34.17 to 22.04 WikiText-2), while further increases up to $P{=}32$ yield smaller gains (22.04→20.98; Appendix E.3).

Table 5: Direct versus progressive pruning on LLaMA-2-7B (perplexity ↓).

	50%		70%		90%	
Route	Wiki	C4	Wiki	C4	Wiki	C4
Progressive (ETAPS)	6.66	**8.20**	**10.99**	**10.87**	**20.98**	**17.39**
Direct projection	**6.54**	8.23	11.47	11.34	34.17	23.14

4.4.2 UNIFORM OR INCREASING ADAPTATION INTERVALS?

Under a fixed total optimizer budget, increasing and uniform adaptation intervals perform comparably at 50% sparsity but diverge at higher levels. As shown in Table 6, an increasing interval allocation yields lower perplexity than a uniform one at 90% sparsity.

Table 6: Adaptation-interval allocation on LLaMA-2-7B (perplexity ↓).

	50%		70%		90%	
Allocation	Wiki	C4	Wiki	C4	Wiki	C4
Increasing (ETAPS)	**6.66**	**8.20**	**10.99**	**10.87**	**20.98**	**17.39**
Uniform	6.68	8.24	11.18	11.05	23.19	18.63

4.4.3 GRADIENT-WEIGHTED SALIENCY

Table 7 compares three saliency criteria. While performance remains close at 50% sparsity, alternative scores degrade rapidly at higher sparsities. At 90% sparsity, magnitude-only selection increases WikiText-2/C4 perplexity to 32.64/24.52 (compared to 20.98/17.39 for the default mean absolute gradient), while taking the absolute value after batch-averaging signed gradients degrades performance further to 33.22/25.05.

Table 7: Saliency-score variants on LLaMA-2-7B (perplexity ↓).

	50%		70%		90%	
Score Variant	Wiki	C4	Wiki	C4	Wiki	C4
Mean absolute gradient (ETAPS)	**6.66**	**8.20**	**10.99**	**10.87**	**20.98**	**17.39**
Magnitude only	6.71	8.56	12.06	12.56	32.64	24.52
Absolute mean gradient	9.31	10.68	17.55	15.66	33.22	25.05

4.4.4 DOES THE SPARSE-DELTA BUDGET MATTER?

Setting $d_\phi = 0$ disables sparse-delta adaptation while keeping the progressive mask updates unchanged. At 90%, this control collapses to 463.15/349.50 WikiText-2/C4 perplexity, whereas even a small positive budget (r=8) recovers to 20.70/18.80 on LLaMA-2-7B. Sparse adaptation is therefore essential in the high-sparsity regime.

We use r=32 as a fixed default rather than tuning the budget per model or sparsity level. Here r denotes a *LoRA-equivalent parameter budget*, not a low-rank ETAPS parameterization: d_ϕ equals the number of trainable parameters that rank-r LORA would add to the same prunable modules. Exact model-specific budgets are reported in Table 10, with the complete budget sweeps in Appendix E.7.

5 DISCUSSION

Our work addresses the challenge of preserving quality at extreme sparsity in LLM pruning, where inexpensive one-shot methods degrade sharply while methods that optimize the sparse mask directly from the dense model can require model-sized optimizer state. We propose ETAPS, a progressive pruning framework in which gradients from the end-to-end loss guide pruning while optimizer state

is maintained only for a sparse delta. Across our evaluated sparsity range, ETAPS achieves lower perplexity than ELSA with substantially less persistent memory, and the same core procedure scales to models up to 70B parameters.

Our central finding is that gradual pruning and sparse adaptation are complementary. Without sparse adaptation, progressive pruning collapses at high sparsity. Without gradual pruning, the same sparse delta also falls short: under a matched adaptation budget, direct projection remains competitive at 50%, but its gap to progressive pruning widens as the pruning target increases. The sweep over the number of pruning stages further suggests that this advantage comes mainly from moving beyond single-step pruning to multiple stages, rather than requiring a fine-grained schedule with many stages. Finally, the saliency controls show that gradient weighting becomes increasingly important in the high-sparsity regime.

Limitations. While ETAPS substantially reduces optimizer memory, it does not reduce the cost of backward computation: the backward pass remains dense, and the bf16 saliency accumulator is also dense. Thus, the memory savings come from restricting optimizer state to the sparse delta rather than from reducing backward FLOPs. The state metrics reported here also exclude activations and transient buffers, and therefore should not be interpreted as peak runtime VRAM. For larger models (27B–70B), we use multi-GPU execution primarily to accommodate memory capacity rather than to scale throughput; this demonstrates execution feasibility but does not establish an efficient multi-GPU implementation.

Our experimental setup also involved limited hyperparameter tuning. Rather than independently tuning each model–sparsity configuration, we selected a single model-specific peak learning rate at 90% sparsity and kept the other protocol choices fixed. The pruning support itself is strictly monotonic, and the reported results are based on single experimental runs. Moreover, the pruning schedule, delta budget, and module-wise sparsity allocation also remain largely empirical design choices, leaving room for more principled approaches. Finally, while perplexity and multiple-choice zero-shot accuracy are useful indicators of language modeling capability, they do not fully capture open-ended generation quality, reasoning capabilities, calibration, or safety preservation, nor do we evaluate end-to-end sparse inference latency.

Conclusion. Our findings show that the current quality–cost trade-off curve at extreme sparsity is not a fixed frontier. Strong quality in this regime need not require the optimization-memory costs associated with adapting the full model. By combining gradual pruning with sparse adaptation, ETAPS achieves comparable or better quality than ELSA while using only a fraction of the persistent optimization memory, suggesting that there is room to substantially reduce the cost of optimization at extreme sparsity without sacrificing model quality. Building on this result, future work can explore how the pruning trajectory and sparse adaptation can be designed more effectively, including how pruning stages are scheduled, how adaptation is allocated across stages, and how the delta budget is distributed across modules. We believe these findings provide a strong starting point for further advances in extreme-sparsity pruning and for exploring approaches that improve the balance between model quality and pruning cost in this challenging regime.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, organization, and LaTeX presentation. The authors are responsible for all technical content, experimental results, analyses, citations, and claims, and verified the final manuscript.

ETHICS STATEMENT

This work compresses existing open-weight language models and introduces no new training data or human-subject data. By reducing the compute and memory required for model deployment, sparsification can lower barriers to both beneficial and harmful uses. The compressed models inherit the biases, safety risks, and failure modes of their source checkpoints, and we make no claim that pruning mitigates them. Our evaluation is limited to English perplexity and zero-shot benchmarks; we therefore do not characterize the effects of extreme sparsity on safety, open-ended generation, or performance in other languages.

Reproducibility statement

The algorithmic components are specified in Section 3: the sparsity schedule in equation 6, adaptation-step allocation in equation 7, saliency in equation 9, and the exact global selection procedure in Section 3.3. Implementation details, hardware, training and evaluation data, and hyperparameters are provided in Appendix C. We use publicly available C4 and WikiText-2 data and open-weight base models. Code and the accompanying reproducibility configuration are provided in the supplementary material.

References

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 1778–1796, 2022.

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M. Ponti. Scaling sparse fine-tuning to large language models. *arXiv preprint arXiv:2401.16405*, 2024.

Vladimír Boža. Fast and effective weight update for pruned large language models. *Transactions on Machine Learning Research*, 2024. arXiv:2401.02938.

DeepSeek-AI. DeepSeek LLM: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Dinh-Truong Do, Kiyoaki Shirai, and Le-Minh Nguyen. WIP: Iterative post-training pruning with weighted importance estimation for large language models. In *Natural Language Processing and Information Systems: 30th International Conference on Applications of Natural Language to Information Systems (NLDB 2025), Proceedings, Part I*, pp. 186–200, Cham, 2025. Springer Nature Switzerland.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11346–11374. PMLR, 2024.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952, 2020.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. MaskLLM: Learnable semi-structured sparsity for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Demi Guo, Alexander Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pp. 4884–4896, 2021.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1135–1143, 2015.

Younes Hourri, Mohammad Mozaffari, and Maryam Mehri Dehnavi. PATCH: Learnable tile-level hybrid sparsity for LLMs. *arXiv preprint arXiv:2509.23410*, 2025.

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 *International Conference on Learning Representations*, 2022. arXiv:2106.09685.

Yuxuan Hu, Tian Tian, Xiaodong Chen, Zhe Zhao, Tao Tao, Weifang Zhang, Yuanfeng Li, Yuhang Liang, Cuiping Li, Hong Chen, and Jing Zhang. Efficient low-rank adaptation for sparse large language model. *Tsinghua Science and Technology*, 31(4):2292–2303, 2026. doi: 10.26599/TST.2025.9010174.

Nathan Hubens, Matei Mancas, Bernard Gosselin, Marius Preda, and Titus Zaharia. One-cycle pruning: Pruning ConvNets with tight training budget. In *IEEE International Conference on Image Processing (ICIP)*, pp. 4128–4132, 2022. doi: 10.1109/ICIP46576.2022.9897980.

Mikołaj Janusz, Tomasz Wojnar, Yawei Li, Luca Benini, and Kamil Adamczewski. One shot vs. iterative: Rethinking pruning strategies for model compression. In *ECAI 2025*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2514–2521. IOS Press, 2025. doi: 10.3233/FAIA251100. arXiv:2508.13836.

Siddhant M. Jayakumar, Razvan Pascanu, Jack W. Rae, Simon Osindero, and Erich Elsen. Top-KAST: Top-K always sparse training. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20744–20754, 2020.

Jie Ji, Gen Li, Lu Yin, Minghai Qin, Geng Yuan, Linke Guo, Shiwei Liu, and Xiaolong Ma. Advancing dynamic sparse training by exploring optimization opportunities. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 21606–21619. PMLR, 2024.

Tian Jin, Ahmed Imtiaz Humayun, Utku Evci, Suvinay Subramanian, Amir Yazdanbakhsh, Dan Alistarh, and Gintare Karolina Dziugaite. The journey matters: Average parameter count over pre-training unifies sparse and dense scaling laws. In *International Conference on Learning Representations*, 2025. arXiv:2501.12486.

Eldar Kurtić, Torsten Hoefler, and Dan Alistarh. How to prune your language model: Recovering accuracy on the "Sparsity May Cry" benchmark. In *Conference on Parsimony and Learning*, volume 234 of *Proceedings of Machine Learning Research*, pp. 542–553. PMLR, 2024. arXiv:2312.13547.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, volume 2, pp. 598–605, 1989.

Dongyeop Lee, Kwanhee Lee, Jinseok Chung, and Namhoon Lee. SAFE: Finding sparse and flat minima to improve pruning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 33300–33321. PMLR, 2025. arXiv:2506.06866.

Kwanhee Lee, Hyeondo Jang, Dongyeop Lee, Dan Alistarh, and Namhoon Lee. The unseen frontier: Pushing the limits of LLM sparsity with surrogate-free ADMM. In *International Conference on Learning Representations*, 2026. arXiv:2510.01650.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9908–9922, 2021.

Xudong Lu, Aojun Zhou, Yuhui Xu, Renrui Zhang, Peng Gao, and Hongsheng Li. SPP: Sparsity-preserved parameter-efficient fine-tuning for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33254–33269. PMLR, 2024.

Vladimír Macko and Vladimír Boža. DELTA4: Sparse matrix-vector multiplication for low sparsity. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. Spotlight; earlier version titled MACKO: Sparse Matrix-Vector Multiplication for Low Sparsity, arXiv:2511.13061.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. ALPS: Improved optimization for highly sparse one-shot pruning for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2406.07831.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. arXiv:1609.07843.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):2383, 2018.

Mohammad Mozaffari, Younes Hourri, Mohammad Rastegari, and Mahyar Najibi. LEAP: Learnable end-to-end adaptive pruning of large language models. In *ICML 2026 Workshop on Resource-Adaptive Foundation Model Inference (AdaptFM)*, 2026. arXiv:2605.17289.

Juan Pablo Munoz, Jinjie Yuan, and Nilesh Jain. SQFT: Low-cost model adaptation in low-precision sparse foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12817–12832, 2024.

Patrik Okanovic, Sameer Deshmukh, Grzegorz Kwasniewski, Yi Zhu, Haruto Fujii, Sakina Fatima, Maciej Besta, Kentaro Katayama, Takumi Honda, Yusuke Nagasaka, and Torsten Hoefler. BLaST: High performance inference and pretraining using block sparse transformers. *arXiv preprint arXiv:2507.03117*, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Christophe Roux, Max Zimmer, Alexandre d'Aspremont, and Sebastian Pokutta. Don't be greedy, just relax! pruning LLMs via Frank-Wolfe. *arXiv preprint arXiv:2510.13713*, 2025.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20378–20389, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024.

Aaquib Syed, Phillip Huang Guo, and Vijaykaarti Sundarapandiyan. Prune and tune: Improving efficient pruning techniques for massive language models. In *International Conference on Learning Representations, Tiny Papers Track*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Nasib Ullah, Erik Schultheis, Mike Lasby, Yani Ioannou, and Rohit Babbar. Navigating extremes: dynamic sparsity in large output spaces. *Advances in Neural Information Processing Systems*, 37: 117210–117236, 2024.

Nasib Ullah, Jinbin Zhang, Jean Lucien Randrianantenaina, Erik Schultheis, and Rohit Babbar. HASTE: Hardware-aware dynamic sparse training for large output spaces. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=PAOhstVuIt.

Haojun Xia, Zhen Zheng, Yuchao Li, Donglin Zhuang, Zhongzhu Zhou, Xiafei Qiu, Yong Li, Wei Lin, and Shuaiwen Leon Song. Flash-LLM: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. *Proceedings of the VLDB Endowment*, 17 (2):211–224, 2023.

Qiao Xiao, Alan Ansell, Boqian Wu, Lu Yin, Mykola Pechenizkiy, Shiwei Liu, and Decebal Constantin Mocanu. Leave it to the specialist: Repair sparse LLMs with sparse fine-tuning via sparsity evolution. *arXiv preprint arXiv:2505.24037*, 2025. Accepted to the CPAL 2026 Recent Spotlight Track (non-archival).

Qiao Xiao, Boqian Wu, Patrik Okanovic, Tomasz Sternal, Maurice van Keulen, Elena Mocanu, Mykola Pechenizkiy, Decebal Constantin Mocanu, and Torsten Hoefler. Memory-efficient LLM training with dynamic sparsity: From stability to practical scaling. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2606.00888.

Lang Xiong, Ning Liu, Ao Ren, Yuheng Bai, Haining Fang, Binyan Zhang, Zhe Jiang, Yujuan Tan, and Duo Liu. D^2Prune: Sparsifying large language models via dual Taylor expansion and attention distribution awareness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27171–27179, 2026. doi: 10.1609/aaai.v40i32.39932. arXiv:2601.09176.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Kumar Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57101–57115. PMLR, 2024.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. PLATON: Pruning large transformer models with upper confidence bound of weight importance. In *International Conference on Machine Learning*, pp. 26809–26823, 2022a.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022b.

Michael H. Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *International Conference on Learning Representations, Workshop Track*, 2018.

Max Zimmer, Megi Andoni, Christoph Spiegel, and Sebastian Pokutta. PERP: Rethinking the prune-retrain paradigm in the era of LLMs. *arXiv preprint arXiv:2312.15230*, 2023.

A Extended Related Work

One-shot pruning with surrogate objectives. One-shot pruning is computationally attractive because it produces a sparse model in a single step without retraining. Most existing methods achieve this by relying on a cheap surrogate objective to determine weight importance, following the approach pioneered by Optimal Brain Damage (LeCun et al., 1989). SparseGPT (Frantar & Alistarh,

2023) approximates layer-wise reconstruction error with a Hessian estimated from calibration activations, pruning weights while compensating updates to the survivors. WANDA (Sun et al., 2024) uses a simpler saliency metric, weight magnitude $|W_{ij}|$ scaled by the norm $\|X_j\|_2$ of the corresponding input activation over the calibration set, without updating surviving weights. Pruner-Zero (Dong et al., 2024) moves beyond manually designed importance scores by using symbolic regression to search for pruning criteria. D^2Prune (Xiong et al., 2026) addresses the mismatch between calibration and test-time activations by extending reconstruction-based pruning to jointly model weight and activation changes through a dual Taylor expansion. However, most one-shot pruners follow a local formulation, preventing comparisons across layers and requiring layer-wise sparsity to be specified separately. While most methods assume uniform ratios, OWL (Yin et al., 2024) instead derives non-uniform ratios from layer-wise weight-outlier distributions. We take a different approach by computing saliency directly from end-to-end loss gradients, allowing weights to compete across layers and sparsity allocation to emerge naturally from a single global ranking. This removes the need for a separate surrogate objective and predefined layer-wise sparsity ratios.

Iterative pruning with retraining. Interleaving pruning with intermediate retraining has long been used to recover accuracy lost during pruning (Han et al., 2015). More recently, Janusz et al. (2025) systematically compare one-shot and iterative pruning on vision models and a small language model, and find one-shot preferable at low pruning ratios and iterative pruning ahead at high ones. Two qualifications of their result bear stating: under a fixed retraining budget they place the crossover near 80%, and the regime they ultimately recommend is a hybrid that takes one large initial cut before switching to fine iterative steps. Their setting is neither LLM-scale nor post-training, so we read the ordering it establishes as motivation rather than as evidence for our own regime. At LLM scale, Syed et al. (2023) similarly show that interleaving SPARSEGPT pruning with short fine-tuning improves high-sparsity performance, while Do et al. (2025) use iterative pruning with row-column weighted importance to prevent channel collapse and outperform one-shot pruning at high sparsity. Iterative pruning can follow different schedules, including progressively increasing sparsity toward the target (Zhu & Gupta, 2018; Hubens et al., 2022; Kurtić et al., 2024) and alternating removal with regrowth to allow previously pruned weights to recover (Liu et al., 2021). A related observation comes from sparse pre-training, where Jin et al. (2025) find that evaluation loss is well modelled by the *average* parameter count maintained over training rather than the final count, so that among schedules reaching the same target, those maintaining more parameters for longer are favoured. The optimum they report is interior rather than monotone: pruning begins at 25% of the training compute and concludes at 75%. Their setting is pre-training rather than post-training pruning, and our schedule removes most weights early in optimizer-step time; we cite it only as evidence that the pruning trajectory, not just the final sparsity, affects final quality. The challenge, however, is applying this paradigm at a billion-parameter scale, where iterative pruning requires multiple rounds of recovery retraining over the model parameters (Syed et al., 2023). ETAPS addresses this problem by allocating sparse-delta and optimizer state for the small initial capacity d_ϕ, set in our experiments to the LORA rank-32-equivalent parameter count. The active count $d_{\phi,i} \leq d_\phi$ can shrink as layer supports contract, while inactive slots remain allocated along with their optimizer state.

Constrained optimization for pruning. A different family formulates pruning as a constrained optimization problem, $\min_w \mathcal{L}(w)$ s.t. $\|w\|_0 \leq K$, where the surviving weights emerge from optimization rather than from a separate scoring criterion. SparseFW (Roux et al., 2025) relaxes the binary mask-selection problem to a convex ℓ_1-constrained formulation and solves it with Frank–Wolfe, accounting for weight interactions while minimising per-layer reconstruction error. ALPS (Meng et al., 2024) solves the layer-wise reconstruction problem under an ℓ_0-constraint using ADMM, jointly optimizing the support and the surviving weights, while Boža (2024) applies ADMM to recover optimal weights on a fixed support. SAFE (Lee et al., 2025) extends the constraint to the whole model, directly optimizing a sparsity-constrained training objective through an augmented Lagrangian. ELSA applies surrogate-free ADMM to the model-level objective under a global sparsity constraint and provides empirical evidence that high-quality solutions can be reached at extreme sparsity levels well beyond those reached by conventional one-shot pruning (Lee et al., 2026). However, the computational cost of this method stems from its global formulation: the loss is optimized through dense backpropagation over all d_θ parameters, while introducing additional ADMM-based auxiliary variables in $\mathbb{R}^{d_\theta}$ to enforce the constraint. ETAPS retains a model-level objective and exact cardinality constraints while reducing the allocated trainable optimization state: sparse-delta and optimizer storage is fixed at $d_\phi \ll d_\theta$ slots, of which only $d_{\phi,i} \leq d_\phi$ are active between pruning

boundaries. Dense effective-weight gradients are accumulated during adaptation for global saliency, and global TopK selection enforces the next cardinality exactly without maintaining model-sized ADMM dual variables throughout optimization.

Mask learning. Mask learning treats the sparse support itself as a learnable object, directly optimizing mask variables rather than pruning weights via an importance criterion. MaskLLM (Fang et al., 2024) targets $N{:}M$ sparsity by parameterizing each group's valid patterns as a categorical distribution and using Gumbel-softmax to obtain a differentiable approximation to discrete pattern selection during training, and PATCH (Hourri et al., 2025) learns instead which tiles of a weight matrix are dense and which are 2:4, so that its learnable state is one choice per tile rather than per weight, at the cost of an effective sparsity capped at 50%. LEAP (Mozaffari et al., 2026) uses a per-weight Bernoulli with Gumbel-sigmoid relaxation, making end-to-end unstructured mask learning tractable at 50%–60% sparsity. Except where the parameterization is coarsened as in PATCH, these methods optimize mask variables separately from the model weights, requiring auxiliary mask state on the order of the parameter count and dense per-step gradients in the per-weight case. We instead use gradient statistics to select the sparse support while adapting a much smaller sparse delta, so the state used for recovery also changes the next pruning decision.

Parameter-efficient sparse adaptation. Parameter-efficient methods reduce the cost of adapting LLMs to downstream tasks by restricting the degrees of freedom available for fine-tuning. LoRA (Hu et al., 2022) achieves efficiency by introducing a separate low-rank update instead of the full weight matrices, inspiring its sparse-native variants (Lu et al., 2024; Munoz et al., 2024; Hu et al., 2026) to further constrain these updates to preserve the sparsity of the base model. However, the LoRA formulation limits the expressiveness of updates to the chosen rank. Sparse fine-tuning methods take a different approach by concentrating weight adaptation in a sparse set of trainable coordinates (Guo et al., 2021; Ansell et al., 2022; 2024), where SEFT (Xiao et al., 2025) is most relevant to our setting. SEFT adapts an already pruned model with two coupled mechanisms: a delta support update that periodically reallocates which coordinates are trainable, including onto previously pruned ones, and a topology adaptation step that prunes the base model back to the target sparsity afterwards. ETAPS reallocates a sparse trainable set in the same spirit, but couples it to progressive pruning, restricting reselection to surviving weights so that the target sparsity is never exceeded and no separate restoration step is needed.

Dynamic sparse training. Dynamic sparse training (DST) studies the training problem where the model is initialized and updated with a sparse parameterization throughout. Its defining mechanism is that sparse connectivity evolves during training through an alternating pruning and regrowth mechanism. SET (Mocanu et al., 2018) established this prune-and-regrow paradigm with random regrowth, while RigL (Evci et al., 2020) uses dense gradient magnitude to guide regrowth. Top-KAST (Jayakumar et al., 2020) maintains a separate exploration set of parameters to identify promising connections for future activations, while BiDST (Ji et al., 2024) formulates prune-and-regrowth as a bi-level optimization problem. More recently, SMET (Xiao et al., 2026) studies the optimization instability caused by newly regrown parameters in sparse pre-training of LLaMA-style models from 60M to 1B, a characteristic challenge in DST, and addresses it with optimizer warm-up together with density-aware learning-rate scaling; we adopt the first of the two. ETAPS draws on this observation when reallocating its trainable parameter budget: newly selected parameters have no optimizer history and therefore often receive excessively large updates. We similarly use a warm-up mechanism to stabilize their introduction during recovery. Beyond LLM finetuning, DST has also been successfully applied in model compression for deep networks with extremely large output layers which arise in scenarios of recommendation systems and tagging (Ullah et al., 2024; 2026).

B Detailed Algorithm

B.1 Overall Procedure

Algorithm 1 summarizes unstructured ETAPS; in $N{:}M$ mode, the global TopK step is replaced by the group-wise projection in Algorithm 3.

Algorithm 1 ETAPS, unstructured mode.

Require: Pretrained weights θ (frozen); schedule parameters $S_0, S_{\max}$; delta budget d_ϕ; boundaries P; total steps T; calibration count C
Ensure: Sparse model state $w = M_P \odot (\theta + \delta_P)$

Initialize $M_0 \equiv \mathbf{1}$, $\Phi_0 = \varnothing$, $\delta_0 \leftarrow \mathbf{0}$
Compute schedule $\{S_i\}_{i=1}^{P}$ via equation 6 with $S_P = S_{\max}$
Allocate intervals $\{k_i\}_{i=1}^{P}$ via equation 7
Set $\widetilde{B} \leftarrow \lfloor d_\phi / L \rfloor$; set $B_\ell \leftarrow \min\{\widetilde{B}, d_\ell\}$ for each module and leave any remainder or capped shortfall unused
Accumulate dense calibration gradient statistic $\bar{g}$ over C sequences
for $i = 1, \ldots, P$ **do**
    Evaluate saliency $s_j \leftarrow |M_{i-1,j}(\theta_j + \delta_{i-1,j})|\, \bar{g}_j$ for $j \in \mathrm{supp}(M_{i-1})$
    Compute target survivor count $K_i \leftarrow \min\left(\lfloor (1 - S_i) d_\theta \rfloor, \|M_{i-1}\|_0\right)$
    $M_i \leftarrow$ the K_i highest-scoring coordinates of $\mathrm{supp}(M_{i-1})$, ranked model-wide by streaming histogram selection (Section 3.3)
    Deactivate delta entries pruned in M_i; carry each survivor's value, (m, v), and local Adam step with its coordinate
    for each module ℓ **do**
        Compute target active delta count $|\Phi_i^{(\ell)}| \leftarrow \min\{B_\ell, \|M_i^{(\ell)}\|_0\}$
        Retain surviving active set $\Phi_{i-1}^{(\ell)}$ and refill up to $|\Phi_i^{(\ell)}|$ using largest $\bar{g}_j$ in module ℓ
        For each new coordinate j: set $\delta_j \leftarrow 0$, $m_j \leftarrow 0$, $v_j \leftarrow \bar{g}_j^2$, $t_j \leftarrow 1$, and apply its fixed local LR ramp
    end for
    Reset gradient accumulator $\bar{g} \leftarrow \mathbf{0}$
    Compute active delta count $d_{\phi,i} \leftarrow \sum_\ell |\Phi_i^{(\ell)}|$
    for $t = 1, \ldots, k_i$ **do**
        Update active delta δ_{Φ_i} with AdamW on $\mathcal{L}_{\mathcal{D}}\left(M_i \odot (\theta + \delta)\right)$
        if $i < P$ **then**
            Accumulate micro-batch gradient contributions into $\bar{g}$
        end if
    end for
end for
return $w = M_P \odot (\theta + \delta_P)$

B.2 MEMORY AND OPTIMIZATION STATE

Only the sparse delta carries optimizer moments, so optimizer state is $\mathcal{O}(d_\phi)$; global selection additionally uses one dense bf16 $\mathcal{O}(d_\theta)$ gradient-statistic accumulator. Base parameters and the accumulator use bf16, while delta values and AdamW moments use fp32. We report pruning-state memory as model, optimizer, delta/index, and saliency storage that does not scale with the activation batch, excluding activations. During gradient accumulation it scales as:

$$\mathcal{O}(d_\theta) + \mathcal{O}(d_\phi), \tag{10}$$

with d_ϕ fixed for a given model throughout pruning and optimizer moments restricted exclusively to the $\mathcal{O}(d_\phi)$ component.

The measured LLaMA-2-7B state values corresponding to this definition are reported in Table 4. They are pruning-state-memory measurements rather than peak runtime-memory measurements.

C EXPERIMENTAL DETAILS

C.1 IMPLEMENTATION AND REPRODUCTION DETAILS

The implementation uses PyTorch (Paszke et al., 2019) and draws primarily on the sparse-adaptation implementation of SEFT (Xiao et al., 2025). All author-run experiments use NVIDIA H200 NVL GPUs. ETAPS uses one GPU through LLaMA-2-13B, two for Gemma-2-27B, and four for

Algorithm 2 Global TopK selection.

Require: Module scores $s^{(\ell)}$ on $\mathrm{supp}(M_{i-1}^{(\ell)})$, recomputed on demand for $\ell = 1, \ldots, L$; budget K; bins $B = 4096$
Ensure: Masks $M^{(\ell)}$ with $\sum_\ell \|M^{(\ell)}\|_0 = K$

$n_+ \leftarrow \sum_{\ell,j} \mathbf{1}[s_j^{(\ell)} > 0]$ ▷ pass 1
if $n_+ \leq K$ **then**
    Keep all positive-score coordinates and fill the remaining $K - n_+$ slots from zero-score coordinates by stable global index order; **return**
end if
$a \leftarrow \log_2 \min_{\ell,j}\{s_j^{(\ell)} : s_j^{(\ell)} > 0\}, \quad b \leftarrow \log_2 \max_{\ell,j} s_j^{(\ell)}$
if $a = b$ **then**
    Retain exactly K positive-score coordinates by stable global index order; **return**
end if
$h \leftarrow \sum_\ell \textsc{Histogram}\Big(\{\log_2 s_j^{(\ell)} : s_j^{(\ell)} > 0\};\ B \text{ bins on } [a, b]\Big)$ ▷ pass 2
$\beta \leftarrow \min\Big\{c : \sum_{c' \geq c} h_{c'} \leq K\Big\}$
$\tau \leftarrow 2^{a+\beta(b-a)/B}, \quad \tau_- \leftarrow 2^{a+(\beta-1)(b-a)/B}$
$M^{(\ell)} \leftarrow \mathbf{1}[s^{(\ell)} \geq \tau]$ for all ℓ ▷ pass 3
$r \leftarrow K - \sum_\ell \|M^{(\ell)}\|_0$
if $r > 0$ **then** ▷ pass 4
    Pool at most r scores with $\tau_- \leq s_j^{(\ell)} < \tau$ from each module
    Admit the r highest pooled scores, breaking ties by stable global index order
else if $r < 0$ **then**
    Remove the $|r|$ lowest-scoring selected coordinates, breaking ties by stable global index order
end if
return $\{M^{(\ell)}\}_{\ell=1}^{L}$

DeepSeek-67B and LLaMA-2-70B. The current multi-GPU implementation is used primarily to fit larger models in memory rather than to improve throughput: some phases execute on a single GPU even when multiple devices are allocated, and phases that use two GPUs leave any additional devices idle. These runs show execution at the reported model sizes but are not a distributed-scaling study.

Unless otherwise noted, Magnitude, WANDA, SPARSEGPT, and ALPS are reproduced across the unstructured model grid, while LORA, full fine-tuning, SEFT, and unstructured ELSA are reproduced through 7B. For LLaMA-2-7B, we reproduce ELSA from 50% through 99% using the authors' official implementation and recommended hyperparameters; our reproduced values are used throughout. At 90%, 95%, and 99%, our runs give 27.98/23.20, 33.22/28.14, and 55.58/39.98 WikiText-2/C4 perplexity, respectively. LLaMA-2-13B ELSA values are taken from Lee et al. (2026), and Gemma-2-27B ELSA-L is reproduced.

For N:M, MaskLLM 2:4 is evaluated from the released checkpoint under the same evaluation protocol.

Reproduced baselines use held-out data only when tuning is required; recommended configurations are otherwise used directly. Hyperparameter-search budgets are not assumed to be identical across methods, and reported WikiText-2/C4 test values are never used for hyperparameter selection.

C.2 TRAINING AND EVALUATION DATA

Calibration and adaptation use C4 (Raffel et al., 2020). Following the ELSA data protocol, ETAPS uses 128 sequences for initial calibration and 32,768 length-2048 C4 sequence presentations for adaptation at every target sparsity (about 67.1M tokens); the optimizer-step and batch-size settings that realize this fixed budget are listed in Table 8. Reproduced ELSA and trained recovery controls use the corresponding C4 adaptation budget, while one-shot baselines use 128 calibration sequences; literature-sourced rows retain their original protocols. The trained and one-shot rows are therefore

lower-cost calibration references rather than data-matched training competitors. Zero-shot accuracy is reported on ARC-Challenge, ARC-Easy, BoolQ, HellaSwag, OpenBookQA, RTE, and Winogrande using one fixed Language Model Evaluation Harness configuration, with the mean taken over these seven tasks. The exact harness version, task metric settings, software dependencies, and model revisions are pinned in the released reproducibility configuration.

C.3 HYPERPARAMETERS

Table 8: ETAPS optimization protocol by target-sparsity regime.

Setting	$S_{\max} \leq 60\%$	$S_{\max} > 60\%$
Initial sparsity S_0	0	0
Pruning boundaries P	8	32
Optimizer steps T	1024†	4096
Effective batch size	32†	8
Sparse-delta budget d_ϕ	LoRA rank-32-equivalent over prunable modules	
Gradient-statistic accumulation	full adaptation interval k_i	
Optimizer	AdamW; β_1=0.9, β_2=0.999, ϵ=10^{-6}; fp32 moments	
Weight decay	0	
Learning-rate schedule	cosine decay	
Run seed	42	

†OPT-125M uses T=2048 and effective batch size 16.

Table 9: Peak learning rates by model.

Peak learning rate	**Models**
1e−3	OPT-125M; LLaMA-2-7B; Gemma-2-27B; DeepSeek-67B; LLaMA-2-70B
5e−4	LLaMA-3.2-3B; LLaMA-2-13B
1e−4	OPT-1.3B; Gemma-2-2B

For models evaluated at multiple sparsity levels, the peak learning rate is selected once at 90% on a held-out C4 validation split, disjoint from adaptation and reported test data, and is then fixed across sparsity levels. The 27B–70B runs use 1e−3. We parameterize the nominal sparse-delta budget with a LoRA-equivalent rank r only to obtain a consistent model-dependent parameter count. For the L prunable matrices, $d_\phi(r) = r\sum_{\ell=1}^{L}(d_{\text{out},\ell} + d_{\text{in},\ell})$, equal to the number of trainable parameters that rank-r LoRA would add to those matrices. ETAPS itself stores d_ϕ individual sparse coordinates rather than low-rank factors. The default is r=32. Module budgets satisfy $\sum_\ell B_\ell \leq d_\phi$, and the active count $d_{\phi,i}$ can be smaller still as surviving supports contract (Section 3.4).

Table 10: Prunable parameter count d_θ and nominal LoRA-equivalent sparse-delta budget $d_\phi(r)$. A fixed r is not a fixed fraction of the model: for the default r=32, d_ϕ/d_θ decreases from 6.25% on OPT-125M to 0.61% on LLaMA-2-70B.

Model	d_θ	d_ϕ			
		r=8	r=16	r=32	r=64
OPT-125M	84,934,656	1,327,104	2,654,208	5,308,416	10,616,832
OPT-1.3B	1,207,959,552	7,077,888	14,155,776	28,311,552	56,623,104
Gemma-2-2B	2,024,275,968	10,383,360	20,766,720	41,533,440	83,066,880
LLaMA-3.2-3B	2,818,572,288	12,156,928	24,313,856	48,627,712	97,255,424
LLaMA-2-7B	6,476,005,376	19,988,480	39,976,960	79,953,920	159,907,840
LLaMA-2-13B	12,687,769,600	31,293,440	62,586,880	125,173,760	250,347,520
Gemma-2-27B	26,046,627,840	57,090,048	114,180,096	228,360,192	456,720,384
DeepSeek-67B	65,745,715,200	107,786,240	215,572,480	431,144,960	862,289,920
LLaMA-2-70B	68,451,041,280	103,546,880	207,093,760	414,187,520	828,375,040

Algorithm 3 Progressive N:M mask selection at boundary i.

function STAGEDNM($\mathbf{s}, M, S_i, N, E$)
Require: Saliency scores $\mathbf{s}$; group size M; scheduled sparsity S_i; final survivors N; current support E
Ensure: Monotone group-wise mask
  $\kappa_i \leftarrow \mathrm{clip}(M(1-S_i), N, M)$
  $k_{\mathrm{lo}} \leftarrow \max(\lfloor \kappa_i \rfloor, N)$, $k_{\mathrm{hi}} \leftarrow \min(\lceil \kappa_i \rceil, M)$
  Reshape $\mathbf{s}$ and E into G contiguous groups; mask coordinates outside E
  Compute t_i, the number of eligible groups assigned k_{lo} survivors to match the scheduled average
  If $k_{\mathrm{lo}} < k_{\mathrm{hi}}$, assign these transitions using the fixed boundary-saliency ordering
  Retain the highest-saliency live coordinates up to the assigned survivor count in each group
  return Updated mask
end function

D SEMI-STRUCTURED N:M SPARSITY

For N:M sparsity, ETAPS replaces model-wide projection with a group-wise operator while keeping the same schedule, saliency, and sparse-delta adaptation. Weight matrices are partitioned into contiguous groups of M parameters with at most N survivors. 2:4 matches the sparsity pattern supported by NVIDIA Sparse Tensor Cores; 4:8 is evaluated only as a less restrictive 50%-density pattern (Mishra et al., 2021).

An N:M pattern fixes target sparsity to

$$S_{\max} = 1 - \frac{N}{M}. \tag{11}$$

The schedule thus governs the sequence of intermediate sparsity levels leading to this endpoint. At stage i, the scheduled average number of survivors per group is

$$\kappa_i = \mathrm{clip}(M(1-S_i),\, N,\, M)\,. \tag{12}$$

Because individual groups must retain an integer number of parameters, we bracket κ_i by

$$k_{\mathrm{lo}} = \max(\lfloor \kappa_i \rfloor, N)\,, \qquad k_{\mathrm{hi}} = \min(\lceil \kappa_i \rceil, M)\,. \tag{13}$$

For non-integer κ_i, eligible groups receive either k_{lo} or k_{hi} survivors to match the scheduled average; previously tightened groups remain fixed. Live coordinates are ranked by the same saliency as unstructured ETAPS, with a fixed deterministic ordering used only to choose which eligible groups transition. The constraint therefore tightens monotonically:

$$M{:}M \;\longrightarrow\; \cdots \;\longrightarrow\; N{:}M. \tag{14}$$

Algorithm 3 details group-wise mask selection.

At boundary P, $S_P = 1 - N/M$ and every group reaches the target N:M pattern. Table 11 reports the LLaMA-2-7B semi-structured evaluation. MaskLLM is evaluated from the released 2:4 checkpoint under the same standard evaluation protocol; no 4:8 MaskLLM checkpoint is included.

Table 11: LLaMA-2-7B semi-structured sparsity: perplexity and seven-task zero-shot accuracy.

		Perplexity (↓)		**Tasks** (↑)							
Pattern	**Method**	**Wiki**	**C4**	**ARC-C**	**ARC-E**	**BoolQ**	**HellaSwag**	**OBQA**	**RTE**	**Winogrande**	**Avg.**
–	DENSE	5.47	7.26	42.75	75.55	79.33	57.13	33.40	63.54	69.77	60.21
2:4	MaskLLM	**7.33**	**9.13**	35.24	69.23	71.47	51.28	28.40	57.76	64.48	**53.98**
2:4	ETAPS	7.50	9.41	33.62	67.59	71.83	50.40	28.20	59.57	64.96	53.74
4:8	ETAPS	7.01	8.89	33.79	67.68	74.43	51.70	28.40	65.34	66.06	55.34

E Additional Ablations and Diagnostics

The main text reports the central mechanistic controls on LLaMA-2-7B. This appendix extends those controls to LLaMA-3.2-3B and OPT-1.3B and reports the complete boundary, schedule, budget, allocation, and mask diagnostics.

E.1 Global versus Uniform Module Allocation

Global selection jointly determines surviving weights and module-wise sparsity allocation. The resulting module density,

$$\rho_{i,\ell} = \frac{\|M_i^{(\ell)}\|_0}{d_\ell}, \tag{15}$$

emerges from global ranking. A matched Uniform per-module control holds the schedule, saliency, delta budget, and delta-support reallocation rule fixed. Table 12 favors global allocation on C4 in all ten comparisons, while WikiText-2 is mixed; we therefore treat allocation as an empirical design choice rather than a universal advantage.

Table 12: Global versus uniform module-wise sparsity allocation (perplexity ↓).

Model	Allocation	50%		60%		70%		80%		90%	
		Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
LLaMA-3.2-3B	Uniform	**10.00**	14.16	**12.55**	16.76	17.51	20.45	24.51	26.03	37.39	35.46
	Global	10.14	**13.97**	12.85	**16.61**	**17.12**	**19.93**	**23.82**	**25.48**	**36.01**	**35.15**
LLaMA-2-7B	Uniform	**6.58**	8.30	**7.53**	9.17	11.33	11.22	15.25	13.49	**20.88**	17.78
	Global	6.66	**8.20**	7.69	**9.04**	**10.99**	**10.87**	**14.17**	**13.09**	20.98	**17.39**

Module-density profiles in Table 20 become more non-uniform as sparsity rises (CV: $0.32 \rightarrow 0.86$ for LLaMA-3.2-3B and 0.46 to 0.98 for LLaMA-2-7B). These profiles are descriptive; the matched global-versus-uniform comparison provides the relevant ablation evidence.

E.2 Direct-Projection Sweep

Table 13 extends the main LLaMA-2-7B route comparison to LLaMA-3.2-3B and OPT-1.3B and additionally tests delta-support reallocation after direct projection on LLaMA-2-7B. Direct variants prune to the target during calibration and use the same sparse-delta optimization budget as the corresponding progressive run.

Table 13: Direct versus progressive pruning across three models (perplexity ↓).

Model	Route	50%		70%		90%	
		Wiki	C4	Wiki	C4	Wiki	C4
LLaMA-2-7B	Progressive (ETAPS)	6.66	**8.20**	**10.99**	**10.87**	**20.98**	**17.39**
	Direct projection	6.54	8.23	11.47	11.34	34.17	23.14
	Direct + delta reallocation	**6.53**	8.23	11.34	11.33	35.81	23.12
LLaMA-3.2-3B	Progressive (ETAPS)	**10.14**	**13.97**	**17.12**	**19.93**	**36.01**	**35.15**
	Direct projection	10.67	14.73	19.84	22.39	60.40	49.97
OPT-1.3B	Progressive (ETAPS)	**14.84**	**16.65**	**20.26**	**20.33**	**54.64**	**42.30**
	Direct projection	15.28	17.00	23.16	22.40	154.07	80.73

At 90%, the progressive route is lower on both datasets for all three models. Delta-support reallocation after direct projection does not close the LLaMA-2-7B gap.

E.3 Number of Pruning Boundaries

The largest gain occurs when moving away from one-shot projection: increasing P from 1 to 4 reduces WikiText-2 perplexity by 35.5%, whereas increasing P from 8 to 32 yields a further 4.7%

reduction. C4 shows the same diminishing-return pattern. Because the total optimizer-step budget T is fixed, this ablation varies boundary count without adding adaptation steps; it motivates the fixed choice of P by sparsity band but does not imply that larger P is universally preferable.

Table 14: Boundary-count and start-sparsity controls on LLaMA-2-7B at 90% (perplexity ↓).

P	1 (one-shot)	4	8	16	32 (default)
Wiki	34.17	22.04	22.02	21.91	**20.98**
C4	23.14	18.12	17.68	17.45	**17.39**
S_0	0%	25%	50%	70%	90%
Wiki	20.98	21.13	**20.61**	22.48	34.17
C4	17.39	17.27	**17.18**	17.45	23.14

The lower block varies the ramp origin S_0 at fixed P and fixed T, shortening the sequence of intermediate sparsity levels without reducing the optimization budget. Starting at 25% or 50% changes perplexity only slightly relative to the dense start, while S_0=70% raises WikiText-2 perplexity from 20.98 to 22.48 and S_0=90% reduces to the direct-projection control (34.17/23.14). Thus, a fully dense start is not required in this experiment, but sufficiently late entry degrades quality. Because every row retains the same T, this ablation does *not* demonstrate that beginning from a pre-sparse checkpoint permits the corresponding fraction of the optimization budget to be skipped.

E.4 Adaptation-Interval Allocation

Table 15 extends the main increasing-versus-uniform interval-allocation control to LLaMA-3.2-3B at the same total optimization budget.

Table 15: Adaptation-interval allocation on LLaMA-3.2-3B (perplexity ↓).

	50%		**70%**		**90%**	
Step allocation	Wiki	C4	Wiki	C4	Wiki	C4
Increasing (ETAPS)	**10.14**	**13.97**	**17.12**	**19.93**	**36.01**	**35.15**
Uniform	10.17	14.04	17.38	20.48	39.66	37.84

The separation is small at 50% and larger at 90%, matching the trend in the LLaMA-2-7B main-text control.

E.5 Schedule Shape

The matched increasing-versus-uniform interval-allocation control is reported in Table 6 and Table 15. Here we instead vary the sparsity trajectory while holding the total step budget T, number of boundaries P, and step-allocation rule fixed. The cubic schedule in equation 6 interpolates between S_0 and $S_{\max}$ with a cubic profile, so this comparison changes only the sparsity attained at each boundary. The cubic profile reaches 87.5% of the total sparsity increase by the midpoint, so its later pruning increments are smaller than those of a linear profile; the geometric profile lies between the two. Table 16 compares all three profiles on LLaMA-2-7B, LLaMA-3.2-3B, and OPT-1.3B.

The cubic profile is best in 17 of the 18 model–dataset–sparsity comparisons; the exception is LLaMA-2-7B WikiText-2 at 50%, where linear is lower by only 0.01 perplexity (6.65 vs. 6.66). At 50%, every geometric/linear result is within 1.5% of the cubic result across all three models, whereas the separation is much larger at 90%. There, linear raises WikiText-2 perplexity by 44.5% on LLaMA-3.2-3B and 89.5% on OPT-1.3B relative to cubic; geometric raises it by 12.2% and 19.4%, respectively. Because T, P, and the step-allocation rule are matched, the comparison changes the sparsity trajectory while holding total adaptation steps and boundary count fixed. It supports the cubic profile for the tested configurations, but does not identify a universal optimal schedule or show that the alternatives receive less total recovery compute.

Table 16: Sparsity schedule at fixed step and boundary budgets (perplexity ↓).

Model	Profile	50%		70%		90%	
		Wiki	C4	Wiki	C4	Wiki	C4
LLaMA-2-7B	Cubic (ETAPS)	6.66	**8.20**	**10.99**	**10.87**	**20.98**	**17.39**
	Geometric	6.72	8.25	11.40	11.22	23.98	18.91
	Linear	**6.65**	8.27	12.14	11.52	29.64	23.15
LLaMA-3.2-3B	Cubic (ETAPS)	**10.14**	**13.97**	**17.12**	**19.93**	**36.01**	**35.15**
	Geometric	10.18	14.14	18.09	21.01	40.41	38.32
	Linear	10.18	14.18	19.14	21.90	52.05	48.72
OPT-1.3B	Cubic (ETAPS)	**14.84**	**16.65**	**20.26**	**20.33**	**54.64**	**42.30**
	Geometric	15.02	16.69	21.58	21.18	65.24	48.37
	Linear	15.01	16.71	22.31	21.85	103.53	66.51

E.6 SALIENCY SCORE

Table 17 extends the LLaMA-2-7B main-text control to LLaMA-3.2-3B and OPT-1.3B. MAGNITUDE ONLY drops the gradient factor, while ABSOLUTE MEAN GRADIENT takes the absolute value only after averaging signed interval gradients; all other settings are held fixed.

Table 17: Cross-model saliency-score extension (perplexity ↓).

Model	Score variant	50%		70%		90%	
		Wiki	C4	Wiki	C4	Wiki	C4
LLaMA-3.2-3B	Mean absolute gradient	**10.14**	**13.97**	**17.12**	**19.93**	**36.01**	**35.15**
	Magnitude only	11.20	15.56	22.87	24.57	58.04	47.77
	Absolute mean gradient	17.20	21.60	32.25	32.30	58.27	48.50
OPT-1.3B	Mean absolute gradient	**14.84**	**16.65**	**20.26**	**20.33**	**54.64**	**42.30**
	Magnitude only	17.62	17.70	24.50	22.97	118.31	69.91
	Absolute mean gradient	20.37	20.64	35.58	30.84	103.91	67.42

The mean-absolute-gradient score is lower than both alternatives in every reported LLaMA-3.2-3B and OPT-1.3B dataset–sparsity cell.

E.7 SPARSE-DELTA BUDGET

Here r indexes the LoRA-equivalent nominal budget defined in Table 10; it does not denote a low-rank ETAPS update. For the two models below, $d_\phi = r \times 1{,}519{,}616$ for LLaMA-3.2-3B and $d_\phi = r \times 2{,}498{,}560$ for LLaMA-2-7B. The r=0 rows disable adaptation while retaining progressive mask updates.

Table 18: LLaMA-3.2-3B sparse-delta budget sweep (perplexity ↓; cells report WikiText-2 / C4).

Budget	50%	70%	90%	95%	99%
r=0 (no adaptation)	14.62 / 20.27	53.41 / 67.00	1783 / 1230	2.1e4 / 1.1e4	1.5e4 / 9710
r=8	10.38 / 14.20	17.73 / 20.93	42.08 / 41.23	62.32 / 56.26	122.2 / 95.04
r=16	10.25 / 14.05	17.24 / 20.35	38.31 / 37.46	55.34 / 48.87	100.6 / 77.41
r=32 (default)	10.14 / 13.97	17.12 / 19.93	36.01 / 35.15	47.63 / 43.01	86.28 / 66.78
r=64	**10.10 / 13.82**	**16.73 / 19.54**	**32.95 / 31.82**	**43.40 / 38.87**	**77.07 / 60.13**

On LLaMA-3.2-3B, larger positive budgets consistently reduce perplexity from 50% through 99%. LLaMA-2-7B is less monotonic through 90%, but larger budgets become increasingly helpful at 95% and 99%. We keep r=32 fixed in the primary experiments rather than tuning budget separately for each model–sparsity pair.

Table 19: LLaMA-2-7B sparse-delta budget sweep (perplexity ↓; cells report WikiText-2 / C4).

Budget	**50%**	**70%**	**90%**	**95%**	**99%**
r=0 (no adaptation)	8.31 / 10.60	22.52 / 25.92	463.2 / 349.5	5438 / 2243	8889 / 9852
r=8	**6.62** / 8.20	**10.19 / 10.73**	20.70 / 18.80	31.29 / 25.46	69.39 / 44.15
r=16	**6.62 / 8.18**	10.46 / 10.74	**20.67** / 18.01	30.88 / 23.56	56.61 / 37.10
r=32 (default)	6.66 / 8.20	10.99 / 10.87	20.98 / 17.39	28.56 / 22.12	46.13 / 33.22
r=64	6.70 / 8.25	12.14 / 11.14	21.75 / **16.98**	**27.71 / 20.81**	**42.29 / 29.74**

E.8 Sparse-Delta Budget Utilization

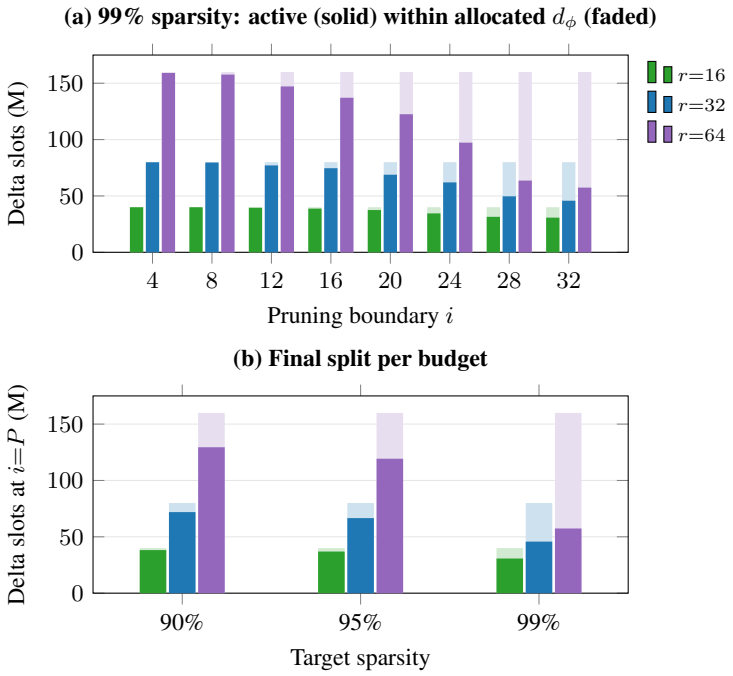

Figure 2: Allocated delta slots (faded) and realized active capacity (solid, $d_{\phi,i}$) on LLaMA-2-7B; the gap corresponds to inactive capacity. Per-layer budgets B_ℓ are fixed at initialization, so slots retire as the surviving pruning support contracts. (a) Evolution across pruning boundaries at 99% sparsity for three budgets. (b) Final active capacity at 90%, 95%, and 99%.

Because per-module budgets B_ℓ are fixed at initialization while module supports contract during pruning, not every allocated delta slot remains usable. When $\|M_i^{(\ell)}\|_0 < B_\ell$, excess slots become inactive rather than being reallocated, so the realized active count $d_{\phi,i}$ can fall below the nominal budget d_ϕ.

Figure 2 quantifies this effect on LLaMA-2-7B. At the final boundary, the inactive share of the default r=32 budget increases from 10.1% at 90% to 16.7% at 95% and 42.8% at 99%. At 99%, increasing the nominal budget to r=64 raises the active count from 45.75M to 57.38M slots while increasing the inactive share to 64.1%. Thus, nominal and realizable delta capacity diverge at extreme sparsity, helping explain the budget sweep in Table 19.

E.9 Module Density Profile

Table 20 summarizes the final module-wise density distributions produced by global selection under the default configuration. Across all six models, module densities become increasingly heterogeneous as sparsity rises from 50% to 90%, while the identities of the most and least preserved mod-

ules remain architecture-dependent. These statistics are descriptive rather than causal; the matched global-versus-uniform comparison in Table 12 provides the controlled evidence for the allocation rule. Mean density is parameter-weighted and fixed by the target sparsity, whereas min, median, max, and CV are unweighted across prunable modules.

Table 20: Final module-density statistics under global selection.

Model	Sparsity	Mean	Min	Median	Max	CV	Most / least preserved module	
OPT-125M	50%	50%	18.5%	50.8%	75.0%	0.23	`L0.fc1`	`L3.fc2`
	70%	30%	8.4%	28.0%	62.2%	0.35	`L0.fc1`	`L3.fc2`
	90%	10%	1.8%	8.9%	34.1%	0.60	`L0.fc1`	`L3.k_proj`
OPT-1.3B	50%	50%	0.9%	52.6%	78.3%	0.40	`L23.k_proj`	`L1.q_proj`
	70%	30%	0.7%	26.4%	55.0%	0.52	`L13.v_proj`	`L1.q_proj`
	90%	10%	0.4%	8.3%	37.9%	0.63	`L0.fc1`	`L1.k_proj`
Gemma-2-2B	50%	50%	12.1%	51.2%	83.2%	0.32	`L7.v_proj`	`L25.q_proj`
	70%	30%	4.0%	29.4%	72.4%	0.50	`L7.v_proj`	`L25.q_proj`
	90%	10%	0.8%	10.2%	61.8%	0.95	`L4.v_proj`	`L10.gate_proj`
LLaMA-3.2-3B	50%	50%	5.8%	50.8%	75.1%	0.32	`L3.v_proj`	`L0.q_proj`
	70%	30%	3.1%	27.1%	66.3%	0.53	`L3.v_proj`	`L25.q_proj`
	90%	10%	1.1%	8.3%	51.0%	0.86	`L14.v_proj`	`L9.gate_proj`
LLaMA-2-7B	50%	50%	0.7%	53.3%	71.5%	0.46	`L2.down_proj`	`L0.q_proj`
	70%	30%	0.3%	27.8%	59.3%	0.66	`L2.down_proj`	`L30.k_proj`
	90%	10%	0.1%	6.5%	42.0%	0.98	`L2.down_proj`	`L30.k_proj`
LLaMA-2-13B	50%	50%	0.6%	51.3%	70.3%	0.49	`L2.down_proj`	`L0.q_proj`
	70%	30%	0.3%	28.5%	60.6%	0.71	`L2.down_proj`	`L33.k_proj`
	90%	10%	0.1%	6.0%	44.6%	1.00	`L2.down_proj`	`L33.k_proj`

F Complete Model Evaluation

F.1 LLaMA-2-7B task-wise results

The task-wise values corresponding to the LLaMA-2-7B columns of Table 1 are reported below.

Table 21: LLaMA-2-7B perplexities and task-wise results at 70%–90% sparsity.

		Perplexity (↓)		Tasks (↑)							
Sparsity	Method	Wiki	C4	ARC-C	ARC-E	BoolQ	HellaSwag	OBQA	RTE	Winogrande	Avg.
0%	Dense	5.47	7.26	42.75	75.55	79.33	57.13	33.40	63.54	69.77	60.21
70%	Magnitude	5.0e4	2.8e4	24.23	28.58	38.10	25.91	15.40	53.43	49.17	33.55
	Wanda	77.08	81.51	18.43	31.02	53.61	28.04	12.60	52.71	50.59	35.29
	SparseGPT	27.77	31.37	21.25	44.61	63.73	32.95	18.20	53.07	58.09	41.70
	ALPS	18.88	20.18	24.74	53.32	64.34	37.85	21.80	53.43	62.19	45.38
	Elsa	12.42	13.69	27.65	56.23	63.91	43.53	23.80	54.15	57.85	46.73
	ETAPS	**10.99**	**10.87**	30.12	61.74	66.79	46.49	25.60	53.07	60.54	49.19
80%	Magnitude	1.1e5	1.1e5	20.14	25.84	46.33	25.64	14.40	51.99	47.59	33.13
	Wanda	2226	2010	21.42	26.73	37.83	25.91	12.40	53.43	48.07	32.25
	SparseGPT	119.4	100.7	17.49	27.57	39.57	27.48	12.60	54.15	48.93	32.54
	ALPS	48.20	47.21	19.45	31.99	61.90	29.20	12.00	51.70	51.70	36.85
	Elsa	17.99	17.34	20.99	47.18	62.11	36.38	18.60	52.71	54.46	41.77
	ETAPS	**14.17**	**13.09**	22.70	54.84	61.99	40.55	23.60	52.71	55.09	44.50
90%	Magnitude	4.8e4	5.4e4	23.55	26.01	42.48	25.65	14.40	46.93	50.83	32.84
	Wanda	6477	5099	21.50	25.76	37.83	25.79	14.80	52.71	49.96	32.62
	SparseGPT	1369	1033	20.99	25.21	37.83	25.71	14.60	52.71	48.62	32.24
	ALPS	232.2	174.5	19.71	27.02	37.86	26.75	13.40	54.51	46.17	32.20
	Elsa	27.98	23.20	19.80	40.53	57.49	31.10	15.80	52.71	51.46	38.41
	ETAPS	**20.98**	**17.39**	19.97	43.98	55.23	32.72	17.00	52.71	50.28	38.84

F.2 LLaMA-2-13B TASK-WISE RESULTS

The task-wise values corresponding to the LLaMA-2-13B columns of Table 1 are reported below.

Table 22: LLaMA-2-13B perplexities and task-wise results at 70%–90% sparsity. ELSA resultes are taken from (Lee et al., 2026).

		Perplexity (↓)		Tasks (↑)							
Sparsity	**Method**	**Wiki**	**C4**	**ARC-C**	**ARC-E**	**BoolQ**	**HellaSwag**	**OBQA**	**RTE**	**Winogrande**	**Avg.**
0%	DENSE	4.88	6.72	47.10	78.91	82.35	60.21	34.60	68.23	72.22	63.37
70%	MAGNITUDE	218.1	196.4	20.39	31.69	38.17	27.75	16.20	52.71	50.43	33.91
	WANDA	45.19	55.31	18.52	38.47	62.23	29.51	13.80	52.71	53.83	38.44
	SPARSEGPT	19.65	23.25	26.79	54.00	66.73	36.91	20.60	52.71	63.85	45.94
	ALPS	14.13	16.35	29.10	59.68	71.25	41.11	25.20	54.15	65.59	49.44
	ELSA	11.14	12.20	34.13	62.42	70.12	47.52	24.80	52.71	60.38	50.30
	ETAPS	**9.37**	**9.29**	36.52	69.57	74.34	53.25	31.20	64.26	64.88	**56.29**
80%	MAGNITUDE	3.9e4	4.8e4	22.10	25.55	41.41	25.78	14.60	52.71	48.07	32.89
	WANDA	1127	877.2	20.39	26.56	37.83	26.45	14.20	52.71	48.93	32.44
	SPARSEGPT	94.64	78.75	18.43	27.86	54.40	27.86	12.40	52.71	48.86	34.65
	ALPS	38.48	36.99	19.97	35.86	62.45	30.28	13.80	52.71	54.78	38.55
	ELSA	17.21	16.60	24.32	50.97	63.52	38.03	19.60	52.71	53.75	43.27
	ETAPS	**11.93**	**11.10**	30.80	63.47	69.54	47.53	26.60	52.71	60.77	**50.20**
90%	MAGNITUDE	7.4e4	6.4e4	21.33	25.55	42.42	25.62	14.80	45.49	49.64	32.12
	WANDA	2.0e4	1.1e4	20.90	25.76	37.83	26.03	14.00	52.71	51.46	32.67
	SPARSEGPT	1294	911.9	21.16	26.05	61.41	25.85	14.40	52.71	50.75	36.05
	ALPS	241.2	149.8	19.45	27.10	39.11	27.22	13.80	53.43	50.36	32.92
	ELSA	30.19	25.07	19.03	36.15	58.44	28.65	16.20	52.71	50.43	37.37
	ETAPS	**17.50**	**14.89**	22.53	49.87	54.92	38.04	22.40	52.71	53.12	**41.94**

F.3 Cross-Model Perplexity Sweep

Table 23: Perplexity across model families at 50%–90% sparsity.

Model	Method	50%		60%		70%		80%		90%	
		Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
OPT-125M Dense: 27.60 / 26.53 5M delta budget	Magnitude	200.3	146.9	937.9	609.3	3855	2323	4882	3209	6513	4355
	Wanda	38.99	34.85	75.51	62.19	344.4	253.8	1873	1007	4333	3113
	SparseGPT	36.98	33.55	58.76	48.92	233.6	161.3	2090	1142	6987	3418
	ALPS	32.73	30.92	43.13	37.13	89.29	65.68	493.65	265.35	2720	1142
	Elsa	36.44	31.55	39.42	33.32	47.56	38.83	62.63	46.42	95.92	61.68
	ETAPS	**30.79**	**28.17**	**35.14**	**30.62**	**42.85**	**35.06**	**58.96**	**42.61**	**79.84**	**55.44**
OPT-1.3B Dense: 14.61 / 16.06 28M delta budget	Magnitude	1818	412.8	8692	4766	9592	6662	1.6e4	1.1e4	2.7e4	1.7e4
	Wanda	18.39	20.62	26.54	28.34	101.0	92.53	2805	1423	1.3e4	7382
	SparseGPT	17.43	19.22	22.06	23.13	49.91	45.31	954.6	363.9	6416	2783
	ALPS	16.86	18.57	20.40	21.49	35.98	34.09	260.1	161.6	4502	1821
	Elsa	19.55	18.60	21.94	19.91	25.32	23.32	36.56	29.50	60.96	42.99
	ETAPS	**14.84**	**16.65**	**16.70**	**18.00**	**20.26**	**20.33**	**29.40**	**26.61**	**54.64**	**42.30**
Gemma-2-2B Dense: 8.71 / 13.16 42M delta budget	Magnitude	54.04	59.54	2231	1919	2.6e7	2.8e6	1.8e9	3.1e8	2.2e10	1.8e10
	Wanda	13.44	19.52	33.63	47.73	212.4	227.3	3003	1666	1.8e4	1.6e4
	SparseGPT	13.01	18.57	21.25	29.39	65.97	80.75	524.4	439.9	6907	3129
	ALPS	10.94	15.75	14.39	20.27	25.25	35.49	74.87	94.07	255.0	248.1
	Elsa	12.52	16.88	16.53	21.71	20.51	23.79	28.70	30.60	41.94	40.87
	ETAPS	**10.33**	**14.08**	**12.49**	**16.14**	**15.67**	**18.74**	**21.88**	**24.23**	**34.74**	**34.84**
LLaMA-3.2-3B Dense: 7.81 / 11.33 49M delta budget	Magnitude	138.6	214.8	1.5e4	1.4e4	1.0e5	8.1e4	3.5e5	3.5e5	3.0e5	2.3e5
	Wanda	12.99	19.04	31.23	42.57	146.1	183.5	2287	1148	1.3e4	8673
	SparseGPT	12.26	17.39	22.61	29.96	68.92	80.36	268.7	226.0	1498	1132
	ALPS	11.38	15.95	18.26	23.00	41.20	44.99	128.7	105.6	538.4	369.1
	Elsa	12.16	17.05	15.82	20.68	23.27	26.65	34.12	35.78	49.58	48.39
	ETAPS	**10.14**	**13.97**	**12.85**	**16.61**	**17.12**	**19.93**	**23.82**	**25.48**	**36.01**	**35.15**
LLaMA-2-7B Dense: 5.47 / 7.26 80M delta budget	Magnitude	16.00	21.31	1920	2071	5.0e4	2.8e4	1.1e5	1.1e5	4.8e4	5.4e4
	Wanda	6.92	9.24	10.79	14.02	77.08	81.51	2226	2010	6477	5099
	SparseGPT	7.00	9.23	10.20	12.93	27.77	31.37	119.4	100.7	1369	1033
	ALPS	6.85	9.01	9.28	11.33	18.88	20.18	48.20	47.21	232.2	174.5
	Elsa	7.27	9.44	8.36	10.61	12.42	13.69	17.99	17.34	27.98	23.20
	ETAPS	**6.66**	**8.20**	**7.69**	**9.04**	**10.99**	**10.87**	**14.17**	**13.09**	**20.98**	**17.39**
LLaMA-2-13B Dense: 4.88 / 6.72 125M delta budget	Magnitude	6.82	9.37	11.82	14.64	218.1	196.4	3.9e4	4.8e4	7.4e4	6.4e4
	Wanda	5.97	8.29	8.39	11.53	45.19	55.31	1127	877.2	2.0e4	1.1e4
	SparseGPT	6.02	8.22	8.25	10.96	19.65	23.25	94.64	78.75	1294	911.9
	ALPS	**5.92**	8.10	7.54	9.91	14.13	16.35	38.48	36.99	241.2	149.8
	Elsa	6.54	8.78	7.96	9.93	11.14	12.20	17.21	16.60	30.19	25.07
	ETAPS	**5.92**	**7.43**	**6.95**	**8.13**	**9.37**	**9.29**	**11.93**	**11.10**	**17.50**	**14.89**