Hidden in Plain Sight: The Overlooked Significance of Canonical Elements for Extreme LLM Sparsity

Hyeondo Jang **Kwanhee Lee** **Dongyeop Lee** **Namhoon Lee**
POSTECH
{hyeondo.jang, kwanhee.lee, dongyeop.lee2, namhoon.lee}@postech.ac.kr

Abstract

Large language models (LLMs) are often considered fragile under aggressive sparsification, and maintaining reliable performance typically requires sticking to moderate sparsity levels. However, recent studies suggest that LLMs are more resilient to high sparsity than previously thought, reframing the problem as a design challenge rather than a fundamental limitation. In this work, we challenge the perceived limits of unstructured post-training LLM pruning by revisiting elementary pruning strategies that have remained relatively underexplored at this scale. Through a progressive sparsification framework with second-order saliency and continued training coordinated with sparsity progression, we show that pretrained LLMs can retain strong performance far beyond commonly studied sparsity regimes. Across LLaMA-2 and Qwen-3 model families, our approach improves perplexity and downstream accuracy up to 99% sparsity, surpassing both the current state-of-the-art and representative baselines. Precisely, on LLaMA-2-7B, our approach achieves WikiText-2 perplexities of 13.48 and 19.67 at 95% and 99% sparsity, respectively, while delivering 3.23× decoding speedup and 6.21× memory savings at 95% sparsity. Taken together, our results show that LLMs can be pushed into extreme sparsity while retaining strong performance, providing a foundation for further improving sparse models in this regime.

1 Introduction

As large language models (LLMs) continue to scale, their deployment incurs substantial compute and memory costs (Brown et al., 2020). Sparsification, also widely known as pruning, is a critical strategy to mitigate these overheads, offering the potential to drastically reduce model size and accelerate inference (Hoefler et al., 2021; Agarwalla et al., 2024). Nevertheless, much of the literature has tended to report sharp degradation in LLM performance beyond moderate sparsity levels around 50–70%, contributing to a prevailing perception that such degradation may reflect a potential intrinsic limitation of LLMs (Frantar and Alistarh, 2023; Sun et al., 2024).

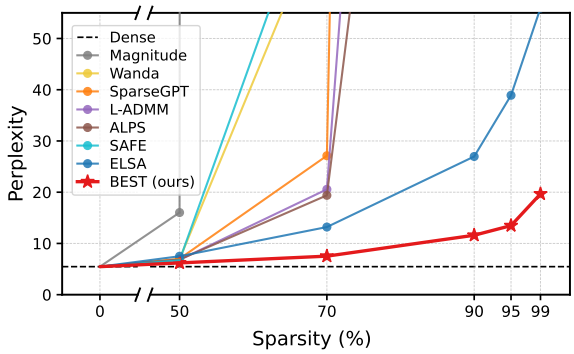


Figure 1: WikiText-2 perplexity of LLaMA-2-7B pruned across a range of sparsity levels. We show that extreme sparsity is more realistically attainable than previously suggested. See Appendix A.1 for configuration details of the prior pruning approaches shown here.

However, recent work has provided indications that these commonly held preconceptions may warrant reevaluation on multiple fronts (Agarwalla et al., 2024; Lee et al., 2026). In particular, emerging evidence suggests that modern LLMs can remain functional well beyond the moderate sparsity regimes often treated as a practical ceiling (Agarwalla et al., 2024; Lee et al., 2026). Notably, Lee et al. (2026) reaches functional performance at 90% sparsity by performing appropriate optimization through continued training, suggesting that the attainable sparsity of pre-trained LLMs may be higher than previously expected.

This prompts a fundamental inquiry: *to what extent can extreme sparsity be pushed, when paired with a carefully tuned optimization procedure?* Yet this question has remained largely underexplored: the prevailing assumptions that LLMs are too large for retraining or hyperparameter search have fa-

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 61372–61394
October 24-29, 2026 ©2026 Association for Computational Linguistics

vored resource-efficient pruning protocols, which may offer only a partial view of the true sparsity potential in LLMs.

Motivated by this gap, we first revisit the elementary strategies of sparsification that drove neural network pruning: second-order saliency (LeCun et al., 1989) and progressive sparsification (Zhu and Gupta, 2017), each of which has garnered limited attention in post-training pruning of modern LLMs. We study how these ingredients should be adapted and scaled for post-training LLM pruning in a continued-training setting. Surprisingly, we show that, when properly coordinated, these strategies alone can push attainable sparsity in LLMs well beyond previously demonstrated levels.

Across LLaMA-2 and Qwen-3 model families, our approach consistently improves both perplexity and downstream accuracy over prior approaches from 70% to 99% sparsity, with the largest gains in the extreme regime, as previewed on LLaMA-2-7B in Figure 1 and further demonstrated in Section 4. Importantly, these gains are not limited to parameter-count reduction: with sparse execution kernels, our pruned LLaMA-2-7B model achieves a $3.23\times$ decoding speedup and a $6.21\times$ reduction in memory footprint over the dense counterpart.

2 Background

2.1 Pruning

Pruning aims to find sparse neural networks whose weight matrices are mostly populated by zeros, so as to skip corresponding computation and memory allocation while preserving task performance (Hoefler et al., 2021). A long line of work has demonstrated that pruning can achieve high compression rates with minimal accuracy loss (LeCun et al., 1989; Han et al., 2015; Hoefler et al., 2021), with resulting sparsity translated into practical speedup through structured formats accelerated on modern GPUs (semi-structured $N{:}M$ sparsity, *e.g.*, 2:4 (Mishra et al., 2021; Zhou et al., 2021), block-structured (Gray et al., 2017)), or through custom SpMV/SpMM kernels for unstructured patterns (Gale et al., 2020; Macko and Boža, 2026).

Early approaches can largely be characterized by their reliance on a single pruning step (commonly known as one-shot pruning) over the entire training procedure, either at initialization (Lee et al., 2019; Tanaka et al., 2020; Wang et al., 2020), or post-hoc on a pretrained model (LeCun et al., 1989; Singh and Alistarh, 2020). However, consistently stronger results follow when pruning during training interleaves sparsity with optimization, at the cost of additional compute—either by gradually increasing the sparsity target (Zhu and Gupta, 2017; Kurtic and Alistarh, 2022), or with allowance for regrowth as in dynamic sparse training (Mocanu et al., 2018; Wortsman et al., 2019). Among these, iterative pruning (Frankle and Carbin, 2018; Renda et al., 2020), which alternates between pruning and retraining over several cycles, stands out for consistently recovering the strongest sparse subnetworks.

Another key design choice in a pruning strategy is parameter selection: how parameters are chosen for removal. One widely adopted heuristic scores each parameter by its estimated contribution to the loss, using signals ranging from raw magnitude (Han et al., 2015) to gradient–weight products (Lee et al., 2019) to second-order curvature (LeCun et al., 1989; Singh and Alistarh, 2020). Alternatively, learning-based methods forgo explicit scoring altogether, instead learning the sparsity pattern jointly with the weights via learnable thresholds (Kusupati et al., 2020) or relaxed binary masks. A separate line of work casts pruning as a constrained optimization problem, with solutions ranging from regularization (Louizos et al., 2017; Wen et al., 2016) to iterative hard thresholding (Peste et al., 2021) and Frank-Wolfe (Lu et al., 2022).

2.2 Pruning in LLMs

With the advent of large language models, however, this design space has largely converged onto a single corner. Early efforts in LLM pruning focused on computationally inexpensive procedures, driven by the rapid scaling of LLMs that rendered conventional pruning pipelines infeasible. Post-training, one-shot pruning with layer-wise surrogate objectives (Frantar and Alistarh, 2023; Sun et al., 2024) emerged as the dominant approach, demonstrating that moderate sparsity could be obtained with minimal cost and overhead. These methods established the feasibility of pruning large models, but their reliance on these surrogate objectives imposed a sharp ceiling: they consistently fail to maintain functional performance much beyond roughly 70% sparsity (Meng et al., 2024; Huang et al., 2025a), while the compression levels at which meaningful hardware benefits materialize lie considerably higher (Agarwalla et al., 2024; Gale et al., 2020).

A growing body of work has sought to push past this ceiling while staying within this paradigm, through better optimization of the layer-wise objec-

tive (Boža, 2024; Meng et al., 2024), non-uniform sparsity allocation across layers (Yin et al., 2024; Sieberling et al., 2025), or lightweight fine-tuning to compensate for pruning error (Zhang et al., 2024; Huang et al., 2025b). These refinements deliver steady but modest gains; none reliably prevent the extreme-sparsity collapse. More recent work questions the template itself, broadening the surrogate from per-layer to block-wise or whole-model objectives (Shin et al., 2024; Bai et al., 2024). Notably, Lee et al. (2026) argue that the field's reliance on cost-minimal, one-shot, layer-wise surrogates has been an unduly restrictive self-imposed constraint. Relaxing this constraint, their iterative, surrogate-free approach demonstrates that functional performance can be retained at 90% sparsity.

Taken together, these results indicate that the prevailing limit on LLM sparsity is not intrinsic, but a consequence of the restricted paradigm through which the field has operated. Motivated by this, we ask how far extreme sparsity, where practical hardware benefits are substantial, can be pushed when essential pruning components given little attention at modern LLM scale are properly applied.

3 Method

Let us start by formulating pruning as a constrained optimization problem:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_0 \leq k, \tag{1}$$

where we minimize the loss $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ through optimizing the model parameters $\mathbf{w} \in \mathbb{R}^d$ with $k < d$ number of non-zero weights computed by the L_0-norm $\|\cdot\|_0$.

While many strategies exist for solving Equation (1) (Zhu and Gupta, 2017; Frankle and Carbin, 2018; Evci et al., 2020; Liu et al., 2020), we build upon the progressive sparsification framework (Zhu and Gupta, 2017), a simple yet flexible procedure whose design space exposes the key components governing sparsification performance.

Within this framework, we decompose the pruning process as:

$$\mathbf{m}^{t+1} = \mathcal{M}^t(\mathbf{m}^t \odot \mathbf{w}^t), \tag{2}$$

$$\text{s.t.} \quad d = \|\mathbf{m}^0\|_0 \geq \cdots \geq \|\mathbf{m}^T\|_0 = k, \tag{3}$$

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{m}^{t+1} \odot \mathbf{w}), \tag{4}$$

where we alternate between the masking operation $\mathcal{M}^t : \mathbb{R}^d \to \{0,1\}^d$ in Equation (2) (subject to the sparsity schedule in Equation (3)) and the weight training for the remaining parameters in Equation (4). The process is initialized with the pretrained weights, setting $\mathbf{w}^0 = \mathbf{w}_{\text{pre}}$ with a dense starting mask $\mathbf{m}^0 = \mathbf{1}$.

The decomposition above exposes three design components governing sparsification performance:

- **Masking strategy** (2) by designing saliency criteria to determine which parameters in $\mathbf{w}$ to remove, along with the sparsity allocation across modules (Section 3.1);
- **Sparsity schedule** (3) by specifying the progression of $\|\mathbf{w}\|_0$ towards the target k (Section 3.2);
- **Training strategy** (4) by optimizing the remaining active parameters in $\mathbf{w}$ for recovery (Section 3.3).

The following subsections present our design choices for each.

3.1 Masking strategy

The ideal design for the masking function $\mathcal{M}^t$ in Equation 2 would be one that solves the combinatorial subproblem

$$\arg\min_{\|\mathbf{m}\|_0 \leq k} \mathcal{L}(\mathbf{m} \odot \mathbf{w}^t). \tag{5}$$

Solving Equation (5) exactly is intractable, so we pursue a tractable saliency-based strategy: a second-order saliency with an optimizer-aware proxy for efficient saliency computation, and global pruning for dynamic sparsity allocation.

Second-order saliency A common heuristic for Equation (5) is to design a saliency criterion (Hoefler et al., 2021), assigning a scalar importance score s_i to each parameter w_i based on the expected increase in loss $\mathcal{L}$ upon its removal (Han et al., 2015; Lee et al., 2019; LeCun et al., 1989). Second-order pruning criteria (LeCun et al., 1989) estimate the loss change induced by removing parameters using a local Taylor expansion:

$$\Delta\mathcal{L} \approx \mathbf{g}^\top \Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^\top H \Delta\mathbf{w}. \tag{6}$$

Since pre-trained models are already converged, the gradient term is expected to be small, i.e., $\mathbf{g} \approx \mathbf{0}$. Under a diagonal Hessian approximation, this yields the following saliency score for pruning parameter w_i:

$$s_i^{\text{2nd}} = \frac{1}{2} H_{ii} w_i^2. \tag{7}$$

This criterion incorporates both parameter magnitude and local loss curvature, providing a direct proxy for the loss increase induced by pruning each parameter. Despite its appeal, directly using Equation (7) at LLM scale is impractical, as estimating Hessian information for models with billions of parameters is prohibitively expensive.

Optimizer-aware proxy for efficient saliency computation To avoid exact Hessian computation, we approximate H by the Generalized Gauss-Newton matrix (Botev et al., 2017), which coincides with the Fisher Information Matrix F under the standard cross-entropy objective (Martens, 2020). For converged models, F is commonly approximated by the Empirical Fisher $F = \mathbb{E}[gg^\top]$, whose diagonal entries $F_{ii} = \mathbb{E}[g_i^2]$ provide an efficient curvature proxy.

Even computing F_{ii}, however, requires an extra pass to collect and average squared gradients across data samples—a bottleneck at LLM scale. We instead reuse the running statistics already maintained by the Adam optimizer (Kingma and Ba, 2014) during retraining: its exponential moving average of squared gradients, $\hat{v}^t$, has been shown to approximate the diagonal Fisher (Li et al., 2025). Substituting $\hat{v}^t$ for F_{ii} in Equation (7) yields a zero-cost saliency score,

$$\hat{s}_i^{\text{2nd}} = \frac{1}{2}\,\hat{v}_i^t\, w_i^2, \tag{8}$$

giving curvature-aware pruning at no additional gradient-computation cost.

Global pruning for dynamic sparsity allocation Once a saliency criterion is chosen, pruning still requires specifying the group within which scores are compared. In principle, global comparison — ranking all parameters jointly under a single threshold — is the most faithful realization of the constrained objective in Equation (5), allocating sparsity in proportion to per-parameter importance without any predetermined per-layer ratio. Prior post-training LLM pruners are nonetheless confined to layer-wise comparison (Frantar and Alistarh, 2023; Sun et al., 2024), a restriction inherited from their layer-wise, local surrogate objectives that offer no signal for cross-layer ranking, forcing uniform per-layer sparsity and motivating separate sparsity-allocation strategies (Yin et al., 2024; Sieberling et al., 2025). Our second-order saliency, by contrast, directly approximates the global loss change and is therefore comparable across modules, admitting a single global threshold without such strategies. We accordingly adopt pruning by global comparison. Implementation-wise, we make this global threshold tractable under distributed training of LLMs via a distributed binary search that keeps every score in place and exchanges only scalar counts, as detailed in Appendix B.1.

3.2 Sparsity schedule

The constraint in Equation (3) only specifies that $\|\mathbf{m}^n\|_0$ must monotonically decrease from d to k; the trajectory is a design choice. Directly projecting $\mathbf{w}_{\text{pre}}$ to the target sparsity can cause a sudden loss of pretrained knowledge, while an overly slow sparsification process may repeatedly interfere with optimization and prevent efficient adaptation. To balance these effects, we employ a cubic gradual sparsity schedule, which is shown effective in prior work (Zhu and Gupta, 2017; Kurtic and Alistarh, 2022), and our ablation (Appendix C.2) as follows:

$$S_t = S_{\text{final}} + (S_{\text{init}} - S_{\text{final}})\left(1 - \frac{t - t_0}{T_{\text{prune}}}\right)^3, \tag{9}$$

where S_{init} and S_{final} denote the initial and target sparsity, t_0 is the step at which pruning begins, and T_{prune} is its duration. In our work, S_{init} is set to 0, although other choices are possible. After step $t_0 + T_{\text{prune}}$ the mask is frozen at S_{final} and training continues until the total budget is exhausted.

3.3 Training strategy

The weight-update step in Equation (4) must reconcile two demands of post-training pruning: keeping the early learning rate small to avoid disturbing the near-converged $\mathbf{w}_{\text{pre}}$, and providing a sufficiently large learning rate later to absorb the loss perturbations introduced by the sparsity ramp in Equation (9).

This leads to a learning rate that starts small, grows with the sparsity ramp, and decays toward convergence—in practice, the standard linear warmup followed by linear decay,

$$\eta_t = \begin{cases} \eta_{\text{target}} \cdot \frac{t}{T_{\text{warmup}}}, & \text{if } t \le T_{\text{warmup}}, \\ \eta_{\text{target}} \cdot \frac{T-t}{T-T_{\text{warmup}}}, & \text{if } t > T_{\text{warmup}}, \end{cases} \tag{10}$$

where T_{warmup} is the warmup length and T is the total number of steps. We empirically find that a short warmup suffices and set $T_{\text{warmup}} = 0.1\,T$ throughout.

Table 1: Perplexities (↓) of LLaMA-2 and Qwen-3-Base models pruned to various sparsity levels using different methods. BEST consistently outperforms baselines across all settings, with the margin widening at higher sparsity.

		LLaMA-2				Qwen-3			
		7B		13B		4B-Base		8B-Base	
Sparsity	Method	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
0%	Dense	5.47	7.26	4.88	6.73	7.82	11.58	6.93	10.41
70%	Magnitude	4.4e4	3.3e4	212.6	163.7	5.8e5	5.6e5	2.2e6	1.1e6
	Wanda	69.91	81.88	44.41	46.24	60.77	71.14	50.17	58.61
	SparseGPT	26.02	29.85	21.28	21.30	27.95	32.93	20.43	24.74
	Magnitude$^+$	7.72	10.04	6.74	9.09	11.00	16.32	9.57	15.05
	Wanda$^+$	8.06	10.44	6.94	9.54	10.83	16.05	9.57	14.95
	SparseGPT$^+$	7.74	10.07	6.84	9.09	10.68	16.92	9.27	14.62
	BEST	**7.52**	**9.56**	**6.64**	**8.89**	**10.15**	**15.40**	**9.00**	**14.06**
90%	Magnitude	NaN	NaN	8.4e4	6.4e4	7.9e5	8.3e5	2.7e6	5.7e6
	Wanda	9313	7419	2.3e4	1.4e4	1.9e5	1.7e5	2.8e4	1.8e4
	SparseGPT	1514	908.4	1389	797.1	1632	1319	369.1	264.8
	Magnitude$^+$	13.18	15.40	11.51	14.94	24.94	31.38	20.55	26.40
	Wanda$^+$	15.61	18.82	13.89	17.02	21.58	28.30	18.22	24.85
	SparseGPT$^+$	13.41	16.04	11.79	15.23	18.95	25.01	16.14	22.76
	BEST	**11.60**	**13.71**	**10.04**	**12.68**	**14.90**	**20.79**	**13.49**	**19.18**
95%	Magnitude	1.7e5	NaN	5.3e4	3.6e4	2.2e5	2.6e5	1.8e6	1.8e6
	Wanda	1.4e4	1.5e4	1.6e5	1.2e5	1.1e4	7076	1.1e5	9.2e4
	SparseGPT	5619	6186	3088	2174	3.3e4	2.7e4	1857	1179
	Magnitude$^+$	20.52	21.50	45.48	41.05	41.11	44.81	35.63	40.69
	Wanda$^+$	21.87	22.47	19.14	21.73	28.86	35.88	26.42	33.52
	SparseGPT$^+$	18.75	20.72	16.25	19.55	29.18	36.49	23.83	30.64
	BEST	**13.48**	**16.52**	**12.72**	**15.33**	**17.67**	**22.86**	**16.08**	**22.08**
99%	Magnitude	NaN	NaN	2.8e4	3.1e4	2.0e5	3.2e5	1.1e5	1.2e5
	Wanda	6.4e5	7.4e5	1.5e4	2.4e4	1.0e5	9.9e4	3.2e5	2.1e5
	SparseGPT	NaN	NaN	8466	3886	4.2e4	5.4e4	3.8e4	4.0e4
	Magnitude$^+$	50.40	45.02	67.79	57.06	77.73	71.73	50.32	52.88
	Wanda$^+$	44.52	38.61	35.98	35.34	60.13	64.93	46.12	51.08
	SparseGPT$^+$	35.53	32.39	31.36	31.28	69.93	70.16	38.03	43.68
	BEST	**19.67**	**22.03**	**22.07**	**24.15**	**25.96**	**31.98**	**23.39**	**29.51**

Together, these three components form our complete framework: a global second-order saliency with an optimizer-aware proxy for masking, a cubic schedule for sparsity progression, and a learning-rate schedule coordinated with the sparsity progression for weight updates. We call this framework BEST, reflecting our goal of pursuing the best performance by designing these components jointly. We analyze the design principles behind each choice in Section 4.4.

4 Experiments

We report our main language-modeling and zero-shot results at extreme sparsity in Section 4.1 and comparisons with recent pruning methods in Section 4.2. We then study the practical value of the resulting sparsity in Section 4.3 and analyze the design and cost of BEST in Sections 4.4 and 4.5.

Setup We evaluate our method on LLaMA-2 (Touvron et al., 2023), Qwen2.5 (Yang et al., 2024), and Qwen-3 (Yang et al., 2025), using WikiText-2 (Merity et al., 2017) and C4 (Raffel et al., 2020) datasets for language modeling evaluation and `lm-eval-harness` for zero-shot generalization performance. Detailed setup for training and evaluation can be found at Appendix B.2.

Baselines We compare BEST against representative one-shot methods: Magnitude (Han et al., 2015), Wanda (Sun et al., 2024), and SparseGPT (Frantar and Alistarh, 2023). To control for training budget, we also evaluate each one-shot method followed by standard retraining, in which the remaining weights are fine-tuned with the mask fixed after pruning, under the same budget as BEST (denoted $^+$). We further compare against recent pruning methods: L-ADMM (Boža, 2024), ALPS (Meng

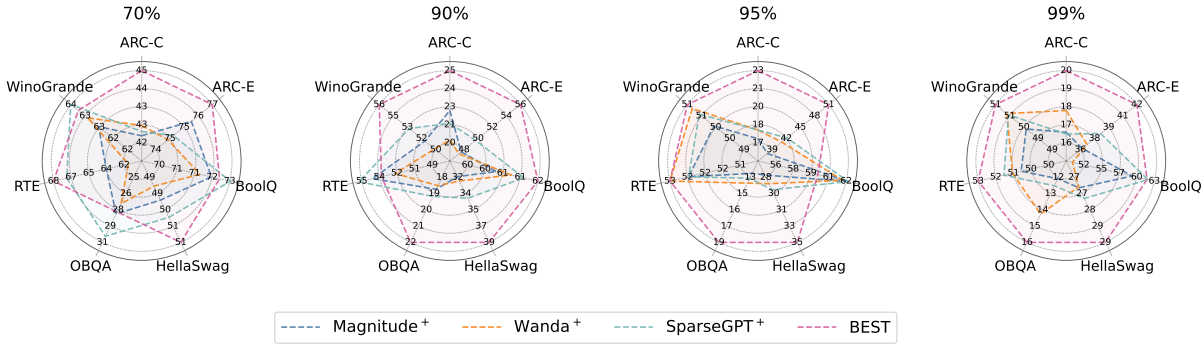


Figure 2: Qwen3-8B zero-shot accuracy across seven tasks at each sparsity. BEST outperforms one-shot methods followed by retraining on most tasks. Full numerical results are provided in Appendix D.5.

Table 2: WikiText-2 perplexity of Qwen2.5-32B. The dense baseline has perplexity 5.12.

Method	Sparsity			
	70%	90%	95%	99%
SparseGPT$^+$	7.28	13.93	24.30	37.60
BEST	**6.71**	**10.08**	**12.98**	**21.85**

Table 3: Comparison with recent pruning methods on LLaMA-2-7B. Baseline numbers are taken from Lee et al. (2026) and therefore serve as contextual comparisons.

Method	Sparsity					
	70%			90%		
	Wiki	C4	AZS	Wiki	C4	AZS
L-ADMM	20.56	22.20	44.22	400.5	287.1	31.88
ALPS	19.39	20.37	45.03	248.8	180.9	32.03
SAFE	86.80	48.54	38.94	1.6e4	1.6e4	32.63
SparseLLM	37.65	35.00	40.54	1267	648.0	31.93
ELSA	13.20	14.08	46.21	26.97	23.14	38.52
BEST	**7.52**	**9.56**	**52.06**	**11.60**	**13.71**	**42.34**

et al., 2024), SAFE (Lee et al., 2025), SparseLLM (Bai et al., 2024), and ELSA (Lee et al., 2026).

4.1 Main results

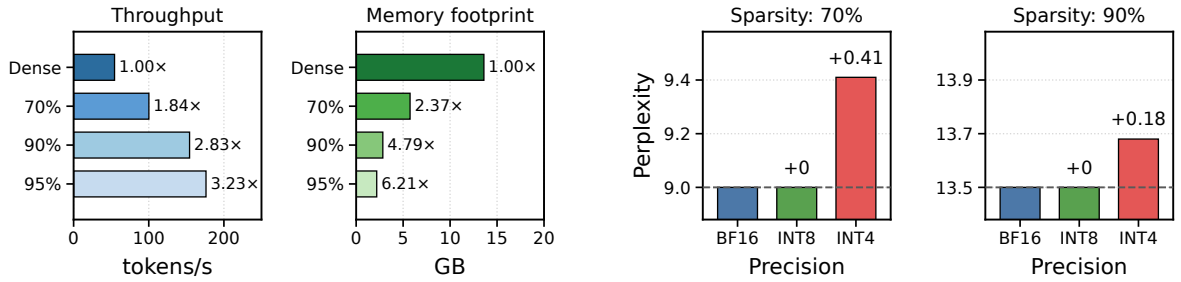
Language modeling performance Table 1 reports WikiText-2 and C4 perplexity for LLaMA-2 and Qwen-3 models from 70% to 99% sparsity. BEST consistently outperforms all baselines, with the margin widening as sparsity increases. In the extreme regime (95%–99%), one-shot pruning collapses outright; retraining substantially mitigates this, yet even the strongest baseline (SparseGPT$^+$) reaches only 35.53 at 99% on LLaMA-2-7B WikiText-2, nearly double BEST's 19.67.

The same pattern holds across scales and model families: at 99%, BEST achieves 22.07 on LLaMA-2-13B and 25.96 on Qwen3-4B-Base, against 31.36 and 60.13 for the respective strongest baselines. This trend extends to the larger-scale Qwen2.5-32B model, as reported in Table 2, where BEST outperforms SparseGPT$^+$ at all sparsity levels. These results demonstrate that standard retraining pipelines are insufficient for extreme compression, validating the necessity of an optimization procedure specifically designed for this regime.

Zero-shot downstream task performance Figure 2 reports per-task zero-shot accuracy for Qwen3-8B from 70% to 99% sparsity. BEST consistently outperforms all baselines, with a clear gap emerging at extreme sparsity. At 70%, methods are competitive: ARC-E 76.35 vs. 74.79 and HellaSwag 51.17 vs. 50.35 for SparseGPT$^+$, but SparseGPT$^+$ drops 32.7 and 21.2 %p on these tasks by 95%. BEST retains 50.76 and 34.91 at 95% and 42.21/29.49 at 99%, outperforming SparseGPT$^+$ on most tasks and on the average at both sparsity levels. This pattern holds across all four model variants (Appendix D.5), demonstrating that BEST consistently outperforms existing retraining pipelines on downstream tasks across all sparsity levels.

4.2 Further comparisons with recent pruning methods

To strengthen our empirical evaluation, we compare BEST with recent pruning methods on LLaMA-2-7B in Table 3, including ELSA, which, to the best of our knowledge, established the prior state of the art at extreme sparsity. BEST outperforms all baselines at both 70% and 90% sparsity in perplexity on both WikiText-2 and C4, with the margin widening substantially at 90% sparsity. At 90% sparsity, BEST achieves 11.60 WikiText-2 perplexity and 13.71 C4 perplexity, compared with 26.97 and 23.14, respectively, for ELSA, the



(a) Decoding throughput and memory footprint.

(b) Perplexity change under RTN quantization.

Figure 3: (a) End-to-end decoding throughput (↑) and memory footprint (↓) of LLaMA-2-7B. (b) WikiText-2 perplexity of Qwen-3-8B-Base after RTN quantization (dense BF16: 6.93); INT8 is lossless, while INT4 causes only minor degradation.

strongest baseline. The same trend holds for average zero-shot accuracy over the seven downstream tasks evaluated in Section 4.1, with BEST achieving 42.34 versus 38.52 for ELSA at 90% sparsity. We extend our evaluation to 95% and 99% sparsity and confirm that BEST continues to outperform ELSA, achieving WikiText-2 perplexities of 13.48 and 19.67, respectively, compared with 38.91 and 55.94 for ELSA, as detailed in Appendix D.1.

4.3 Validating sparsity benefits

In this subsection, we assess the practical value of sparsity achieved by BEST. We evaluate whether the resulting sparse models deliver inference efficiency and remain amenable to further compression via quantization.

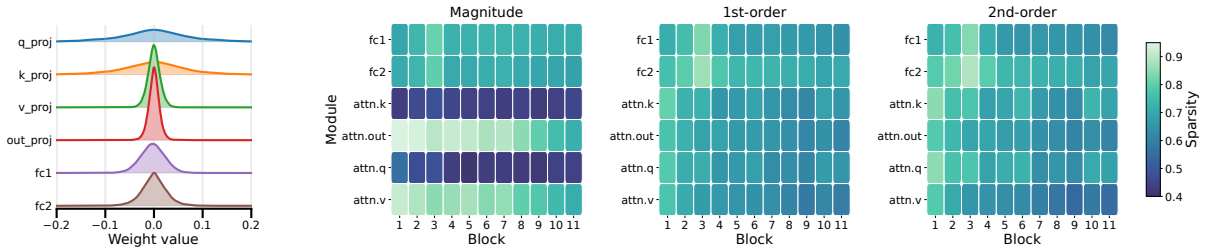
Inference speedup & memory savings To assess whether parameter reduction leads to actual inference gains, we evaluate our pruned LLaMA-2-7B model in one concrete deployment setting, using the recent SpMV kernel (Macko and Boža, 2026) on a single RTX 3090, following its batch-size-1, 100-token decode-only protocol. Figure 3a reports end-to-end decoding throughput and GPU memory footprint including sparse index metadata across sparsity levels. At 95% sparsity, our model achieves **3.23**× higher decoding throughput than the dense baseline (176.18 vs. 54.47 tokens/s) and a memory footprint **6.21**× lower (2.19 GB vs. 13.6 GB), providing preliminary evidence that extreme sparsity can translate into practical inference gains in this setting. Notably, this 95%-sparse model attains language modeling quality comparable to ELSA's 70%-sparse counterpart (WikiText-2 PPL 13.48 vs. 13.20; Tables 1 and 3). In other words, BEST realizes the above speedup and memory gains while preserving the language modeling quality of prior sparse models.

Pruning and quantization Stable sparse models can also be further compressed through quantization. We apply simple round-to-nearest (RTN) post-training quantization to the remaining nonzero weights of BEST-pruned Qwen-3-8B-Base. As shown in Figure 3b, INT8 quantization is almost lossless at either sparsity level, while INT4 incurs a modest increase at 70% (+0.41), and an even smaller degradation at 90% (+0.18). These results suggest that the sparse models produced by BEST remain stable under low-precision compression, providing a simple path to combine pruning and quantization for further compression.

4.4 Ablation study

In this section, to assess the contribution of each to extreme-sparsity performance, we conduct an ablation study on the three components of BEST: masking strategy (Section 3.1), progressive sparsity schedule (Section 3.2), and training strategy (Section 3.3). All variants share the same training and evaluation setup; full details and additional sensitivity analyses are in Appendix C.

Masking strategy We compare three saliency criteria: magnitude ($|w_i|$), which assigns importance by parameter scale alone; first-order, which captures sensitivity via the gradient term from Equation (6) ($|g_i w_i|$); and second-order, which additionally accounts for curvature. Following the same zero-cost principle, both gradient-based criteria substitute Adam optimizer states for raw gradients: $|\hat{m}_i^t w_i|$ for first-order and $\frac{1}{2}\hat{v}_i^t w_i^2$ for second-order (see Appendix C.4 for additional ablation on using optimizer states). Each criterion is evaluated under both layer-wise and global comparison.



(a) Weight distribution

(b) Sparsity allocation heatmaps at 70% sparsity

Figure 4: (a) Per-module weight distributions in OPT-125M first block, shown as a ridgeline with one row per module; different scales make magnitude unreliable as an importance proxy. (b) Per-module sparsity at 70% across saliency criteria; magnitude assigns excessive sparsity to attention projections, nearly collapsing them.

Table 4: Effect of saliency and comparison group on OPT-125M perplexity; 2nd-order consistently achieves the best, and global pruning further improves it.

		Sparsity			
		70%		90%	
Saliency	Group	Wiki	C4	Wiki	C4
Magnitude	Layer	**32.44**	**31.29**	**47.62**	**42.28**
	Global	34.54	31.41	54.79	46.43
1st-order	Layer	36.61	34.85	56.32	48.07
	Global	**36.49**	**33.00**	**54.30**	**45.50**
2nd-order	Layer	31.28	30.84	44.49	41.23
	Global	**30.77**	**29.16**	**43.84**	**39.15**

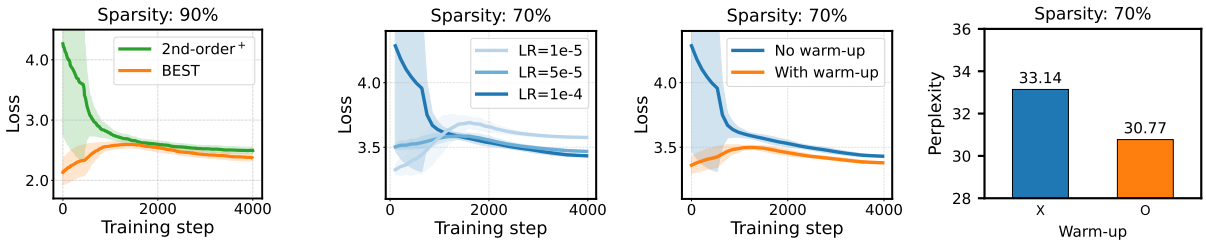
Table 4 shows that global second-order pruning achieves the best configuration across all sparsity levels. Notably, second-order saliency improves further under global comparison (-0.65 at 90% Wiki), indicating that its scores are reliable across modules, whereas magnitude degrades under the same setting ($+7.17$ at 90% Wiki). To understand this gap, we examine the behavior of magnitude saliency under global sparsity allocation. As shown in Figure 4 (a), weight distributions vary substantially across modules, causing magnitude-based global allocation to be biased by module scale rather than functional importance. This bias is reflected in Figure 4 (b), where magnitude saliency disproportionately allocates sparsity: it over-prunes (`attn.out`, `attn.v`), nearly collapsing them even at 70% global sparsity, while leaving (`attn.q`, `attn.k`) relatively dense. By contrast, first- and second-order criteria yield more balanced allocations, with second-order saliency further improving performance by incorporating curvature. Additional analysis is provided in Appendix D.2.

Sparsity schedule To validate the necessity of progressive sparsification, we compare BEST against 2nd-order$^+$, which removes only the gradual sparsity schedule by performing a single pruning step using BEST's saliency before retraining. As shown in Figure 5a, one-shot pruning imposes the target sparsity abruptly, driving the model into a high-loss regime early in optimization and leading to suboptimal solutions. In contrast, our progressive sparsification avoids this failure mode by introducing sparsity incrementally and maintaining stable optimization, yielding a final loss of 2.38 and perplexity of 11.60 at 90% sparsity, compared with 2.49 and 14.11 for 2nd-order$^+$, respectively. These results support the necessity of a progressive schedule to reach extreme sparsity reliably. Additional results across sparsity levels, including the resulting perplexities and a comparison with a competitive retrained one-shot baseline, are provided in Appendix C.3.

Training strategy With progressive sparsification in place, we next ablate the training strategy. Figure 5b (left) shows that a larger learning rate is needed for better convergence under high sparsity. However, applying this large learning rate immediately to pretrained models causes an initial loss spike (Figure 5b, middle), whereas a short warm-up mitigates the instability and yields lower perplexity (Figure 5b, right). The same improvement carries over to 90% sparsity; additional results and warm-up-length ablations are in Appendix C.1.

4.5 Cost analysis and budget-matched comparisons

Table 5 reports the pruning cost and resulting perplexity at 95% sparsity. SparseGPT, a representative one-shot method, is highly cost-efficient, completing in only 0.25 hours on a single GPU. However, its perplexity of 5619 highlights the limitation of cheap one-shot pruning in this extreme regime, motivating additional optimization and shifting the relevant comparison to model quality at compara-



(a) One-shot vs. Progressive

(b) Learning Rate Warmup

Figure 5: (a) Training loss curves for LLaMA-2-7B at 90% sparsity. 2nd-order$^+$ is BEST without the gradual schedule: one-shot pruning causes a catastrophic loss increase, while progressive sparsification maintains stable training. (b) Validation loss under different LRs on OPT-125M at 70% sparsity (left): larger LRs are required but applying them directly destabilizes training; a short warm-up resolves the instability by ramping LR gradually (middle), yielding consistently lower WikiText-2 perplexity (right).

Table 5: Cost of pruning LLaMA-2-7B to 95% sparsity on A100 80GB GPUs and the resulting WikiText-2 perplexity for each run.

Method	Training tokens	#GPUs	Wall-clock (h)	PPL
SparseGPT	0	1	0.25	5619
ELSA	67M	4	1.87	38.91
BEST‡	67M	4	1.74	**24.34**
SparseGPT$^+$	524M	4	10.4	18.75
BEST	524M	4	10.3	**13.48**

ble training cost.

To examine this trade-off under continued training, we first compare BEST with ELSA (Lee et al., 2026). ELSA established the prior state of the art at extreme sparsity by directly optimizing the original language-modeling objective via ADMM (Boyd et al., 2011). ELSA uses C4 with a 67M-token budget, whereas our main setting uses SlimPajama with a 524M-token budget because BEST remains substantially undertrained under ELSA's setting. For a controlled cost analysis, we run BEST using the same dataset and token budget as ELSA and denote this matched variant as BEST‡. BEST‡ achieves lower perplexity than ELSA (24.34 vs. 38.91) with slightly lower wall-clock time (1.74 vs. 1.87 hours). This advantage persists at other sparsity levels, as detailed in Appendix D.1. Turning to our main 524M-token setting, BEST also outperforms SparseGPT$^+$ at comparable wall-clock time (13.48 vs. 18.75 perplexity; 10.3 vs. 10.4 hours).

Overall, under matched training budgets, BEST achieves strong performance without additional wall-clock overhead, demonstrating a favorable performance–cost trade-off enabled by an efficient implementation that reuses optimizer states for saliency estimation and employs distributed binary search for global thresholding.

5 Conclusion

Extreme sparsity in LLMs is an increasingly important problem, and a growing body of work has sought to push the limits of how sparse a model can be while retaining its capabilities. In this work, we find that faithfully applying essential components of the pruning process proves sufficient to establish a new state of the art, with the resulting sparse models yielding measurable inference speedup on commodity hardware. We hope our framework and analysis provide a foundation for further innovations on extreme LLM sparsity, an active research target.

Limitations

Several limitations remain in this work, which we plan to address in future research.

First, although BEST achieves strong performance at extreme sparsity, its optimization procedure requires continued training and thus incurs additional computational cost. Reducing this cost remains important, and selecting training data so that fewer tokens suffice (Lin et al., 2024; Xia et al., 2024) is one promising direction.

Second, despite the minimal increase in perplexity, the decline in downstream task performance may limit the practical utility of these compressed models. This limitation becomes more pronounced in free-form generation and reasoning: as reported in Appendix D.4, performance on GSM8K and NQ-Open deteriorates, particularly in the extreme-sparsity regime. Our pipeline operates on a base pretrained model and therefore defers downstream performance recovery to solution-quality-oriented techniques such as post-training alignment (Touvron et al., 2023; Yang et al., 2025). Exploring the

synergy between extreme sparsification and modern alignment pipelines such as GRPO (Shao et al., 2024), together with targeted data curation (Penedo et al., 2023; Li et al., 2024), is a natural direction for recovering nuanced reasoning capabilities.

Moreover, our experiments focus on dense Transformers. Whether the same design principles extend to other architectures, such as Mixture-of-Experts models (Jiang et al., 2024), is an important question that we leave to future work.

Translating strong performance at extreme sparsity into practical inference gains remains contingent on a dedicated sparse kernel (Macko and Boža, 2026) in the absence of native hardware support for irregular sparsity (Agarwalla et al., 2024). This dependence exposes a deployment trade-off: structured and semi-structured methods adopt hardware-friendly patterns that make acceleration more readily attainable on commodity hardware (Mishra et al., 2021; Zhou et al., 2021), albeit with greater difficulty preserving model quality under stronger structural constraints. More efficient kernels and broader hardware support are therefore important for fully realizing the value of extreme sparsity.

Finally, a practically relevant comparison is against small dense models at comparable effective parameter counts. We outperform such counterparts even with our static framework (Appendix D.3), but the margin remains narrow; incorporating dynamic sparsity mechanisms (Evci et al., 2020; Liu et al., 2020) or knowledge distillation (Muralidharan et al., 2024) may be one way to widen it.

Acknowledgments

This work was partly supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH); RS-2024-00457882, National AI Research Lab Project; RS-2026-25507427, Development of Efficient Architectures and Training Techniques for High-Performance Lightweight AI Models), the Korea Basic Science Institute (National Research Facilities and Equipment Center) grant funded by the Korean government (MSIT) (RS-2026-25500419) and the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2023-00210466, RS-2025-02264052). Hyeondo Jang was supported by Kwanjeong Educational Foundation Scholarship.

References

Abhinav Agarwalla, Abhay Gupta, Alexandre Marques, Shubhra Pandit, Michael Goin, Eldar Kurtic, Kevin Leong, Tuan Nguyen, Mahmoud Salem, Dan Alistarh, Sean Lie, and Mark Kurtz. 2024. Enabling high-sparsity foundational LLaMA models with efficient pretraining and deployment. *arXiv preprint arXiv:2405.03594*.

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. 2024. Sparsellm: Towards global pruning of pre-trained language models. *NeurIPS*.

Aleksandar Botev, Hippolyt Ritter, and David Barber. 2017. Practical gauss-newton optimisation for deep learning. *ICML*.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*.

Vladimír Boža. 2024. Fast and effective weight update for pruned large language models. *TMLR*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *NeurIPS*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *NAACL*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. *ICML*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *ICML*.

Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. 2020. Sparse gpu kernels for deep learning. *SC*.

Scott Gray, Alec Radford, and Diederik P Kingma. 2017. Gpu kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *NeurIPS*.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *JMLR*.

Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Fei Chao, and Rongrong Ji. 2025a. Determining layer-wise sparsity for large language models through a theoretical perspective. *ICML*.

Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Yang Liu, Jing Lin, Yiwu Yao, and Rongrong Ji. 2025b. Dynamic low-rank sparse adaptation for large language models. *ICLR*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Eldar Kurtic and Dan Alistarh. 2022. Gmp*: Well-tuned gradual magnitude pruning can outperform most bert-pruning methods. *arXiv preprint arXiv:2210.06384*.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. 2020. Soft threshold weight reparameterization for learnable sparsity. *ICML*.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *NeurIPS*.

Dongyeop Lee, Kwanhee Lee, Jinseok Chung, and Namhoon Lee. 2025. SAFE: Finding sparse and flat minima to improve pruning. *ICML*.

Kwanhee Lee, Hyeondo Jang, Dongyeop Lee, Dan Alistarh, and Namhoon Lee. 2026. The unseen frontier: Pushing the limits of llm sparsity with surrogate-free admm. *ICLR*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2019. Snip: Single-shot network pruning based on connection sensitivity. *ICLR*.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, and 1 others. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *NeurIPS*.

YuXin Li, Felix Dangel, Derek Tam, and Colin Raffel. 2025. Fishers for free? approximating the fisher information matrix by recycling the squared gradient accumulator. *arXiv preprint arXiv:2507.18807*.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and 1 others. 2024. Not all tokens are what you need for pretraining. *NeurIPS*.

Junjie Liu, Zhe Xu, Runbin Shi, Ray CC Cheung, and Hayden KH So. 2020. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. *arXiv preprint arXiv:2005.06870*.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

Miao Lu, Xiaolong Luo, Tianlong Chen, Wuyang Chen, Dong Liu, and Zhangyang Wang. 2022. Learning pruning-friendly networks via frank-wolfe: One-shot, any-sparsity, and no retraining. In *ICLR*.

Vladimír Macko and Vladimír Boža. 2026. MACKO: Sparse matrix-vector multiplication for low sparsity. *ICML*.

James Martens. 2020. New insights and perspectives on the natural gradient method. *JMLR*.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. 2024. Alps: Improved optimization for highly sparse one-shot pruning for large language models. *NeurIPS*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *ICLR*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *EMNLP*.

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

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

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. *NeurIPS*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only. *NeurIPS Datasets and Benchmarks Track*.

Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. 2021. Ac/dc: Alternating compressed/decompressed training of deep neural networks. *NeurIPS*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Sungbin Shin, Wonpyo Park, Jaeho Lee, and Namhoon Lee. 2024. Rethinking pruning large language models: Benefits and pitfalls of reconstruction error minimization. *EMNLP*.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. 2025. EvoPress: Accurate dynamic model compression via evolutionary search. *ICML*.

Sidak Pal Singh and Dan Alistarh. 2020. Woodfisher: Efficient second-order approximation for neural network compression. *NeurIPS*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. *ICLR*.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. *NeurIPS*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP Workshop BlackboxNLP*.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *NeurIPS*.

Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. 2019. Discovering neural wirings. *NeurIPS*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared LLaMA: Accelerating language model pre-training via structured pruning. *ICLR*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, and 1 others. 2024. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. *ICML*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *ACL*.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Sun Yunyun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. 2024. Dynamic sparse no training: Training-free fine-tuning for sparse LLMs. *ICLR*.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, and 1 others. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2021. Learning n:m fine-grained structured sparse neural networks from scratch. *ICLR*.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv*.

A Additional Discussion

A.1 Further details on prior approaches

Here, we provide a more detailed description of the pruning procedures adopted in prior work. Magnitude prunes weights directly by their magnitude, whereas Wanda (Sun et al., 2024) and SparseGPT (Frantar and Alistarh, 2023) are one-shot methods that determine pruning masks using information from a small calibration dataset. ALPS (Meng et al., 2024), L-ADMM (Boža, 2024), and SAFE (Lee et al., 2025) instead perform iterative layer-wise optimization using calibration data. ELSA (Lee et al., 2026) directly optimizes the original language-modeling objective on additional data in a continued-training setting, but we find that this setting leaves BEST undertrained and therefore adopt a different one for our main results. Budget-matched comparisons with one-shot methods and ELSA are provided in Sections 4.1 and 4.5, respectively, with BEST achieving better performance in both cases.

B Experimental details

B.1 Implementation and reproducibility details

Our experiments are implemented in PyTorch (Paszke et al., 2019), with model architectures and datasets managed through the HuggingFace `transformers` and `datasets` libraries. Our training pipeline is implemented on top of the HuggingFace `Trainer` framework and supports distributed execution via PyTorch FSDP-2 (Zhao et al., 2023), integrated with HuggingFace Accelerate. Unless otherwise noted, all experiments are conducted using a single unified codebase to ensure fair and consistent comparisons; baseline methods are integrated into our codebase and evaluated under the same evaluation framework, with implementations adapted from official releases where available.

Hardware Experiments are executed on NVIDIA A6000, A100, PRO6000, and H200 GPUs. Computational resources are scaled with model size: LLaMA-2-7B/13B and Qwen3-4B/8B are trained using 2–4 NVIDIA A100 GPUs (or PRO6000 GPUs) depending on the experimental configuration, while smaller models such as OPT-125M are trained on a single A6000 GPU. For Qwen2.5-32B, we use 8 PRO6000 GPUs and H200 GPUs.

Algorithm 1 Distributed binary search for global threshold τ.

Require: Saliency scores $\{s_i\}_{i=1}^{N}$ partitioned across G ranks, with I_r denoting the indices held on rank r; target count k; iterations T

$\tau_{\min} \leftarrow 0$
$\tau_{\max} \leftarrow$ ALLREDUCE-MAX$\big(\max_{i \in I_r} s_i\big)$
for $t = 1, \ldots, T$ **do**
  $\tau \leftarrow (\tau_{\min} + \tau_{\max})/2$
  $c_r \leftarrow |\{\, i \in I_r : s_i \leq \tau \,\}|$
  $C \leftarrow$ ALLREDUCE-SUM(c_r)
  if $C \geq k$ **then**
    $\tau_{\max} \leftarrow \tau$
  else
    $\tau_{\min} \leftarrow \tau$
  end if
end for
return $\tau_{\max}$

Efficient global thresholding in a distributed setup Global pruning requires the threshold τ to be the global k-th smallest among the N saliency scores. The naive realization gathers the N sharded scores onto a single device and selects the k-th element there, costing $\mathcal{O}(N)$ memory on that device and $\mathcal{O}(N)$ communication; the gather alone becomes a non-trivial bottleneck at LLM scale. We instead obtain the same threshold without any gather, via the distributed binary search of Algorithm 1: each of the G ranks (i.e., devices) keeps its score shard in place, and only scalar counts are exchanged. Since our saliency criteria are non-negative, we fix $\tau_{\min}{=}0$ and obtain $\tau_{\max}$ via a single ALLREDUCE-MAX; the bracket is then halved $T{=}50$ times, with each iteration requiring one scalar ALLREDUCE-SUM of the per-rank counts c_r. Both the $\mathcal{O}(N)$ memory and $\mathcal{O}(N)$ communication costs of the gather are eliminated, replaced by only $T{+}1$ scalar reductions and a per-rank transient boolean mask of size $\mathcal{O}(N/G)$—only one byte per parameter. This yields the global k-th-value threshold after just T halvings.

B.2 Detailed experimental setup

Calibration data To perform one-shot pruning for Wanda$^+$ and SparseGPT$^+$, we follow the standard one-shot pruning protocol of Frantar and Alistarh (2023). Specifically, we sample 128 calibration sequences from the C4 dataset with a sequence length of 2048, which are used solely for one-shot pruning.

Training data and details. Except for BEST‡, which uses C4 to match ELSA's setting, all continued-training experiments use SlimPajama[1], a cleaned and deduplicated version of RedPajama that combines pretraining corpora from diverse sources including C4, ArXiv, and GitHub.

For both BEST and the one-shot+retrain baselines in our main setting, training data are constructed by randomly sampling non-overlapping token sequences of length 2,048. To ensure a controlled comparison, the total training budget is fixed at $64 \times 4000 \times 2048$ tokens (batch size 64, training steps 4,000, and sequence length 2,048). For ELSA and BEST‡, we instead use ELSA's setting of batch size 8, 4,096 training steps, and sequence length 2,048, corresponding to approximately 67M training tokens.

Model parameters and optimizer states are kept in full precision, while forward/backward passes use bf16 automatic mixed precision for efficiency.

Progressive sparsification For BEST, given a total training budget of T_{total} steps, we set the pruning phase to $T_{\text{prune}} = 0.5\,T_{\text{total}}$, start from $S_{\text{init}} = 0$, and perform $K = 50$ mask updates uniformly within this phase, so that masks are updated every $\Delta t = T_{\text{prune}}/K$ steps (40 steps when $T_{\text{total}} = 4000$). At each mask update, the cubic schedule in Equation (9) determines the new sparsity level, the global mask is recomputed from the current weights, and gradient updates resume on the remaining parameters. After T_{prune}, the mask is frozen at the final target sparsity and the remaining steps are used to train the fixed sparse model. A full list of hyperparameter configurations is provided in Tables 6 to 8.

Evaluation We evaluate language-modeling perplexity on the held-out validation splits of Wikitext-2 (`Salesforce/wikitext-2-raw-v1`) (Merity et al., 2017) and C4 (`allenai/c4`, English subset) (Raffel et al., 2020), following the standard perplexity protocol established by Frantar and Alistarh (2023) and adopted in subsequent LLM pruning work (Sun et al., 2024; Meng et al., 2024): each model uses its own pretrained tokenizer (HuggingFace `AutoTokenizer`) with a fixed context length of $L = 2048$, the validation text is tokenized once and split into contiguous, non-overlapping chunks of length L (no sliding-window stride), and each chunk is fed to the model as a single sequence to compute token-level cross-entropy with the standard shift-by-one labelling. We evaluate the entire Wikitext-2 validation split and the first $256 \times L$ tokens of the C4 validation split, applying the same protocol to both dense and sparse models. For zero-shot accuracy we use `lm-eval-harness` on seven standard benchmarks — ARC-Easy and ARC-Challenge (ARC-E/C) (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), OpenBookQA (OBQA) (Mihaylov et al., 2018), RTE (Wang et al., 2018), and Winogrande (Sakaguchi et al., 2021) — and report the mean accuracy across the seven tasks.

[1] DKYoon/SlimPajama-6B

Table 6: Global hyperparameters of BEST and one-shot+retrain strategy shared across all models.

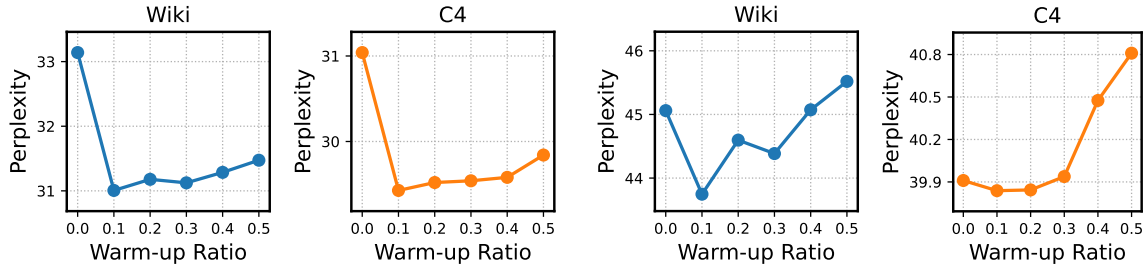
Hyperparameter	BEST	one-shot+retrain
LR schedule	Linear decay	Linear decay
Pruning interval k	40	–
Adam (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$
Batch size	64	64
Training steps	4000	4000
Warm-up steps	400	400

Table 7: Learning rate (η) configurations of BEST across sparsity levels.

Sparsity	OPT-125M	LLaMA-2-7B	LLaMA-2-13B	Qwen3-4B-Base	Qwen3-8B-Base
70%	5e−4	5e−5	5e−5	5e−5	1e−4
90%	1e−3	1e−4	1e−4	5e−4	5e−4
95%	–	5e−4	1e−4	5e−4	5e−4
99%	–	5e−4	1e−4	5e−4	1e−3

Table 8: Learning rate (η) configurations of one-shot+retrain strategy across sparsity levels.

Sparsity	OPT-125M	LLaMA-2-7B	LLaMA-2-13B	Qwen3-4B-Base	Qwen3-8B-Base
70%	5e−4	1e−4	5e−5	1e−4	1e−4
90%	1e−3	1e−4	1e−4	5e−4	5e−4
95%	–	5e−4	1e−4	5e−4	5e−4
99%	–	5e−4	1e−4	5e−4	1e−3



(a) Sparsity: 70%

(b) Sparsity: 90%

Figure 6: Effect of warm-up ratio (of total 4000 steps) on perplexity for OPT-125M across sparsity levels; a short warm-up (∼10%) consistently achieves stable performance.

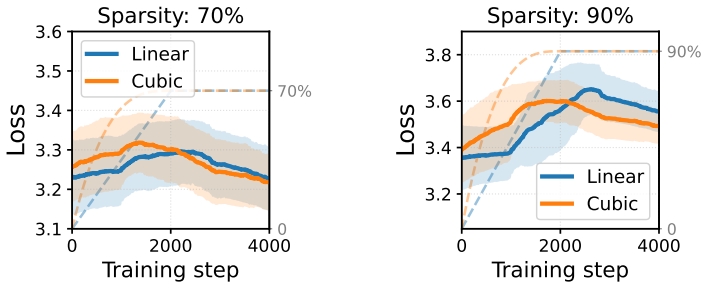


Figure 7: Training-loss curves on OPT-125M under linear and cubic sparsity schedules (solid: train loss; dashed: sparsity, right axis). Cubic reaches lower final loss, with the gap widening at higher sparsity.

Table 9: Final perplexity on Wikitext-2 and C4 for OPT-125M under cubic vs linear sparsity schedule.

Schedule	Sparsity			
	70%		90%	
	Wiki	C4	Wiki	C4
Cubic	**30.77**	**29.95**	**43.84**	**39.15**
Linear	31.64	30.40	47.51	42.70

C Ablation study

C.1 Warm-up steps

Here, we describe how the number of warm-up steps was chosen. Using OPT-125M with $T_{\text{total}} = 4000$ training steps, we sweep the warm-up ratio $r \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ ($T_{\text{warmup}} = r \cdot T_{\text{total}} \in \{400, 800, 1200, 1600, 2000\}$) and report Wikitext-2 and C4 perplexity in Figure 6. Across all sparsity levels, any warm-up is beneficial; longer warm-up, however, degrades performance because the sparsity schedule has already reached high sparsity while the learning rate is still ramping up, leaving insufficient capacity to recover from the aggressive pruning. This trend becomes more pronounced at higher sparsity. We therefore use a short warm-up of $r = 0.1$ ($T_{\text{warmup}} = 400$ steps) throughout the paper.

C.2 Cubic vs linear sparsity schedules

Cubic sparsity schedules have been reported to be effective in prior pruning works.

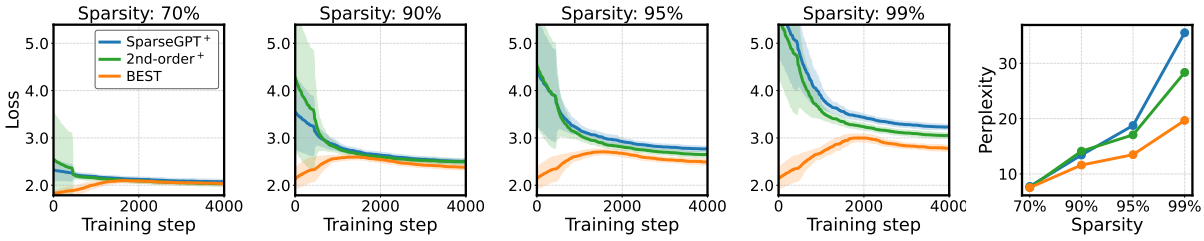
To check whether this still holds in our setting, we compare the cubic schedule adopted in Section 3.2 (Equation (9)) against the linear alternative defined as

$$S_t = S_{\text{init}} + (S_{\text{final}} - S_{\text{init}}) \frac{t - t_0}{T_{\text{prune}}}. \tag{11}$$

On OPT-125M, with training-loss curves in Figure 7 and final perplexity in Table 9, the cubic schedule introduces sparsity more aggressively in the early phase, causing a larger initial loss increase, but ends up at a lower final loss than the linear schedule, with the advantage most visible at 90% sparsity. The same trend holds in perplexity: cubic wins at every sparsity level and the gap widens at higher sparsity. We therefore use the cubic schedule throughout the paper.

C.3 Additional results on the sparsity schedule

Section 4.4 demonstrates the benefit of progressive sparsification over one-shot pruning at 90% sparsity. To further substantiate this finding, Figure 8 extends the comparison to all evaluated sparsity levels on LLaMA-2-7B and reports the resulting WikiText-2 perplexities. For reference,



(a) One-shot vs Progressive (b) Perplexity

Figure 8: Training loss curves on LLaMA-2-7B at each sparsity level, with the resulting WikiText-2 perplexity on the right. The comparison between BEST and 2nd-order$^+$ isolates the gradual schedule. The advantage of the gradual schedule grows with sparsity, with BEST pulling further ahead at each higher level.

Table 10: Effect of reusing Adam optimizer state for diagonal Fisher approximation. $|D|$ denotes the calibration dataset size.

Recompute	$\|D\|$	Sparsity			
		70%		90%	
		Wiki	C4	Wiki	C4
✗	–	**30.77**	29.95	43.84	**39.15**
✓	128	31.32	29.61	44.51	39.76
	512	30.96	**29.54**	**43.79**	39.39
	2048	31.17	29.62	43.96	39.24

Table 11: Controlled comparison of global thresholding (BEST) and ATP layer-wise allocation on OPT-125M, with all other components fixed.

	Sparsity			
	70%		90%	
Allocation	Wiki	C4	Wiki	C4
ATP	31.75	30.54	45.28	40.78
BEST	**30.77**	**29.16**	**43.84**	**39.15**

we also include SparseGPT$^+$ as a competitive retrained one-shot variant. At every evaluated sparsity level, BEST yields lower perplexity than both 2nd-order$^+$ and SparseGPT$^+$, with the margins widening as sparsity increases. Specifically, from 70% to 99% sparsity, the gap grows from 0.01 to 8.68 against 2nd-order$^+$ and from 0.22 to 15.86 against SparseGPT$^+$. These results indicate that progressive sparsification mitigates the limitations of imposing sparsity in a single step and becomes increasingly important at higher sparsity.

C.4 Diagonal Fisher approximation via Adam second moment

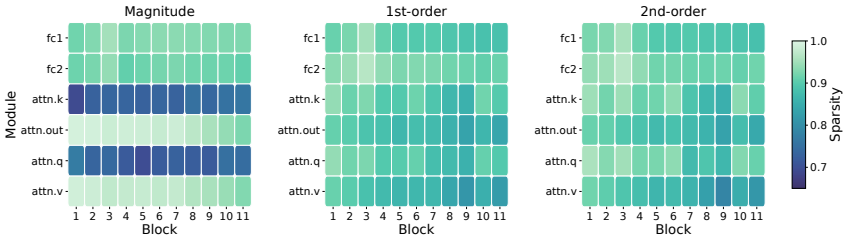
We conduct ablation studies on OPT-125M to validate the effectiveness of reusing the Adam optimizer state for second-order saliency. To compute the saliency score for each parameter w_i, we contrast two strategies: directly reusing the accumulated second-moment element $v_{t,i}$ versus recomputing the empirical Fisher $\mathbb{E}[g_i^2]$ using gradients collected from 128, 512, and 2048 data samples.

Table 10 presents the perplexity on Wikitext-2 and C4 datasets across sparsity levels of 70% and 90%. The results demonstrate that recomputing the second-moment estimate yields no consistent performance improvement over simple state reuse. Even with 2048 calibration sequences, the perplexity differences remain marginal across all sparsity regimes. Considering that recomputing $\mathbb{E}[g_i^2]$ introduces non-trivial computational overhead due to additional forward-backward passes, we conclude that reusing the existing Adam state provides a sufficiently accurate and efficient proxy for curvature in gradual pruning.

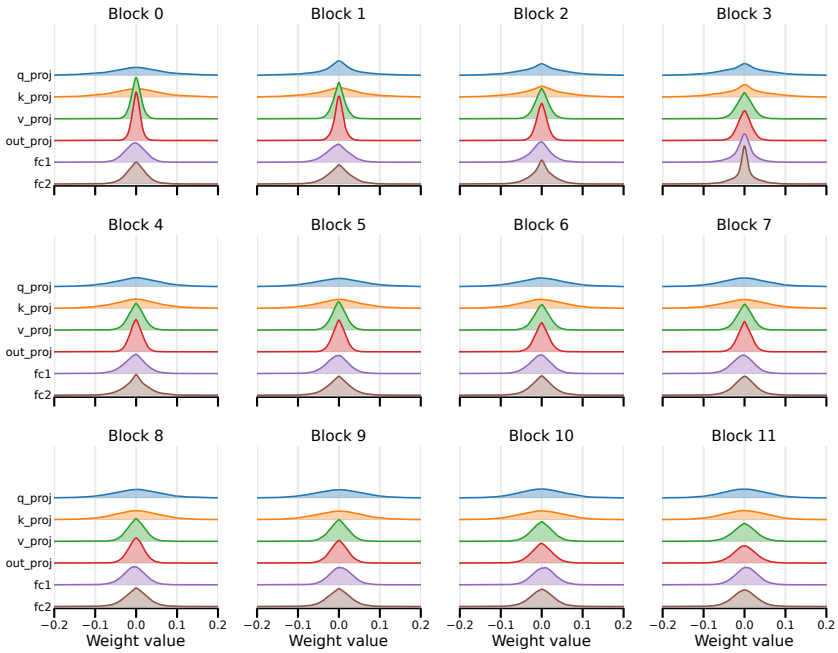
C.5 Alternative layer-wise sparsity allocation

Section 4.4 compares BEST's global thresholding with uniform layer-wise allocation. To further validate the effectiveness of our allocation strategy, we replace only the allocation rule with ATP (Huang et al., 2025a) on OPT-125M, retaining the saliency criterion, gradual schedule, optimization, and training budget of BEST. ATP assigns per-block sparsity targets along an arithmetic progression; at each target sparsity, we tune its common difference over a five-point grid spanning the feasible range.

Table 11 shows that our global thresholding achieves lower perplexity than ATP at both sparsity levels on both corpora. These results demonstrate that our approach provides an effective sparsity allocation across layers in this setting.



(a) Sparsity allocation heatmaps for OPT-125M pruned to 90% sparsity.



(b) Per-block weight distribution of OPT-125M (KDE), drawn as a ridgeline with one row per Linear module type. Each panel corresponds to one Transformer block, and all panels share a single density scale so they are directly comparable. Module-level heterogeneity is most pronounced in early blocks and gradually flattens in later blocks.

Figure 9: Why magnitude pruning fails under global pruning. (a) Module-level sparsity allocation under magnitude vs. first-/second-order saliencies on OPT-125M at 90% sparsity. (b) Block-wise weight distributions explain the allocation pattern.

Table 12: WikiText-2 perplexity of LLaMA-2-7B across sparsity levels. BEST‡ denotes BEST under ELSA's training setting.

Method	Sparsity			
	70%	90%	95%	99%
ELSA	13.20	26.97	38.91	55.94
BEST‡	<u>12.58</u>	<u>18.84</u>	<u>24.34</u>	<u>41.05</u>
BEST	**7.52**	**11.60**	**13.48**	**19.67**

D Additional results

D.1 Extended comparison with ELSA

We report the complete comparison with ELSA from 70% to 99% sparsity, including a matched variant of our method. Specifically, BEST‡ follows ELSA's exact training setting, using C4 and a 67M-token budget. As shown in Table 12, BEST‡ outperforms ELSA at every sparsity level. Notably, at 95% sparsity, BEST‡ even achieves lower perplexity than ELSA at 90% sparsity (24.34 vs. 26.97).

Scaling the training budget to 524M tokens and using SlimPajama, a cleaned and deduplicated pretraining corpus (Soboleva et al., 2023), further improves performance in the extreme-sparsity regime. Under this setting, BEST achieves perplexities of 13.48 at 95% sparsity and 19.67 at 99% sparsity.

D.2 Why magnitude pruning fails: sparsity allocation and module-level weight distributions

In addition to the 70% heatmap shown in Section 4.4, we report the per-module sparsity allocation of OPT-125M at 90% (Figure 9a). The same trend persists: magnitude pruning over-allocates sparsity to specific modules (v_{proj} and out_{proj}) while

Table 13: Wikitext-2 perplexity of models with comparable effective parameter counts.

Model	Sparsity	Perplexity
OPT-125M	0%	27.60
OPT-1.3B	90.5%	**25.23**
Qwen2.5-0.5B	0%	12.87
Qwen2.5-1.5B	67%	**12.03**

Table 14: Open-ended generation and reasoning accuracy (%) for Qwen3-8B-Base pruned with BEST. Reasoning degrades far faster than perplexity or multiple-choice accuracy.

Benchmark	Dense	Sparsity			
		70%	90%	95%	99%
GSM8K (8-shot, CoT)	87.64	28.05	2.35	2.88	2.12
NQ-Open (5-shot)	25.9	14.4	5.6	3.6	1.0

Table 15: Representative GSM8K generations from Qwen3-8B-Base pruned with BEST. The 70% model stays fluent but skips the overtime hours; the 95% model collapses into repetition.

Eliza earns \$10/hr for the first 40 hours and 1.2× that rate for overtime. She worked 45 hours. What are her earnings? (answer: \$460)	
Dense	"$40 \times 10 = 400$. Overtime $1.2 \times 10 = \$12$/hr, $5 \times 12 = 60$. Total $400 + 60 = 460$."
70%	"$40 \times 10 = 400$. Overtime $1.2 \times 10 = \$12$. Total $400 + 12 = 412$."
95%	"Eliza's rate per hour for the first 40 hours… If Eliza worked for 45 hours… Eliza's rate…"

leaving others (q_{proj} and k_{proj}) relatively dense. This excessive allocation to a few modules is what causes magnitude pruning to fail under global pruning, as observed in Table 4 of Section 4.4.

To explain the underlying cause, we analyze the weight distribution of every Transformer block in OPT-125M. Figure 9b shows the empirical density of weight values per block, with each panel showing one row per Linear module type (q_{proj}, k_{proj}, v_{proj}, out_{proj}, fc_1, fc_2). Each block exhibits clear module-level heterogeneity, which is most severe in the early blocks and gradually flattens toward later blocks. As a result, as shown in Figure 9a, magnitude pruning assigns disproportionately high sparsity to the early blocks of v_{proj} and out_{proj}, leading to layer collapse. This indicates that the magnitude saliency score does not faithfully capture the actual loss contribution of each parameter.

D.3 Small dense vs. large sparse

A practical question is whether BEST can convert an existing large pretrained model into compact models of different target sizes, rather than training a new small dense model from scratch for each size, which typically requires orders of magnitude more compute. We therefore compare small dense models with larger models pruned by BEST at comparable effective parameter counts, using this setting as a quality-oriented proxy for compact model construction. As shown in Table 13, the large sparse models achieve lower Wikitext-2 perplexity than their dense counterparts in both pairs. This trend holds across the OPT and Qwen2.5 families, suggesting that, when a strong pretrained model is already available, sparsification can provide a flexible and compute-efficient route to competitive compact models. By varying the target sparsity, BEST can produce models at different effective sizes without requiring a separate pretraining run for each target scale. We note that this comparison concerns model quality at a comparable effective parameter count; small dense models may still be preferable for deployment on hardware without efficient sparse computation support.

D.4 Open-ended generation and reasoning

To strengthen our assessment of the practical capabilities of sparse models, we extend the evaluation beyond perplexity and multiple-choice accuracy to open-ended generation and reasoning. Specifically, we evaluate Qwen3-8B-Base pruned with BEST on GSM8K (chain-of-thought, 8-shot) and NQ-Open (5-shot) using `lm-eval-harness`. These evaluations use the pruned base models with-

out task-specific data curation or adaptation.

Table 14 shows that open-ended generation and reasoning remain challenging as sparsity increases. Mathematical reasoning degrades fastest: GSM8K falls from 87.64 to 28.05 already at 70% sparsity and to near-zero accuracy beyond it, while factual recall on NQ-Open decays more gradually. Table 15 illustrates this failure mode: at 70%, the model still produces well-formed arithmetic but omits a step of the problem, whereas at 95%, it stops solving and repeats fragments of the question. These findings show that preserving perplexity and multiple-choice accuracy under extreme sparsity does not ensure robust free-form generation or reasoning. Recovering these capabilities through sparsity-aware adaptation or targeted recovery training is therefore an important direction for future work.

D.5 Numerical results of zero-shot downstream task accuracy

We further provide the full per-task zero-shot results for LLaMA-2-7B/13B and Qwen3-4B/8B-Base, as discussed in Section 4.

Table 16: Zero-shot accuracy (%) of LLaMA-2-7B across multiple tasks under different sparsity regimes. Numbers for L-ADMM, ALPS, SAFE, SparseLLM, and ELSA are taken from Lee et al. (2026).

Sparsity	Method	ARC-C	ARC-E	BoolQ	HellaSwag	OBQA	RTE	Winogrande	Avg
0%	Dense	43.35	76.26	77.68	57.14	31.40	62.82	69.06	59.67
70%	Magnitude	22.70	27.82	37.95	25.92	17.00	53.07	49.09	33.36
	Wanda	18.52	30.93	47.65	28.11	13.20	52.71	49.01	34.30
	SparseGPT	21.84	44.61	65.02	34.53	16.80	52.71	59.12	42.09
	L-ADMM	23.81	50.63	63.21	36.57	20.40	54.15	60.77	44.22
	ALPS	25.51	52.78	63.46	37.54	20.8	53.43	61.72	45.03
	SAFE	24.23	45.62	43.76	34.74	18.40	52.71	53.12	38.94
	SparseLLM	20.90	40.32	61.87	32.74	16.0	54.51	57.46	40.54
	ELSA	27.13	55.81	63.61	43.16	22.40	52.71	58.64	46.21
	Magnitude$^+$	32.08	66.41	68.01	48.73	**26.80**	**59.93**	63.30	**52.18**
	Wanda$^+$	30.55	63.68	65.11	46.45	22.60	57.04	60.06	49.36
	SparseGPT$^+$	32.94	64.23	**69.24**	48.47	25.40	53.07	**63.69**	51.00
	BEST	**34.30**	**68.77**	67.98	**49.67**	25.60	55.57	62.51	52.06
90%	Magnitude	**22.53**	25.88	39.17	25.50	16.00	47.29	49.96	32.33
	Wanda	20.90	26.05	37.83	25.76	15.20	52.71	49.88	32.62
	SparseGPT	21.25	25.84	38.04	25.59	12.60	53.43	49.88	32.38
	L-ADMM	19.97	26.14	37.83	26.46	13.60	51.62	47.51	31.88
	ALPS	19.45	26.89	37.8	26.81	12.8	**53.79**	46.65	32.03
	SAFE	21.84	26.52	37.83	25.91	15.80	52.71	47.83	32.63
	SparseLLM	20.56	25.72	37.83	25.94	13.8	52.71	46.96	31.93
	ELSA	18.52	41.33	57.25	31.54	16.60	52.71	51.70	38.52
	Magnitude$^+$	20.48	48.27	58.07	33.82	19.20	52.71	51.54	40.58
	Wanda$^+$	18.94	42.63	58.78	30.49	15.40	53.07	49.09	38.34
	SparseGPT$^+$	19.03	45.12	59.69	31.86	16.40	52.71	52.17	39.57
	BEST	20.48	**49.49**	**60.95**	**36.51**	**22.4**	52.71	**53.83**	**42.34**
95%	Magnitude	**22.01**	25.97	52.05	25.64	14.60	45.13	49.80	33.60
	Wanda	21.59	25.72	37.83	25.78	13.60	52.71	49.57	32.40
	SparseGPT	21.42	26.89	37.83	25.64	15.80	**53.07**	51.30	33.14
	Magnitude$^+$	18.09	39.31	60.64	28.15	15.20	52.35	51.38	37.87
	Wanda$^+$	17.58	36.9	61.65	27.94	15.00	52.71	**52.25**	37.72
	SparseGPT$^+$	17.92	39.35	**61.83**	28.45	16.60	52.35	49.09	37.94
	BEST	19.63	**46.63**	57.83	**32.72**	**17.80**	52.71	50.67	**39.71**
99%	Magnitude	21.67	26.52	51.71	25.36	15.20	48.74	49.57	34.11
	Wanda	**23.21**	25.17	**62.17**	25.62	**16.60**	45.85	47.59	35.17
	SparseGPT	22.18	25.34	41.90	25.65	14.40	46.21	50.12	32.26
	Magnitude$^+$	16.98	34.39	61.96	26.95	11.60	51.26	**52.25**	36.48
	Wanda$^+$	18.34	34.47	61.53	26.91	14.60	53.07	51.54	37.21
	SparseGPT$^+$	17.58	33.67	60.09	27.27	13.20	**53.79**	50.51	36.59
	BEST	17.58	**40.19**	61.41	**28.43**	13.60	53.07	50.59	**37.84**

Table 17: Zero-shot accuracy (%) of LLaMA-2-13B across multiple tasks under different sparsity regimes.

Sparsity	Method	ARC-C	ARC-E	BoolQ	HellaSwag	OBQA	RTE	Winogrande	Avg
0%	Dense	48.46	79.38	80.55	60.04	35.20	65.34	72.14	63.02
70%	Magnitude	20.65	31.27	38.65	27.53	14.60	52.71	49.33	33.54
	Wanda	18.94	40.15	62.17	29.96	14.60	52.71	52.72	38.75
	SparseGPT	24.74	52.82	68.10	36.73	19.80	52.71	61.25	45.16
	Magnitude$^+$	36.26	70.45	71.35	52.85	28.20	61.37	66.14	55.23
	Wanda$^+$	34.39	68.43	71.04	51.75	27.20	57.76	63.06	53.38
	SparseGPT$^+$	36.60	69.99	**72.78**	53.13	28.60	59.21	**66.30**	55.23
	BEST	**37.63**	**72.69**	71.59	**54.50**	**31.60**	**62.45**	65.19	**56.52**
90%	Magnitude	21.67	24.83	44.19	25.71	15.00	45.85	52.01	32.75
	Wanda	22.10	25.55	37.83	25.80	13.40	52.71	52.80	32.88
	SparseGPT	22.18	25.88	56.82	25.92	13.00	52.71	51.07	35.37
	Magnitude$^+$	22.78	52.19	58.29	36.98	21.60	52.71	53.43	42.57
	Wanda$^+$	17.75	43.14	60.06	31.48	14.60	52.71	51.62	38.76
	SparseGPT$^+$	21.33	48.78	58.38	34.49	17.00	53.07	51.54	40.65
	BEST	**25.43**	**56.73**	**62.14**	**40.78**	**23.00**	52.71	**57.06**	**45.41**
95%	Magnitude	**24.06**	25.63	58.23	25.48	15.40	47.29	48.62	34.96
	Wanda	22.27	25.80	37.83	25.79	13.00	52.71	51.14	32.65
	SparseGPT	21.76	25.72	37.83	25.76	14.60	52.71	50.12	32.64
	Magnitude$^+$	17.75	33.75	61.96	26.77	12.80	52.71	50.67	36.63
	Wanda$^+$	17.06	39.65	61.74	28.19	16.00	52.71	50.91	38.04
	SparseGPT$^+$	17.83	40.32	61.47	29.35	13.80	**53.07**	52.72	38.37
	BEST	20.56	**48.70**	**61.99**	**34.46**	**18.60**	52.71	**53.51**	**41.50**
99%	Magnitude	**22.78**	25.21	52.72	25.61	14.80	51.99	49.96	34.72
	Wanda	22.35	25.38	39.30	25.75	13.60	47.29	50.12	31.97
	SparseGPT	21.67	25.63	43.27	25.65	14.00	53.79	48.93	33.28
	Magnitude$^+$	17.75	33.67	60.43	26.87	13.40	52.71	49.49	36.33
	Wanda$^+$	17.92	34.68	60.95	27.03	13.00	52.71	50.59	36.70
	SparseGPT$^+$	17.41	34.64	**62.23**	27.33	12.60	**54.15**	**52.17**	**37.22**
	BEST	18.09	**36.53**	59.57	**27.95**	15.20	53.07	50.12	**37.22**

Table 18: Zero-shot accuracy (%) of Qwen3-4B-Base across multiple tasks under different sparsity regimes.

Sparsity	Method	ARC-C	ARC-E	BoolQ	HellaSwag	OBQA	RTE	Winogrande	Avg
0%	Dense	48.38	79.17	83.12	54.50	31.60	76.90	70.56	63.46
70%	Magnitude	22.18	25.29	49.27	25.69	16.80	53.43	50.59	34.75
	Wanda	19.03	43.22	62.14	29.31	12.00	52.71	50.67	38.44
	SparseGPT	25.68	55.18	62.26	35.71	17.80	54.51	57.14	44.04
	Magnitude$^+$	35.58	69.57	70.40	45.48	26.40	**59.93**	61.17	52.65
	Wanda$^+$	34.13	69.57	67.19	44.85	26.20	57.76	58.96	51.24
	SparseGPT$^+$	37.80	69.99	70.80	46.17	26.80	58.84	**62.12**	53.22
	BEST	**39.16**	**73.44**	**73.21**	**47.73**	**27.80**	57.76	61.25	**54.34**
90%	Magnitude	19.88	24.49	60.18	25.95	14.80	46.93	52.57	34.97
	Wanda	20.56	24.62	37.86	25.59	14.20	51.99	49.09	31.99
	SparseGPT	20.05	26.89	38.23	26.21	11.20	**52.71**	48.78	32.01
	Magnitude$^+$	17.75	43.27	60.06	28.99	16.40	50.90	50.43	38.26
	Wanda$^+$	20.31	44.40	58.32	29.59	13.40	52.35	50.75	38.45
	SparseGPT$^+$	20.56	46.42	**61.68**	31.27	16.80	52.35	50.75	39.83
	BEST	**22.78**	**53.32**	60.09	**36.42**	**20.80**	**52.71**	**54.78**	**42.99**
95%	Magnitude	20.73	24.41	38.53	25.57	15.60	48.38	49.41	31.80
	Wanda	**21.42**	26.47	37.83	25.76	14.00	52.71	48.07	32.32
	SparseGPT	20.56	26.09	38.07	25.95	**19.40**	52.71	49.88	33.24
	Magnitude$^+$	18.60	37.21	60.76	27.18	13.00	**53.43**	51.38	37.37
	Wanda$^+$	18.17	39.90	59.02	28.08	14.00	**53.43**	50.04	37.52
	SparseGPT$^+$	17.49	39.81	**62.17**	27.74	13.20	52.71	50.20	37.62
	BEST	20.65	**49.24**	61.10	**32.73**	17.40	53.07	**51.62**	**40.83**
99%	Magnitude	21.84	25.25	38.56	25.56	16.00	53.07	50.51	32.97
	Wanda	**24.83**	24.62	38.23	25.15	**16.20**	52.71	50.75	33.20
	SparseGPT	21.84	24.37	37.80	25.79	13.20	52.35	51.38	32.39
	Magnitude$^+$	18.34	33.67	45.57	26.48	13.60	51.62	51.14	34.35
	Wanda$^+$	17.15	35.10	54.46	26.70	11.80	**54.87**	50.43	35.79
	SparseGPT$^+$	18.52	34.34	61.01	26.44	13.40	53.43	50.12	36.75
	BEST	18.52	**40.11**	**61.80**	**28.50**	12.80	52.35	**52.09**	**38.02**

Table 19: Zero-shot accuracy (%) of Qwen3-8B-Base across multiple tasks under different sparsity regimes.

Sparsity	Method	ARC-C	ARC-E	BoolQ	HellaSwag	OBQA	RTE	Winogrande	Avg
0%	Dense	52.56	81.69	83.09	58.80	32.40	74.73	72.38	65.09
70%	Magnitude	21.84	27.48	44.31	25.84	17.20	53.43	49.96	34.30
	Wanda	20.31	48.99	62.17	30.47	16.00	52.71	53.67	40.62
	SparseGPT	29.78	62.04	65.66	38.75	21.80	58.84	60.69	48.22
	Magnitude$^+$	42.06	75.80	72.17	49.63	28.20	63.90	62.83	56.37
	Wanda$^+$	42.49	74.49	71.65	49.05	27.20	62.09	63.30	55.75
	SparseGPT$^+$	42.24	74.79	73.03	50.35	**30.40**	**67.15**	64.01	57.42
	BEST	**44.80**	**76.35**	**73.36**	**51.17**	27.80	64.62	**66.69**	**57.83**
90%	Magnitude	22.10	25.72	43.03	26.00	15.00	51.26	48.78	33.13
	Wanda	20.14	24.79	37.77	26.15	12.60	52.35	49.25	31.86
	SparseGPT	20.22	27.15	44.40	26.57	12.60	51.99	50.12	33.29
	Magnitude$^+$	22.35	47.69	60.43	31.71	18.60	54.15	51.62	40.93
	Wanda$^+$	20.39	47.35	60.89	32.18	18.40	52.35	50.59	40.31
	SparseGPT$^+$	21.50	49.83	61.28	34.06	19.00	**55.23**	53.28	42.03
	BEST	**25.00**	**56.48**	**61.71**	**38.87**	**21.60**	53.43	**56.43**	**44.79**
95%	Magnitude	21.59	25.46	40.55	25.72	16.80	**52.71**	48.30	33.02
	Wanda	**23.46**	25.93	54.56	25.72	16.60	52.35	49.33	35.42
	SparseGPT	20.22	28.03	37.83	25.77	13.20	**52.71**	49.64	32.48
	Magnitude$^+$	16.98	37.96	**62.17**	27.54	13.00	52.35	50.43	37.21
	Wanda$^+$	17.75	40.95	61.71	28.49	13.80	**52.71**	51.14	38.08
	SparseGPT$^+$	17.75	42.05	**62.17**	29.15	13.40	52.35	50.91	38.25
	BEST	23.04	**50.76**	59.76	**34.91**	**18.60**	**52.71**	**51.30**	**41.58**
99%	Magnitude	23.29	25.51	37.83	25.52	**17.00**	**52.71**	50.75	33.23
	Wanda	**23.38**	25.38	54.95	25.63	15.00	51.99	49.96	35.18
	SparseGPT	22.35	25.46	38.10	25.45	15.80	47.29	49.17	31.95
	Magnitude$^+$	16.47	36.62	59.48	27.15	12.00	50.90	49.96	36.08
	Wanda$^+$	17.58	37.12	50.00	27.11	14.00	51.26	50.91	35.43
	SparseGPT$^+$	16.38	38.51	**62.02**	27.61	12.40	51.62	50.83	37.05
	BEST	19.62	**42.21**	61.87	**29.49**	15.60	**52.71**	**51.46**	**38.99**