Block-OBS-GS: Exact Per-Block Joint Brain Surgery with Gauss–Seidel Refinement for LLM Pruning

Yuwen Huang
Thrust of Data Science and Analytics
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
yuwen.huang@ieee.org

Xiang Pan*
Division of Industrial Data Science
School of Data Science
Lingnan University
Hong Kong, China
xxiangpan@ln.edu.hk

Abstract

We propose Block-OBS-GS, a retraining-free pruner for large language models (LLMs) that solves the per-block joint Optimal Brain Surgeon (OBS) reconstruction exactly and then refines it with Gauss–Seidel (GS) sweeps. SparseGPT and unstructured Thanos reconstruct the retained weights in increasing column order, one column or column block (a group of consecutive columns) at a time, and never change earlier retained weights again. Viewing reconstruction as constrained maximum a posteriori (MAP) estimation explains why this forward compensation, even with an exact block solve, need not minimize the layer-output error under the completed mask. Block-OBS-GS therefore adds GS sweeps over the retained weights of all blocks, with the mask fixed, after a forward pass that solves each block jointly within each row. We prove three guarantees: the block solve is the unique minimizer of its per-block problem; with the mask fixed and the damped Hessian positive definite, the distance to the unique fixed-mask minimizer contracts geometrically with each GS sweep; and, for fixed block width and number of GS sweeps, the method stays in SparseGPT's complexity class. On Llama-3 models from 3B to 70B parameters, Block-OBS-GS gives lower WikiText-2 perplexity than SparseGPT at every tested sparsity, and the gain grows with the sparsity. On Llama-3.1-8B, its perplexity is 31.69 versus SparseGPT's 36.86 at 75% sparsity, with pruning times of 952.8 versus 474.7 seconds, and 47% lower at 87.5% sparsity (137.19 versus 256.71). Its seven-task zero-shot average and five-shot Massive Multitask Language Understanding (MMLU) accuracies stay within 0.8 percentage points of SparseGPT's. In summary, Block-OBS-GS lowers perplexity at a measured additional pruning cost.

1 Introduction

Retraining-free pruning decides which weights of a large language model to set to zero and how to limit the resulting quality loss without retraining. The zeros produce sparsity that suitable storage formats and execution kernels can exploit, reducing deployment storage and computation [Frantar and Alistarh, 2023, Sun et al., 2024]. Retraining a large model to recover lost quality is costly; retraining-free methods therefore guide pruning with a small calibration dataset [Frantar and Alistarh, 2023]. The standard tool for limiting the quality loss in this setting is layer reconstruction.

Layer reconstruction adjusts the retained weights of each linear layer to reduce the output change caused by the zeros. Following SparseGPT [Frantar and Alistarh, 2023] and Thanos [Ilin and Richtárik, 2025], we measure this change by the sum of squared Euclidean distances between the

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

original and pruned layer outputs on the same calibration inputs; this layer-wise surrogate avoids end-to-end retraining but does not directly minimize prediction loss. Jointly optimizing the zero locations and the retained values is NP-hard in general [Natarajan, 1995, Frantar and Alistarh, 2023]. With the zero locations (the pruning mask) fixed, however, the error is a convex quadratic in the retained values. Practical methods therefore separate the two tasks: they select zeros and reconstruct under a restriction, which the next paragraph identifies.

The restriction shared by SparseGPT and Thanos leaves reducible error under the completed mask. These pruners select zeros with second-order scores derived from Optimal Brain Surgeon (OBS) [Hassibi and Stork, 1992, Frantar and Alistarh, 2023, Ilin and Richtárik, 2025] and compensate only through later columns, never changing earlier retained weights again; we call this *forward compensation*. Later zeros can shift the error-minimizing values of earlier weights; hence even exact blockwise compensation can miss the fixed-mask optimum (Section 2). Our research question is therefore how to reduce this remaining error without changing the selected zeros or increasing the asymptotic computational cost.

We answer it with Block-OBS-GS, which follows forward block reconstruction by refinement under the completed pruning mask. In the first stage, we process one column block at a time: we select weights to zero, jointly reconstruct the retained weights in each row of that block, and propagate compensation to later columns. In the second stage, we fix the selected zeros and apply Gauss–Seidel (GS) updates [Saad, 2003] to retained weights throughout the layer, including those in earlier blocks. In each *sweep*, we update every retained weight in column order using the latest values of the other weights. The first stage thus produces the mask, and the second removes the error that forward compensation leaves under it.

Our analysis shows that this refinement reaches the fixed-mask optimum without raising the complexity order. We prove that repeated GS sweeps converge geometrically to the unique minimizer over the retained weights when the damped error has a positive-definite Hessian; the guarantee covers this objective, not mask selection, perplexity, or finitely many sweeps. For fixed layer dimensions, block width, and sweep count, Block-OBS-GS stays in SparseGPT's asymptotic complexity class, although refinement adds runtime. Viewing reconstruction as constrained maximum a posteriori (MAP) estimation of a Gaussian model [Loeliger et al., 2007] explains the gap: forward pruners make sequential hard decisions, and GS is the classical coordinate-wise MAP refinement [Besag, 1986]. These results predict lower reconstruction error at a bounded additional cost, which our experiments measure.

Our experiments confirm lower perplexity (PPL) at additional pruning time. With the mask fixed on Llama-3.2-3B at 75% sparsity, one GS sweep lowers WikiText-2 PPL from 63.1194 after first-stage reconstruction to 52.6043. On Llama-3.1-8B at 75% sparsity, Block-OBS-GS gives PPL 31.6882 versus 36.8552 for SparseGPT, with pruning times of 952.8 and 474.7 seconds, respectively. On Llama-3.3-70B-Instruct at 50% sparsity, the two methods give similar PPL and pruning time (Section 5.4). In summary, we contribute (i) joint block reconstruction with fixed-mask refinement; (ii) optimality, convergence, and complexity analysis with a constrained-MAP interpretation; and (iii) quality–cost comparisons and fixed-mask reconstruction tests.

2 Reconstruction objective and prior approximation

In this section, we show that forward pruning can miss the optimum of the fixed-mask reconstruction problem. We define the problem and its solution, describe the forward approximation, give an example of the gap, and interpret both stages as MAP estimation. We use the following notation throughout.

- For any positive integer n, $[n] \triangleq \{1, \dots, n\}$. Index sets are written in calligraphic letters, and vectors and matrices in bold.
- For $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, the scalar $A_{s,t}$ denotes its (s,t) entry. For $\mathcal{S} \subseteq [m]$ and $\mathcal{T} \subseteq [n]$, $\boldsymbol{A}_{\mathcal{S},\mathcal{T}} \triangleq (A_{s,t})_{s \in \mathcal{S}, t \in \mathcal{T}} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{T}|}$ is the submatrix with rows in $\mathcal{S}$ and columns in $\mathcal{T}$, in increasing index order; its rows and columns keep the labels in $\mathcal{S}$ and $\mathcal{T}$. A single index i stands for $\{i\}$ and a colon for all indices; for example, $\boldsymbol{A}_{i,:} \triangleq \boldsymbol{A}_{\{i\},[n]}$ and $\boldsymbol{A}_{:,j} \triangleq \boldsymbol{A}_{[m],\{j\}}$. For a vector $\boldsymbol{a}$, $\boldsymbol{a}_{\mathcal{S}}$ selects the coordinates in $\mathcal{S}$.

- A submatrix is taken first: $\boldsymbol{A}^{-1}_{\mathcal{S},\mathcal{T}} \triangleq (\boldsymbol{A}_{\mathcal{S},\mathcal{T}})^{-1}$ and $\boldsymbol{A}^{\top}_{\mathcal{S},\mathcal{T}} \triangleq (\boldsymbol{A}_{\mathcal{S},\mathcal{T}})^{\top}$, whereas $(\boldsymbol{A}^{-1})_{\mathcal{S},\mathcal{T}}$ is a submatrix of the inverse.

Definition 1 (Layer and calibration inputs)**.** We fix positive integers m, n, T.

(a) *Layer.* A linear layer $\boldsymbol{y} = \boldsymbol{W} \cdot \boldsymbol{x}$ maps input $\boldsymbol{x} \in \mathbb{R}^n$ to output $\boldsymbol{y} \in \mathbb{R}^m$ through the weight matrix $\boldsymbol{W} \triangleq (W_{i,j})_{i\in[m],j\in[n]} \in \mathbb{R}^{m\times n}$; $W_{i,j}$ connects input feature j to output feature i.

(b) *Calibration inputs.* $\boldsymbol{X} \triangleq (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{n\times T}$ collects calibration inputs $\boldsymbol{x}_t \in \mathbb{R}^n$, $t \in [T]$.

(c) *Candidate weights.* $\boldsymbol{V} \triangleq (V_{i,j})_{i\in[m],j\in[n]} \in \mathbb{R}^{m\times n}$; the original and reconstructed outputs on the calibration inputs are $\boldsymbol{W} \cdot \boldsymbol{X}$ and $\boldsymbol{V} \cdot \boldsymbol{X}$. △

Following SparseGPT and Thanos, we measure their difference by the squared Frobenius norm $\|(\boldsymbol{V} - \boldsymbol{W}) \cdot \boldsymbol{X}\|_F^2$ [Frantar and Alistarh, 2023, Ilin and Richtárik, 2025].

Definition 2 (Pruning mask and damped reconstruction)**.** The fixed-mask problem has four ingredients.

(a) *Mask.* $\boldsymbol{M} \triangleq (M_{i,j})_{i\in[m],j\in[n]} \in \{0,1\}^{m\times n}$ is a fixed pruning mask [LeCun et al., 1989, Hassibi and Stork, 1992, Han et al., 2015]: retained entries ($M_{i,j} = 1$) may change, and deleted entries ($M_{i,j} = 0$) are fixed at zero. A retained entry can still have value zero.

(b) *Curvature.* $\boldsymbol{H}_0 \triangleq c \cdot \boldsymbol{X} \cdot \boldsymbol{X}^{\top}$ with normalization $c \triangleq 2/N$ for N calibration sequences [Frantar and Alistarh, 2023]; hence $\boldsymbol{H}_0$ is the Hessian of the sequence-averaged output error.

(c) *Damping.* $\boldsymbol{H} \triangleq (H_{i,j})_{i,j\in[n]} \triangleq \boldsymbol{H}_0 + \lambda \cdot \boldsymbol{I}$, with $\boldsymbol{I}$ the identity and $\lambda \triangleq \alpha \cdot (1/n) \cdot \sum_{j\in[n]} (\boldsymbol{H}_0)_{j,j}$ a fraction $\alpha \triangleq 0.01$ of the average diagonal entry of $\boldsymbol{H}_0$ [Levenberg, 1944, Hoerl and Kennard, 1970, Frantar and Alistarh, 2023].

(d) *Objective.* The damped reconstruction error function $Q : \mathbb{R}^{m\times n} \to \mathbb{R}_{\geq 0}$, a layer-wise reconstruction objective [Dong et al., 2017, Frantar et al., 2022, Frantar and Alistarh, 2023], is

$$Q(\boldsymbol{V}) \triangleq \operatorname{tr}\big((\boldsymbol{V} - \boldsymbol{W}) \cdot \boldsymbol{H} \cdot (\boldsymbol{V} - \boldsymbol{W})^{\top}\big), \qquad V_{i,j} = 0 \text{ when } M_{i,j} = 0. \tag{1}$$

△

Minimizing Q subject to the mask constraints is a well-posed convex problem. Indeed, $Q(\boldsymbol{V}) = c \cdot \|(\boldsymbol{V} - \boldsymbol{W}) \cdot \boldsymbol{X}\|_F^2 + \lambda \cdot \|\boldsymbol{V} - \boldsymbol{W}\|_F^2$. Each row's Hessian is $2 \cdot \boldsymbol{H}$, and $\boldsymbol{H}$ is symmetric positive definite (SPD) even when $T < n$, whereas $\boldsymbol{H}_0$ may be singular. Scaling c and λ by a common factor leaves the minimizers of Q unchanged, which is why λ is set relative to $\boldsymbol{H}_0$ (Appendix B.6). The problem therefore has a unique solution, which we derive next.

Fixed-mask solution. The objective and zero constraints separate across output rows. For one row i, the row objective is $q(\boldsymbol{V}_{i,:}) \triangleq (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:}) \cdot \boldsymbol{H} \cdot (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:})^{\top}$. We let $\mathcal{U} \subseteq [n]$ be the allowed column indices (the row's *support*) and $\mathcal{D} \triangleq [n] \setminus \mathcal{U}$. The following lemma gives the fixed-mask solution.

Lemma 3 (Fixed-mask reconstruction)**.** *For $\boldsymbol{H} \succ 0$ and nonempty $\mathcal{U}$, the unique minimizer of q subject to $\boldsymbol{V}_{i,\mathcal{D}} = 0$ is*

$$\boldsymbol{V}^*_{i,\mathcal{U}} = \boldsymbol{W}_{i,\mathcal{U}} + \boldsymbol{W}_{i,\mathcal{D}} \cdot \boldsymbol{H}_{\mathcal{D},\mathcal{U}} \cdot \boldsymbol{H}^{-1}_{\mathcal{U},\mathcal{U}}, \qquad \boldsymbol{V}^*_{i,\mathcal{D}} = 0. \tag{2}$$

Proof. Stationarity gives $(\boldsymbol{V}_{i,\mathcal{U}} - \boldsymbol{W}_{i,\mathcal{U}}) \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{U}} = \boldsymbol{W}_{i,\mathcal{D}} \cdot \boldsymbol{H}_{\mathcal{D},\mathcal{U}}$. Since $\boldsymbol{H}_{\mathcal{U},\mathcal{U}} \succ 0$, the solution is unique. If $\mathcal{U}$ is empty, the row is fixed at zero. ■

Evaluating Equation (2) for every row is impractical for two reasons. First, rows have different supports $\mathcal{U}$; hence each row needs its own factorization of $\boldsymbol{H}_{\mathcal{U},\mathcal{U}}$, which costs $O(m \cdot n^3)$ per layer [Frantar and Alistarh, 2023], where $O(\cdot)$ denotes an asymptotic upper bound [Knuth, 1997, Sec. 1.2.11.1]. Second, Equation (2) requires each row's deleted set $\mathcal{D}$ in advance, whereas a pruner selects the deletions in block $\mathcal{B}_r$ by scoring weights that compensation for the deletions in $\mathcal{B}_1, \ldots, \mathcal{B}_{r-1}$ has already changed; hence $\mathcal{D}$ is known only after the last block. Practical pruners therefore approximate Equation (2), as described next.

Forward approximation. Forward pruners replace Equation (2) with one left-to-right pass over the *column blocks* $\mathcal{B}_1, \ldots, \mathcal{B}_J$, consecutive groups of at most $b \in [n]$ columns, where $J \triangleq \lceil n/b \rceil$. For each block $\mathcal{B}_r$, a forward pruner selects deletions in $\boldsymbol{V}_{:,\mathcal{B}_r}$, compensates by changing only the columns of $\mathcal{B}_r, \mathcal{B}_{r+1}, \ldots$, and never changes earlier blocks again; one shared upper-triangular Cholesky factor $\boldsymbol{F} \in \mathbb{R}^{n\times n}$ of $\boldsymbol{H}^{-1}$, defined by $\boldsymbol{F}^\top \cdot \boldsymbol{F} = \boldsymbol{H}^{-1}$, keeps the cost at $O(m \cdot n^2)$. Our baselines span the compensation rules: none (Wanda, pruning by weights and activations [Sun et al., 2024]), one column at a time with score $V_{i,j}^2/F_{j,j}^2$ (SparseGPT [Frantar and Alistarh, 2023]), and one joint solve per row and block (Thanos [Ilin and Richtárik, 2025]). Block-OBS-GS combines SparseGPT's score with a Thanos-style row solve and adds GS sweeps that, with the mask fixed, revisit all retained weights; each sweep costs $O(m \cdot n^2)$, and the sweeps converge to Lemma 3's minimizer (the upcoming Theorem 7). The sweeps are needed because earlier blocks are frozen; a forward result therefore need not satisfy Lemma 3 for the final mask, as the following example shows.

Example 4 (Two column blocks)**.** We consider one row with $\boldsymbol{W}_{i,:} \triangleq \mathbf{1}^\top \in \mathbb{R}^{1\times 4}$, $\boldsymbol{H} \triangleq \boldsymbol{I} + \mathbf{1}\cdot\mathbf{1}^\top$, and $\mathcal{D} \triangleq \{1,3\}$; then $q(\boldsymbol{V}_{i,:}) = \big(\sum_{j\in[4]}(V_{i,j}-1)\big)^2 + \sum_{j\in[4]}(V_{i,j}-1)^2$. Zeroing without compensation gives $q = 6$; Equation (2) gives the optimum $(0, 5/3, 0, 5/3)$ with $q = 10/3$. Now prune with blocks $\{1,2\}$ and $\{3,4\}$. Block 1 zeros $V_{i,1}$; the weights $V_{i,2}, V_{i,3}, V_{i,4}$ are still free, all move to $5/4$, and $V_{i,2}$ is then optimal. Block 2 zeros $V_{i,3}$, but only $V_{i,4}$ may still change; it moves to $15/8$, giving $q = 115/32$. Now $\partial q/\partial V_{i,2} = -5/4 \neq 0$: zeroing $V_{i,3}$ has made $V_{i,2}$ suboptimal, and the forward pass never revisits it. One GS sweep with the mask fixed updates $V_{i,2}$ and $V_{i,4}$ and lowers q to $1715/512$; further sweeps approach $10/3$. Table 5 in Appendix A.4 summarizes this trace, where $R_{2,i}$, the $(2,i)$ entry of the GS residual $\boldsymbol{R} \triangleq \boldsymbol{H}\cdot(\boldsymbol{V}-\boldsymbol{W})^\top$ (Section 3.2), is half this derivative; Appendix A.4 gives every step. The MAP view below explains this gap in general. □

Reconstruction as MAP estimation on a factor graph. Minimizing q under the mask is MAP estimation on a factor graph, and this view locates the gap of the forward pass. For row i, define

$$
\begin{aligned}
&p(\boldsymbol{V}_{i,:}) \triangleq \prod_{t\in[T]} f_t(\boldsymbol{V}_{i,:}) \cdot \prod_{j\in[n]} g_j(V_{i,j}) \cdot \prod_{j\in\mathcal{D}} \delta_j(V_{i,j}),\\
&f_t(\boldsymbol{V}_{i,:}) \triangleq \exp\Big(-c\cdot\big((\boldsymbol{V}_{i,:}-\boldsymbol{W}_{i,:})\cdot\boldsymbol{x}_t\big)^2\Big), \qquad g_j(V_{i,j}) \triangleq \exp\big(-\lambda\cdot(V_{i,j}-W_{i,j})^2\big),\\
&\delta_j(V_{i,j}) \triangleq \mathbf{1}\{V_{i,j}=0\}.
\end{aligned}
\tag{3}
$$

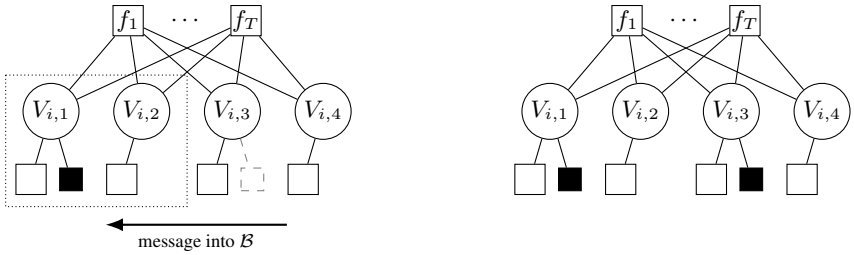
Then $p(\boldsymbol{V}_{i,:}) = \exp\big(-q(\boldsymbol{V}_{i,:})\big)\cdot\mathbf{1}\{\boldsymbol{V}_{i,\mathcal{D}}=0\}$, where q is the row objective of Lemma 3. Up to normalization, p is a Gaussian density with mean $\boldsymbol{W}_{i,:}$ and precision $2\cdot\boldsymbol{H}$, restricted to the mask; minimizing q under the mask is therefore MAP estimation, and its solution is Equation (2). The factorization in Equation (3) can be depicted by the factor graph in Figure 1 [Kschischang et al., 2001, Loeliger, 2004]. The two stages of Block-OBS-GS then read as follows.

(i) *Forward pass* (first stage; also SparseGPT and Thanos). When block $\mathcal{B} \triangleq \mathcal{B}_r$ is processed, the deletions in the later columns $\mathcal{R} \triangleq \bigcup_{r'>r}\mathcal{B}_{r'}$ have not been chosen yet; their constraint factors are therefore missing from the graph (Figure 1(a)). Treating the later weights as free, maximizing $\boldsymbol{V}_{i,\mathcal{R}}$ out of p gives a Gaussian message to $\boldsymbol{V}_{i,\mathcal{B}}$ with precision $2\cdot(\boldsymbol{F}_{\mathcal{B},\mathcal{B}}^\top\cdot\boldsymbol{F}_{\mathcal{B},\mathcal{B}})^{-1}$, where $\boldsymbol{F}_{\mathcal{B},\mathcal{B}}^\top\cdot\boldsymbol{F}_{\mathcal{B},\mathcal{B}} = \big((\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}\big)_{\mathcal{B},\mathcal{B}}$ because $\boldsymbol{F}$ is upper triangular (Appendix A). The first stage maximizes this message under the zeros of $\mathcal{B}$ and then fixes $\boldsymbol{V}_{i,\mathcal{B}}$ for good.

(ii) *The gap.* Later blocks add their constraint factors, which would change the message into $\mathcal{B}$, but $\boldsymbol{V}_{i,\mathcal{B}}$ is not updated (Example 4).

(iii) *Refinement* (second stage, ours). On the complete graph (Figure 1(b)), each GS update replaces one retained weight $V_{i,j}$ by its conditional mode, the maximizer of p over $V_{i,j}$ with all other weights fixed; it uses only the factors adjacent to $V_{i,j}$. Repeating such updates is iterated conditional modes (ICM) [Besag, 1986]; one GS sweep is thus one ICM sweep. ICM reaches only a local maximum in general, but p is log-concave on the mask; hence the limit is the global maximizer (the upcoming Theorem 7).

In short, the first stage decides on an incomplete graph and the second corrects on the complete one; Section 3 specifies both.



(a) Processing block $\mathcal{B} = \{1, 2\}$ (b) Complete graph, refined by GS

Figure 1: Factor graph of Equation (3) for Example 4: a bipartite graph with one variable node (circle) per weight $V_{i,j}$ and one factor node (square) per factor f_t, g_j, or δ_j (filled); a variable node and a factor node are joined by an edge if and only if the variable is an argument of the factor. (a) While block $\{1, 2\}$ is processed, the constraint on $V_{i,3}$ is missing (dashed). (b) GS acts on the complete graph.

3 Conditional block reconstruction and refinement

In this section, we specify the two stages of Block-OBS-GS: a forward pass followed by fixed-mask refinement. Figure 2 and Algorithm 1 summarize these steps.

3.1 Joint reconstruction within a column block

In the first stage, we visit the column blocks $\mathcal{B}_1, \ldots, \mathcal{B}_J$ of Section 2 from left to right. Each visit performs two steps: it selects the deletions in the block, and then reconstructs each row in the block and the later columns by one closed-form update. We first fix the quantities involved.

Definition 5 (Forward-pass quantities)**.** We fix a block index r and write $\mathcal{B} \triangleq \mathcal{B}_r$.

(a) *Iterates.* $\boldsymbol{V}^{(0)} \triangleq \boldsymbol{W}$, and $\boldsymbol{V}^{(r)} \in \mathbb{R}^{m \times n}$ denotes the candidate weights after $\mathcal{B}_r$ is processed, for $r \in [J]$; $\boldsymbol{V}^{(r-1)}$ is the current iterate and $\boldsymbol{V}^{(r)}$ the next.

(b) *Trailing columns.* $\mathcal{R} \triangleq \bigcup_{r' > r} \mathcal{B}_{r'}$ collects the columns not yet processed.

(c) *Block support.* For row $i \in [m]$, $\mathcal{D}_\mathcal{B} \triangleq \mathcal{D} \cap \mathcal{B}$ and $\mathcal{U}_\mathcal{B} \triangleq \mathcal{U} \cap \mathcal{B}$ are its deleted and retained columns in $\mathcal{B}$. $\triangle$

Step 1: selection. We score each entry of $\boldsymbol{V}^{(r-1)}_{:,\mathcal{B}}$ by the error its deletion would cause [Hassibi and Stork, 1992, Frantar and Alistarh, 2023]. If $V^{(r-1)}_{i,j}$ alone is set to zero and the columns $j, \ldots, n$ compensate optimally, q increases by $(V^{(r-1)}_{i,j})^2 / \big((\boldsymbol{H}_{\mathcal{S},\mathcal{S}})^{-1}\big)_{j,j}$ with $\mathcal{S} \triangleq \{j, \ldots, n\}$, which equals $(V^{(r-1)}_{i,j})^2 / F^2_{j,j}$ because $\boldsymbol{F}$ is upper triangular (Appendix C). We then delete the entries at or below one score threshold, chosen to delete the target fraction s of the block; a single threshold for all rows lets the rows keep different numbers of weights. This fixes $\mathcal{D}_\mathcal{B}$ and $\mathcal{U}_\mathcal{B}$; entries scoring exactly at the threshold are all deleted; we therefore report the actual zero counts.

Step 2: reconstruction. For row i, the *block solve* addresses the row problem of Lemma 3 with the information available at block $\mathcal{B}$: the earlier blocks are fixed, the deletions in $\mathcal{D}_\mathcal{B}$ are imposed, and the trailing weights stay free because their deletions are not chosen yet. That is, the block solve considers

$$\min_{\boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}}} q(\boldsymbol{V}_{i,:}) \quad \text{subject to } \boldsymbol{V}_{i,\mathcal{D}_\mathcal{B}} = 0, \text{ with } V_{i,j} = V^{(r-1)}_{i,j} \text{ for } j \in \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_{r-1}. \tag{4}$$

Equation (4) is again a problem of the form in Lemma 3. Because $\boldsymbol{V}^{(r-1)}_{i,\mathcal{B}\cup\mathcal{R}}$ minimizes q given the earlier blocks (Appendix A.1), q equals a constant plus $(\boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}\cup\mathcal{R}}) \cdot \boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}} \cdot (\boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}\cup\mathcal{R}})^\top$, which is the row objective of Lemma 3 on the columns $\mathcal{B} \cup \mathcal{R}$, with $\boldsymbol{V}^{(r-1)}_{i,\mathcal{B}\cup\mathcal{R}}$ in place of $\boldsymbol{W}_{i,:}$ and deleted set $\mathcal{D}_\mathcal{B}$. Lemma 3 would solve it with the large retained block of $\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}}$. Written instead with the inverse $(\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}$, as in OBS [Hassibi and Stork, 1992], whose rows

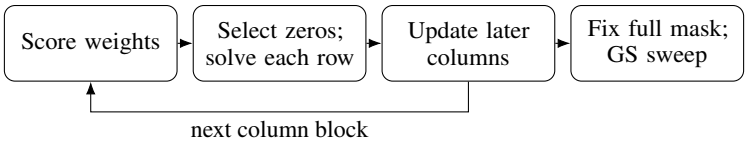


Figure 2: The two stages of Block-OBS-GS. The return arrow repeats for each column block.

in $\mathcal{D}_\mathcal{B}$ equal $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}\cup\mathcal{R}}$ because $\boldsymbol{F}$ is upper triangular, the same unique minimizer needs only the small deleted block (the upcoming Proposition 6):

$$\boldsymbol{V}^{(r)}_{i,\mathcal{B}\cup\mathcal{R}} = \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}\cup\mathcal{R}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{D}_\mathcal{B}} \cdot \left((\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}\right)^{-1} \cdot (\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}\cup\mathcal{R}}, \tag{5}$$

and the earlier columns are kept. On $\mathcal{D}_\mathcal{B}$, Equation (5) yields zero, because $\left((\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}\right)^{-1} \cdot (\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}} = \boldsymbol{I}$; on $\mathcal{U}_\mathcal{B}$, it adjusts the retained weights jointly; and on $\mathcal{R}$, it compensates through the trailing weights. To evaluate Equation (5), we obtain the row vector $\boldsymbol{V}^{(r-1)}_{i,\mathcal{D}_\mathcal{B}} \cdot \left((\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}\right)^{-1}$ by solving one linear system with the $|\mathcal{D}_\mathcal{B}| \times |\mathcal{D}_\mathcal{B}|$ matrix $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}$ (by Cholesky factorization, without forming its inverse), and multiply it by $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}\cup\mathcal{R}}$; both matrices are read off the shared factor $\boldsymbol{F}$ (Appendix C). Rows with equal deletion counts are solved in one batched call. After the last block, the deleted sets of all blocks form the mask $\boldsymbol{M}$ of Definition 2: $M_{i,j} = 0$ if and only if entry (i, j) was deleted in some block. The second stage keeps $\boldsymbol{M}$ fixed.

3.2 Scalar-coordinate Gauss–Seidel refinement

Refinement adjusts retained weights without changing the pruning mask $\boldsymbol{M}$. The iterates continue the same counter: $\boldsymbol{V}^{(J+k)}$ denotes the weights after k GS sweeps. Each iterate $t \in \{J+1, \ldots, J+K\}$ is one sweep that visits the columns $j = 1, \ldots, n$ in increasing order. For $M_{i,j} = 1$, i.e., for every retained entry, we set

$$V^{(t)}_{i,j} \triangleq V^{(t-1)}_{i,j} - \frac{1}{H_{j,j}} \cdot \underbrace{\Big(\sum_{l<j} H_{j,l} \cdot \big(V^{(t)}_{i,l} - W_{i,l}\big) + \sum_{l\geq j} H_{j,l} \cdot \big(V^{(t-1)}_{i,l} - W_{i,l}\big)\Big)}_{\triangleq R_{j,i}}, \tag{6}$$

and every deleted entry stays zero, i.e., $V^{(t)}_{i,j} \triangleq 0$ for $M_{i,j} = 0$. Each retained entry thus uses the latest values of the other entries, from iterate t for the columns $l < j$ and from iterate $t-1$ for $l \geq j$. This is cyclic coordinate GS, not a Jacobi or block-GS step. Thus $R_{j,i}$ is the (j, i) entry of the GS residual $\boldsymbol{R} = \boldsymbol{H} \cdot (\boldsymbol{V} - \boldsymbol{W})^\top \in \mathbb{R}^{n\times m}$ at these latest values. For all rows at once, Equation (6) therefore reads $\boldsymbol{V}^{(t)}_{:,j} = \boldsymbol{V}^{(t-1)}_{:,j} - (\boldsymbol{R}_{j,:})^\top / H_{j,j}$ on the retained entries. Since $\boldsymbol{R} = \sum_{l\in[n]} \boldsymbol{H}_{:,l} \cdot (\boldsymbol{V}_{:,l} - \boldsymbol{W}_{:,l})^\top$ and only column j changes, $\boldsymbol{R}$ then changes by $\boldsymbol{H}_{:,j} \cdot (\boldsymbol{V}^{(t)}_{:,j} - \boldsymbol{V}^{(t-1)}_{:,j})^\top$, which keeps it current for the next column.

To reduce overhead, the implementation applies these residual changes once per *chunk* of up to b consecutive columns, which yields the same iterates as Equation (6) up to floating-point rounding (Appendix B.3). We use $K = 1$ sweep by default and examine $K = 0, 2, 4$ in Section 5.3.

4 Guarantees and computational cost

In this section, we show that both stages are optimal for their own problems and that refinement keeps SparseGPT's complexity class.

4.1 What the reconstruction guarantees establish

Proposition 6 shows that the update in Equation (5) is optimal given $\boldsymbol{V}^{(r-1)}$, $\boldsymbol{H}$, and $\mathcal{D}_\mathcal{B}$, and Theorem 7 shows that repeated sweeps of Equation (6) reach the fixed-mask optimum in Equation (2).

Algorithm 1 Block-OBS-GS for one linear layer

Require: Weights $\boldsymbol{W}$, damped $\boldsymbol{H} \succ 0$ with upper Cholesky factor $\boldsymbol{F}$ of $\boldsymbol{H}^{-1}$, sparsity $s \in (0,1)$, blocks $\mathcal{B}_1, \ldots, \mathcal{B}_J$, and sweep count K.

Ensure: Pruned weights $\boldsymbol{V}^{(J+K)}$ and pruning mask $\boldsymbol{M}$.

Initialize the forward iterate as $\boldsymbol{V}^{(0)} \triangleq \boldsymbol{W}$. (first stage, Section 3)
for $r = 1, \ldots, J$, with $\mathcal{B} \triangleq \mathcal{B}_r$ and $\mathcal{R} \triangleq \bigcup_{r'>r} \mathcal{B}_{r'}$ **do**
  Delete the entries of $\boldsymbol{V}^{(r-1)}_{:,\mathcal{B}}$ with the lowest scores $(V^{(r-1)}_{i,j})^2 / F^2_{j,j}$. (Step 1)
  Compute $\boldsymbol{V}^{(r)}$ from Equation (5) for all rows, batching rows with equal $|\mathcal{D}_{\mathcal{B}}|$. (Step 2)
end for
Let $\boldsymbol{M}$ be the mask formed by all deleted entries.
Compute the residual $\boldsymbol{R} \triangleq \boldsymbol{H} \cdot (\boldsymbol{V}^{(J)} - \boldsymbol{W})^\top$. (second stage, Section 3.2)
for $t = J+1, \ldots, J+K$ **do**
  for $j = 1, \ldots, n$ **do**
    Set $\boldsymbol{V}^{(t)}_{:,j} \triangleq \boldsymbol{V}^{(t-1)}_{:,j} - (\boldsymbol{R}_{j,:})^\top / H_{j,j}$, keeping deleted entries zero (Equation (6)).
    Add $\boldsymbol{H}_{:,j} \cdot (\boldsymbol{V}^{(t)}_{:,j} - \boldsymbol{V}^{(t-1)}_{:,j})^\top$ to $\boldsymbol{R}$, because only column j has changed.
  end for
end for
return $\boldsymbol{V}^{(J+K)}$ and $\boldsymbol{M}$.

Proposition 6 (Conditional local optimality)**.** *For $\boldsymbol{H} \succ 0$ and a fixed deleted set $\mathcal{D}_{\mathcal{B}}$, Equation* (5) *gives the unique minimizer of Equation* (4)*. Its objective value is no larger than that of any other feasible point of Equation* (4) *for the same current iterate $\boldsymbol{V}^{(r-1)}$, damped curvature $\boldsymbol{H}$, and deleted set $\mathcal{D}_{\mathcal{B}}$, with strict inequality for every feasible point other than the minimizer.*

Proof. Lagrangian stationarity [Lagrange, 1788, Boyd and Vandenberghe, 2004] gives Equation (5). Any nonzero feasible perturbation increases the objective by a positive quadratic form. Appendix B.2 gives the derivation. ■

Theorem 7 (Descent and geometric convergence of refinement)**.** *We fix the mask $\boldsymbol{M}$ and an SPD matrix $\boldsymbol{H} \in \mathbb{R}^{n \times n}$. Each update of a retained weight $V_{i,j}$ in Equation* (6) *changes the row objective q of Lemma 3 by $-R^2_{j,i}/H_{j,j} \leq 0$, where $R_{j,i}$ is the (j,i) entry of the GS residual $\boldsymbol{R}$ in Equation* (6)*. For a nonempty retained set $\mathcal{U}$ of row i, let $\boldsymbol{e}^{(K)} \triangleq (\boldsymbol{V}^{(J+K)}_{i,\mathcal{U}} - \boldsymbol{V}^{*}_{i,\mathcal{U}})^\top$ be the error after K complete cyclic sweeps, with $\boldsymbol{V}^{*}_{i,\mathcal{U}}$ from Equation* (2)*. There is a factor $0 \leq \gamma < 1$, depending on $\boldsymbol{H}_{\mathcal{U},\mathcal{U}}$ and the coordinate order but not the initialization, such that*

$$\|\boldsymbol{e}^{(K)}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}} \leq \gamma^K \cdot \|\boldsymbol{e}^{(0)}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}}, \qquad \|\boldsymbol{e}\|^2_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}} \triangleq \boldsymbol{e}^\top \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{U}} \cdot \boldsymbol{e}. \tag{7}$$

Thus repeated sweeps converge geometrically to the unique fixed-mask minimizer in Equation (2)*. An empty retained set requires no refinement.*

Proof outline. Each coordinate update minimizes a one-dimensional quadratic. A complete sweep strictly decreases $\|\boldsymbol{e}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}}$ unless it starts at the optimum. Compactness of the unit sphere of $\|\cdot\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}}$ and the extreme value theorem [Rudin, 1976, Thm. 4.16] give a uniform contraction factor $\gamma < 1$. Appendix B.3 gives the full argument, a specialization of classical GS convergence [Saad, 2003]. ■

4.2 Same complexity class, different executed work

The forward pass costs the same order as SparseGPT's at fixed block width. We let $\ell_{i,r} \triangleq |\mathcal{D} \cap \mathcal{B}_r| \leq b$ be the number of weights deleted from row i in block $\mathcal{B}_r$. Recall that $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{X} \in \mathbb{R}^{n \times T}$ (Definition 1). Forming $\boldsymbol{H}_0 = c \cdot \boldsymbol{X} \cdot \boldsymbol{X}^\top$ (Definition 2) by a dense matrix product costs $O(n^2 \cdot T)$, and computing $\boldsymbol{F}$ by Cholesky factorizations of $\boldsymbol{H}$ and $\boldsymbol{H}^{-1}$ costs $O(n^3)$ [Golub and Van Loan, 2013]. In block $\mathcal{B}_r$, row i forms the $\ell_{i,r} \times \ell_{i,r}$ matrix $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_{\mathcal{B}}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_{\mathcal{B}}}$ of Equation (5) in $O(b \cdot \ell^2_{i,r})$, factors it by Cholesky factorization in $O(\ell^3_{i,r})$, and applies its inverse by two triangular solves in $O(\ell^2_{i,r})$; over all rows and blocks, this costs $O\big(\sum_{i,r}(\ell^3_{i,r} + b \cdot \ell^2_{i,r})\big) \subseteq O(m \cdot n \cdot b^2)$, because $\ell_{i,r} \leq b$

and $\sum_r \ell_{i,r} \le n$. Multiplying the resulting row vectors by $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}\cup\mathcal{R}}$ for all rows costs $O(m \cdot n^2 + m \cdot n \cdot b)$ in total.

Refinement adds work of the same order per sweep. Initializing $\boldsymbol{R} = \boldsymbol{H} \cdot (\boldsymbol{V}^{(J)} - \boldsymbol{W})^\top$, a product of an $n \times n$ and an $n \times m$ matrix, costs $O(m \cdot n^2)$. Within a sweep, column j reads $\boldsymbol{R}_{j,:}$ in $O(m)$ and applies the rank-one update $\boldsymbol{H}_{:,j} \cdot (\boldsymbol{V}^{(t)}_{:,j} - \boldsymbol{V}^{(t-1)}_{:,j})^\top$ to $\boldsymbol{R}$ in $O(m \cdot n)$; one sweep over the n columns therefore costs $O(m \cdot n^2)$; the chunked implementation of Section 3.2 performs the same updates. Summing the terms above over the J blocks and K sweeps gives

$$O\big(n^2 \cdot T + n^3 + (1+K) \cdot m \cdot n^2 + m \cdot n \cdot b^2\big). \tag{8}$$

For fixed b and K, the local-solve term satisfies $m \cdot n \cdot b^2 = O(m \cdot n)$ and the reconstruction term $(1+K) \cdot m \cdot n^2 = O(m \cdot n^2)$; hence Equation (8) is $O(n^3 + n^2 \cdot T + m \cdot n^2)$. This order is also attained. Both Block-OBS-GS and SparseGPT execute three dense computations: the product $\boldsymbol{X} \cdot \boldsymbol{X}^\top$; the Cholesky factorizations that give $\boldsymbol{F}$; and, in every block $\mathcal{B}_r$, the product of an $m \times b$ matrix with the $b \times |\mathcal{R}_r|$ matrix $\boldsymbol{F}_{\mathcal{B}_r,\mathcal{R}_r}$, where $\mathcal{R}_r \triangleq \bigcup_{r'>r} \mathcal{B}_{r'}$. In Block-OBS-GS, this last product is the trailing part of Equation (5); in SparseGPT, it is the lazy batch update of the trailing columns, and the other two computations are its Hessian and inverse-Cholesky steps [Frantar and Alistarh, 2023, Alg. 1]. The product $\boldsymbol{X} \cdot \boldsymbol{X}^\top$ takes $\Theta(n^2 \cdot T)$ operations, the Cholesky factorizations take $\Theta(n^3)$, and the block products take $\sum_{r \in [J]} m \cdot b \cdot |\mathcal{R}_r|$; this sum equals $m \cdot n \cdot (n-b)/2 = \Theta(m \cdot n^2)$ when b divides n and $n \ge 2 \cdot b$, because $|\mathcal{R}_r| = n - r \cdot b$. Both methods therefore lie in the class $\Theta(n^3 + n^2 \cdot T + m \cdot n^2)$, where $\Theta(\cdot)$ denotes a tight (upper and lower) bound [Knuth, 1976]; it counts the operations of these implementations, not a lower bound for pruning.

A fixed number of sweeps K therefore adds runtime but not asymptotic order. The block products in the trailing part of Equation (5) and the rank-one updates of $\boldsymbol{R}$ after Equation (6) have the same size at every sparsity, because zeros are stored and multiplied like other entries. A higher sparsity changes only the per-row systems in Equation (5), whose size $\ell_{i,r}$ is the number of deleted weights and thus grows with the sparsity; the pruning time therefore does not decrease as the sparsity increases (Section 5.2). SparseGPT's cost does not fall with the sparsity for the same reason: it visits every column of every block and updates all trailing columns with a dense row of $\boldsymbol{F}$, whatever the number of zeros [Frantar and Alistarh, 2023, Alg. 1]. The block part of Equation (5) can also be computed from the retained side, with one $|\mathcal{U}_\mathcal{B}| \times |\mathcal{U}_\mathcal{B}|$ system per row that shrinks as the sparsity grows; the evaluated implementation always uses the deleted side.

5 Experiments

In this section, we measure the quality gain, its time cost, its source, and its deployment consequences.

5.1 Protocol and quality at different sparsities

All quality comparisons share one calibration protocol. They use Llama-3.2-3B and Llama-3.1-8B [Grattafiori et al., 2024], bfloat16 (BF16) model weights, 32-bit floating-point (FP32) reconstruction, and WikiText-2 [Merity et al., 2017]. Calibration used 128 sequences of length 2,048 and seed 0; Block-OBS-GS used column-block width 128 and one GS sweep. Table 1 reports each pruned model against its matching dense reference; Appendix G gives the checkpoints, the source of each row, and the two evaluation families (end-to-end and fixed-mask).

Table 1 shows small differences at 25% and larger improvements at aggressive sparsities. At 87.5% sparsity on Llama-3.1-8B, SparseGPT gives PPL 256.7083 and Block-OBS-GS gives 137.1916, a reduction of 119.5167. The dense PPL of 6.2403 shows that this is lower PPL under severe pruning, not preserved dense-model quality. The PPL reduction thus grows with sparsity; whether it carries over to downstream accuracy is examined next.

Downstream gains are not uniform. Table 2 reports the dense and pruned 8B accuracies on a seven-task zero-shot average (Avg7) and five-shot Massive Multitask Language Understanding (MMLU). At 25%, 50%, and 75% sparsity, Block-OBS-GS trails SparseGPT on Avg7 by at most 0.2132 percentage points, and on five-shot MMLU it ties, loses 0.2208 points, and gains 0.7691 points, respectively. Lower PPL therefore does not establish downstream superiority.

Table 1: WikiText-2 PPL (lower is better; one calibration seed). The last column is the increase over the dense model.

Model	Sparsity	Dense	SparseGPT	Block-OBS-GS	Block − dense
3B	25%	7.8166	8.0679	8.0496	0.2330
3B	75%	7.8166	63.3754	49.0861	41.2695
3B	87.5%	7.8166	351.0129	206.2949	198.4782
8B	25%	6.2403	6.4881	6.4764	0.2362
8B	50%	6.2403	8.2735	8.0960	1.8557
8B	75%	6.2403	36.8552	31.6882	25.4479
8B	87.5%	6.2403	256.7083	137.1916	130.9513

Table 2: Llama-3.1-8B accuracy (%): seven-task zero-shot average (Avg7) and five-shot Massive Multitask Language Understanding (MMLU).

	Avg7		MMLU	
8B sparsity	SparseGPT	Block-OBS-GS	SparseGPT	Block-OBS-GS
Dense (0%)	71.4616	71.4616	65.3112	65.3112
25%	70.4328	70.4141	64.4281	64.4281
50%	64.8776	64.6967	53.3685	53.1477
75%	39.2113	38.9981	25.9578	26.7270

Table 3: Pruning time (s) at 8B/75%. Total wall time excludes PPL evaluation; the listed stages need not sum to it.

8B / 75% phase	SparseGPT (s)	Block-OBS-GS (s)
Calibration / Hessian	112.345	110.029
Shared factor preparation	8.961	9.293
Mask selection	2.509	3.239
Phase-1 reconstruction	236.926	672.853
GS refinement	0.000	56.706
Total pruning wall	474.731	952.782

Table 4: PPL at 3B/75% with the pruning mask fixed.

3B / 75% fixed-mask reconstruction	PPL
No reconstruction	1327.8287
SparseGPT reconstruction	63.3754
Block solve only ($K = 0$)	63.1194
Block solve + one GS sweep	52.6043
Block solve + two GS sweeps	51.3353
Block solve + four GS sweeps	50.6087

5.2 Pruning time and the reconstruction bottleneck

Table 3 compares the total pruning time and the time of each stage at 8B/75%.

Where the remaining cost occurs. The block solve of Equation (5), not GS, dominates the gap. In Table 3, Block-OBS-GS takes 478.051 seconds longer than SparseGPT in total (952.782 versus 474.731 seconds). Of this difference, 435.927 seconds come from Phase 1, the block solve (672.853 versus 236.926 seconds), and GS adds 56.706 seconds. The stage that computes the shared factor $\boldsymbol{F}$ takes 8.961 seconds for SparseGPT and 9.293 seconds for Block-OBS-GS, a difference of only 0.332 seconds; it does not include the per-row Cholesky factorizations of Equation (5), which are timed within Phase 1. The dense trailing products have the same shapes in both methods; the extra time lies in the row-specific work of Phase 1, which gathers the deleted columns of each row, factors many small $\ell_{i,r} \times \ell_{i,r}$ matrices, and scatters the results in small, irregular graphics processing unit (GPU) kernels (Table 9 in Appendix C). Phase 1 is therefore the main optimization target.

Engineering reduces, but does not close, the measured gap. An optimized implementation of Phase 1 (Appendix C) shortens pruning by about a quarter, while the gap to SparseGPT remains. On Llama-3.2-3B, the optimized implementation reduces median pruning time from 388.11 to 296.50 seconds at 25% sparsity and from 459.93 to 340.65 seconds at 50%: $1.309\times$ and $1.350\times$ speedups over the original implementation. These optimized times are $1.760\times$ and $2.056\times$ SparseGPT's medians, respectively. PPL changes by less than 0.005, although only 7 of 196 prunable-tensor masks match exactly (Table 12 in Appendix E.1). The next subsection attributes the PPL gain to the two stages.

5.3 What each stage contributes

GS refinement (Equation (6)) contributes more than the block solve (Equation (5)). Same-mask ablations compare reconstruction rules on exactly the same mask $\boldsymbol{M}$ and the same $\boldsymbol{H}$ within each condition; SparseGPT reconstruction denotes SparseGPT's column-by-column compensation (Section 2) applied to that mask. Table 4 shows that, at 3B/75%, one GS sweep adds more benefit than the block solve alone. Two and four sweeps further lower PPL to 51.3353 and 50.6087 (Table 4). Because the mask and $\boldsymbol{H}$ are fixed, the rows of Table 4 differ only in the reconstruction. In a complete run, reconstructed weights also change the inputs and masks of later layers, which explains the different values in Table 1.

The same comparison over all 12 model–sparsity conditions (Table 10 in Appendix D) gives a consistent result: in every condition, one GS sweep lowers PPL below both the block solve alone and

SparseGPT reconstruction, and each further sweep lowers it again. The block solve alone, in contrast, is worse than SparseGPT reconstruction in three conditions.

A second mask. The refinement gain persists on a second fixed mask, selected blockwise by Wanda-style magnitude–activation scores [Sun et al., 2024]. On 3B at 75% sparsity with this mask fixed, SparseGPT reconstruction gives PPL 182.9546, the block solve alone gives 195.4004, and the block solve followed by one GS sweep gives 159.6290. On this mask, the gain thus comes entirely from GS. On 8B, the corresponding values are 124.0989 and 111.0445.

Pruning followed by quantization. The PPL advantage survives subsequent quantization. In a matched 3B/50% experiment with GPTQ [Frantar et al., 2023] four-bit weight quantization (W4, group size 128), PPL after quantization is 16.1311 for SparseGPT and 15.3665 for Block-OBS-GS, against 13.4737 for GPTQ alone and 7.8166 for the BF16 dense model (Table 11 in Appendix D).

5.4 Scale, memory, and actual sparse execution

At 70B and 50% sparsity, the two methods give similar PPL and pruning time. On Llama-3.3-70B-Instruct, original Block-OBS-GS gives PPL 6.0389 in 4,313.3 seconds, versus 6.0800 in 4,247.8 seconds for SparseGPT. This single six-GPU run (Table 14 in Appendix E) establishes no scaling law. At 75%, an earlier run was much slower for Block-OBS-GS (Appendix E).

Pruning memory. Pruning memory shows no general advantage for either method (Appendix F). On 8B, the peak allocated GPU memory during pruning is 3.7355 GiB for both SparseGPT and Block-OBS-GS. At 70B/50%, the largest per-GPU allocated peak is 34.2492 GiB for Wanda, 38.8598 for SparseGPT, and 40.8522 for Block-OBS-GS.

Sparse storage and execution. Zeros save memory and time only if they are not stored. We therefore converted the pruned 8B model at 75% sparsity to a standard sparse format, compressed sparse row (CSR), and compared it with the same FP32 weights stored densely. For both methods, the weights shrink from 29.915 to 16.920 GiB, and generating tokens one at a time becomes about $1.28\times$ faster, but processing a 2,048-token prompt becomes about $34.9\times$ slower (Table 16 in Appendix F). The sparse FP32 weights are still larger than the usual dense BF16 weights (14.9575 GiB).

6 Related work

Block-OBS-GS follows the second-order line of OBS [Hassibi and Stork, 1992]. The ExactOBS solver of Optimal Brain Compression (OBC) updates all remaining weights of a row after each deletion [Frantar et al., 2022]; SparseGPT and Thanos freeze earlier blocks, which our GS stage revisits (Table 8 in Appendix C). SparseLLM couples several layers [Bai et al., 2024], whereas we refine one layer under a fixed mask. Mask selection [Sun et al., 2024, Yin et al., 2024, Fang et al., 2024], structured pruning [An et al., 2024], and quantization [Frantar et al., 2023] are complementary.

7 Conclusion

Block-OBS-GS adds fixed-mask GS sweeps to forward block reconstruction. The sweeps converge geometrically to the fixed-mask optimum without raising the complexity order, and in our experiments they lower PPL, most strongly at high sparsity, at the price of a longer pruning time. The study has limits: it covers Llama checkpoints, one calibration seed, and WikiText-2 PPL as the main metric; downstream accuracy does not improve consistently; and pruning remains about twice as slow as SparseGPT, which remains preferable when pruning time matters most. Future work includes faster batched implementations of Equation (5), an adaptive number of sweeps that stops once the residual-based error bound of Appendix A.3 falls below a tolerance, joint optimization of masks and weights, and refinement across layers.

Acknowledgments and Disclosure of Funding

The work described in this paper was partially supported by the National Natural Science Foundation of China under Grant 62601349, the Research Grants Council of Hong Kong under Grant 23201626, the Hong Kong University of Science and Technology (Guangzhou) under Grant G0101000351, and Lingnan University under Grant SDS24A16. The authors declare no competing interests.

AI-use statement

Generative artificial intelligence (AI) assisted with manuscript organization and editing, inspection of experimental records and implementation code, numerical checks, proof drafting, and preparation of tables and document-checking tools. Earlier AI-assisted work also contributed to experimental tooling and analysis. The disclosure describes assistance, not independent scientific validation. The human authors are responsible for checking the final claims, proofs, references, and experimental interpretations before submission.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873, 2024.

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. SparseLLM: Towards global pruning of pre-trained language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society: Series B*, 48(3):259–279, 1986.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337, 2023.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 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. Zenodo, version v0.4.3, 2024.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 4th edition, 2013.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal Brain Surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pages 164–171, 1992.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Ivan Ilin and Peter Richtárik. Thanos: A block-wise pruning algorithm for efficient large language model compression. *arXiv preprint arXiv:2504.05346*, 2025.

Donald E. Knuth. Big omicron and big omega and big theta. *ACM SIGACT News*, 8(2):18–24, 1976.

Donald E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, Reading, MA, 3rd edition, 1997.

Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

Joseph-Louis Lagrange. *Mécanique Analytique*. Desaint, Paris, 1788.

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

Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

Hans-Andrea Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1): 28–41, 2004.

Hans-Andrea Loeliger, Justin Dauwels, Junli Hu, Sascha Korl, Li Ping, and Frank R. Kschischang. The factor graph approach to model-based signal processing. *Proceedings of the IEEE*, 95(6): 1295–1322, 2007.

Dmitry M. Malioutov, Jason K. Johnson, and Alan S. Willsky. Walk-sums and belief propagation in Gaussian graphical models. *Journal of Machine Learning Research*, 7:2031–2064, 2006.

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

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2): 227–234, 1995.

Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, New York, 3rd edition, 1976.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.

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.

Yair Weiss and William T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

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, pages 57101–57115, 2024.

A Reconstruction as constrained MAP estimation

This appendix derives the MAP view of Section 2; it is an exact restatement of the fixed-mask quadratic, not a new approximation. Expanding the calibration term of the row objective gives

$$c \cdot \sum_{t\in[T]} \big((\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:}) \cdot \boldsymbol{x}_t\big)^2 = (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:}) \cdot \boldsymbol{H}_0 \cdot (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:})^\top,$$

and adding the damping term gives $q(\boldsymbol{V}_{i,:}) = (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:}) \cdot \boldsymbol{H} \cdot (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:})^\top$. Hence $\exp(-q) = \exp\big(-(1/2) \cdot (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:}) \cdot (2 \cdot \boldsymbol{H}) \cdot (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:})^\top\big)$ is an unnormalized Gaussian density with mean $\boldsymbol{W}_{i,:}$ and precision $2 \cdot \boldsymbol{H} \succ 0$. Restricted to the mask, it factors as in Equation (3). This factorization is the factor graph of Figure 1 [Kschischang et al., 2001, Loeliger, 2004]; MAP estimation on it combines messages obtained by maximizing variables out [Loeliger et al., 2007], which for Gaussian factors coincides with marginalizing them out. Every calibration factor is adjacent to every weight; hence the graph is dense and has cycles. Gaussian belief propagation on such a graph yields exact means when it converges, but convergence is not guaranteed [Weiss and Freeman, 2001, Malioutov et al., 2006], whereas GS converges geometrically (Theorem 7). Since a calibration term depends on all weights of the row, the graph does not imply a sparse $\boldsymbol{H}$. The indicator restricts the Gaussian to the affine subspace $\boldsymbol{V}_{i,\mathcal{D}} = 0$; the maximizer of a Gaussian on that subspace is its conditional mean, $\boldsymbol{V}^*_{i,\mathcal{U}} = \boldsymbol{W}_{i,\mathcal{U}} + \boldsymbol{W}_{i,\mathcal{D}} \cdot \boldsymbol{H}_{\mathcal{D},\mathcal{U}} \cdot \boldsymbol{H}^{-1}_{\mathcal{U},\mathcal{U}}$, which is Equation (2); MAP estimation needs no normalization constant. The following subsections apply this view to the forward pass and to GS.

If $(\boldsymbol{H}_0)_{j,j} = 0$ for some j, then the jth row and column of $\boldsymbol{H}_0$ vanish because $\boldsymbol{H}_0 \succeq 0$. The implementation then sets $\boldsymbol{W}_{:,j}$ to zero and $(\boldsymbol{H}_0)_{j,j}$ to one before damping; then q gains the term $V^2_{i,j}$, and the model above still applies.

A.1 Eliminating the trailing weights

Eliminating a trailing weight means marginalizing (equivalently, for a Gaussian, maximizing) over its possible values, not fixing it at zero (also called clamping). After earlier column blocks have been fixed, only the columns $\mathcal{B} \cup \mathcal{R}$ of row i can change. The current row $\boldsymbol{V}^{(r-1)}_{i,:}$ (Definition 5) minimizes q given the earlier blocks (shown in the next paragraph); hence the gradient of q in the columns $\mathcal{B} \cup \mathcal{R}$ vanishes there, and

$$q(\boldsymbol{V}_{i,:}) = q(\boldsymbol{V}^{(r-1)}_{i,:}) + (\boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}\cup\mathcal{R}}) \cdot \boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}} \cdot (\boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}\cup\mathcal{R}})^\top.$$

For fixed $\boldsymbol{V}_{i,\mathcal{B}}$, minimizing over the still-free trailing weights gives

$$\boldsymbol{V}_{i,\mathcal{R}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{R}} = -(\boldsymbol{V}_{i,\mathcal{B}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}}) \cdot \boldsymbol{H}_{\mathcal{B},\mathcal{R}} \cdot \boldsymbol{H}^{-1}_{\mathcal{R},\mathcal{R}}. \tag{9}$$

Substituting this expression leaves $q(\boldsymbol{V}^{(r-1)}_{i,:}) + (\boldsymbol{V}_{i,\mathcal{B}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}}) \cdot \boldsymbol{S} \cdot (\boldsymbol{V}_{i,\mathcal{B}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}})^\top$, where

$$\boldsymbol{S} \triangleq \boldsymbol{H}_{\mathcal{B},\mathcal{B}} - \boldsymbol{H}_{\mathcal{B},\mathcal{R}} \cdot \boldsymbol{H}^{-1}_{\mathcal{R},\mathcal{R}} \cdot \boldsymbol{H}_{\mathcal{R},\mathcal{B}} \tag{10}$$

is the Schur complement; up to a constant factor, it is the precision of the message from $\mathcal{R}$ into $\mathcal{B}$ in Figure 1(a). The block-inverse identity and the triangularity of $\boldsymbol{F}$ give $\boldsymbol{S}^{-1} = \big((\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}\big)_{\mathcal{B},\mathcal{B}} = \boldsymbol{F}^\top_{\mathcal{B},\mathcal{B}} \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}}$; hence the reduced quadratic in $\boldsymbol{V}_{i,\mathcal{B}}$ has the matrix $(\boldsymbol{F}^\top_{\mathcal{B},\mathcal{B}} \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}})^{-1}$, not $\boldsymbol{H}_{\mathcal{B},\mathcal{B}}$ alone.

The trailing part of Equation (5) is the minimizing trailing assignment in Equation (9). Indeed, the block part of Equation (5) gives $\boldsymbol{V}_{i,\mathcal{B}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{B}} = -\boldsymbol{V}^{(r-1)}_{i,\mathcal{D}_\mathcal{B}} \cdot \big((\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}\big)^{-1} \cdot (\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}}$, and the block-inverse identity gives $\boldsymbol{H}_{\mathcal{B},\mathcal{R}} \cdot \boldsymbol{H}^{-1}_{\mathcal{R},\mathcal{R}} = -\boldsymbol{F}^{-1}_{\mathcal{B},\mathcal{B}} \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{R}}$. Substituting both into Equation (9) yields $\boldsymbol{V}_{i,\mathcal{R}} - \boldsymbol{V}^{(r-1)}_{i,\mathcal{R}} = -\boldsymbol{V}^{(r-1)}_{i,\mathcal{D}_\mathcal{B}} \cdot \big((\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}\big)^{-1} \cdot (\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{R}}$, which is the trailing part of Equation (5). Initially, $\boldsymbol{V}^{(0)} = \boldsymbol{W}$ minimizes q without constraints. By induction over r, the trailing part of Equation (5) keeps $\boldsymbol{V}^{(r)}_{i,\mathcal{R}}$ the minimizer of q given the columns already fixed, which proves the claim used above. It also explains the limitation: future deletion constraints are not part of the current elimination problem. Once those constraints are imposed, previously fixed retained coordinates can cease to be optimal.

In MAP terms, eliminating the future weights produces the message from $\mathcal{R}$ into $\mathcal{B}$: a Gaussian factor on the current block with precision proportional to $\boldsymbol{S}$, and the trailing part of Equation (5) is the conditional mean of the trailing weights given the block update [Loeliger et al., 2007]. Since this factor can be dense, the graph yields no reduction in cost.

Table 5: Trace of Example 4. A weight $V_{i,2}$ with $R_{2,i} = 0$ is optimal given the other weights.

State in Example 4	$(V_{i,2}, V_{i,4})$	$R_{2,i}$	q
After block 1	$(5/4, 5/4)$	0	–
After block 2	$(5/4, 15/8)$	$-5/8$	$115/32$
After one GS sweep	$(25/16, 55/32)$	$-5/32$	$1715/512$
Optimum, Equation (2)	$(5/3, 5/3)$	0	$10/3$

A.2 Why later changes can invalidate an earlier adjustment

Later compensation shifts the residual of earlier weights and can thereby make them suboptimal. For row i, the gradient of q at $\boldsymbol{V}_{i,:}$ is $2 \cdot \boldsymbol{R}_{:,i}^\top$, where $\boldsymbol{R}_{:,i} = \boldsymbol{H} \cdot (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:})^\top$ is the ith column of the GS residual $\boldsymbol{R}$ defined in Section 3.2. We consider an earlier retained set $\mathcal{U}$ and a disjoint set $\mathcal{L}$ of later coordinates. If subsequent zeroing and compensation change only $\boldsymbol{V}_{i,\mathcal{L}}$, from $\boldsymbol{V}_{i,\mathcal{L}}^{\text{old}}$ to $\boldsymbol{V}_{i,\mathcal{L}}^{\text{new}}$, direct subtraction gives

$$\boldsymbol{R}_{\mathcal{U},i}^{\text{new}} = \boldsymbol{R}_{\mathcal{U},i}^{\text{old}} + \boldsymbol{H}_{\mathcal{U},\mathcal{L}} \cdot (\boldsymbol{V}_{i,\mathcal{L}}^{\text{new}} - \boldsymbol{V}_{i,\mathcal{L}}^{\text{old}})^\top. \tag{11}$$

Thus, even when $\boldsymbol{R}_{\mathcal{U},i}^{\text{old}} = 0$, the earlier weights cease to be stationary if $\boldsymbol{H}_{\mathcal{U},\mathcal{L}} \cdot (\boldsymbol{V}_{i,\mathcal{L}}^{\text{new}} - \boldsymbol{V}_{i,\mathcal{L}}^{\text{old}})^\top \neq 0$. For a retained coordinate j with $R_{j,i}^{\text{new}} \neq 0$, changing only $V_{i,j}$ by h changes q by $2 \cdot h \cdot R_{j,i}^{\text{new}} + h^2 \cdot H_{j,j}$. If $\boldsymbol{H} \succ 0$, taking $h = -R_{j,i}^{\text{new}} / H_{j,j}$ decreases it by $(R_{j,i}^{\text{new}})^2 / H_{j,j} > 0$ without violating the mask. Hence a forward-pass result with $\boldsymbol{R}_{\mathcal{U},i}^{\text{new}} \neq 0$ is not the fixed-mask optimum. Nonzero coupling alone is insufficient: $\boldsymbol{H}_{\mathcal{U},\mathcal{L}} \cdot (\boldsymbol{V}_{i,\mathcal{L}}^{\text{new}} - \boldsymbol{V}_{i,\mathcal{L}}^{\text{old}})^\top$ may vanish, or the earlier residual may already be nonzero. The claim is therefore a possible remaining error, not strict improvement for every mask or update. For $\boldsymbol{H} = c \cdot \boldsymbol{X} \cdot \boldsymbol{X}^\top + \lambda \cdot \boldsymbol{I}$ and disjoint $\mathcal{U}, \mathcal{L}$, the coupling is $\boldsymbol{H}_{\mathcal{U},\mathcal{L}} = c \cdot \boldsymbol{X}_{\mathcal{U},:} \cdot \boldsymbol{X}_{\mathcal{L},:}^\top$; these cross-products, rather than a qualitative claim about feature correlation, determine the effect.

A.3 The retained residual certifies the remaining error

The residual distinguishes an exact local solve from a completed global reconstruction. With the mask $\boldsymbol{M}$ fixed, consider the retained part $\boldsymbol{R}_{\mathcal{U},i}$ of the ith column of the GS residual $\boldsymbol{R}$, namely $\boldsymbol{R}_{:,i} = \boldsymbol{H} \cdot (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:})^\top$. Completing the square around the unique constrained optimum $\boldsymbol{V}_{i,:}^*$ of Equation (2) gives

$$q(\boldsymbol{V}_{i,:}) - q(\boldsymbol{V}_{i,:}^*) = (\boldsymbol{V}_{i,\mathcal{U}} - \boldsymbol{V}_{i,\mathcal{U}}^*) \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{U}} \cdot (\boldsymbol{V}_{i,\mathcal{U}} - \boldsymbol{V}_{i,\mathcal{U}}^*)^\top = \boldsymbol{R}_{\mathcal{U},i}^\top \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{U}}^{-1} \cdot \boldsymbol{R}_{\mathcal{U},i}. \tag{12}$$

Thus $\boldsymbol{R}_{\mathcal{U},i} = 0$ if and only if the fixed-mask quadratic has been solved globally. If $\boldsymbol{H} = \boldsymbol{H}_0 + \lambda \cdot \boldsymbol{I}$ with $\boldsymbol{H}_0 \succeq 0$ and $\lambda > 0$, then

$$0 \leq q(\boldsymbol{V}_{i,:}) - q(\boldsymbol{V}_{i,:}^*) \leq \|\boldsymbol{R}_{\mathcal{U},i}\|_2^2 / \lambda.$$

The bound uses the residual already maintained by Equation (6), without solving another full system. It gives a conservative stopping rule for the quadratic; our experiments use fixed sweep counts.

Each retained-coordinate update minimizes its one-dimensional quadratic: the derivative is $2 \cdot R_{j,i}$ and the curvature $2 \cdot H_{j,j}$. Subtracting $R_{j,i} / H_{j,j}$ from $V_{i,j}$ is therefore exactly Equation (6). In the MAP view, this update maximizes p in Equation (3) over $V_{i,j}$ with all other weights held fixed, which is one ICM step [Besag, 1986]. The view explains why later zero constraints create a residual to revisit; the SPD argument, rather than a generic message-passing claim, proves descent and convergence.

A.4 A worked two-block reconstruction

A two-block example makes this gap concrete. For one output row i, we take $\boldsymbol{W}_{i,:} \triangleq (1, 1, 1, 1)$ and $\boldsymbol{H} \triangleq \boldsymbol{I} + \boldsymbol{1} \cdot \boldsymbol{1}^\top$, which gives

$$q(\boldsymbol{V}_{i,:}) = \left(\sum_{j=1}^{4} (V_{i,j} - 1) \right)^2 + \sum_{j=1}^{4} (V_{i,j} - 1)^2.$$

We fix $V_{i,1} = V_{i,3} = 0$ and use blocks $\mathcal{B}_1 \triangleq \{1,2\}$ and $\mathcal{B}_2 \triangleq \{3,4\}$. For $\mathcal{B}_1$, the matrix $\boldsymbol{F}_{\mathcal{B}_1,\mathcal{B}_1}^\top \cdot \boldsymbol{F}_{\mathcal{B}_1,\mathcal{B}_1}$ and the incoming weights are

$$\boldsymbol{F}_{\mathcal{B}_1,\mathcal{B}_1}^\top \cdot \boldsymbol{F}_{\mathcal{B}_1,\mathcal{B}_1} = \begin{pmatrix} 4/5 & -1/5 \\ -1/5 & 4/5 \end{pmatrix}, \qquad \boldsymbol{V}_{i,\mathcal{B}_1} = (1,1).$$

Equation (5) gives $V_{i,2} = 5/4$ and changes the incoming second block to $\boldsymbol{V}_{i,\mathcal{B}_2} = (5/4, 5/4)$. For $\mathcal{B}_2$,

$$\boldsymbol{F}_{\mathcal{B}_2,\mathcal{B}_2}^\top \cdot \boldsymbol{F}_{\mathcal{B}_2,\mathcal{B}_2} = \begin{pmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \end{pmatrix},$$

so Equation (5) gives $V_{i,4} = 15/8$. The first stage therefore ends at $(0, 5/4, 0, 15/8)$ with $q = 115/32$.

Processing $\mathcal{B}_2$ makes $\partial q / \partial V_{i,2} = -5/4 \neq 0$ (Example 4). One left-to-right GS sweep revisits coordinates 2 and 4, giving $(0, 25/16, 0, 55/32)$ and $q = 1715/512$, below $115/32$. The fixed-mask quadratic optimum is $(0, 5/3, 0, 5/3)$ with $q = 10/3$. Thus the two solves of Equation (5) are exact for their conditional problems, one sweep improves the full objective, and that sweep need not reach the optimum.

B Proofs and scope of guarantees

This appendix proves the results of Sections 2 and 4 and delimits their scope.

B.1 Details for Lemma 3

The fixed-mask row problem is a strictly convex quadratic in the retained weights. For row i, we set $\boldsymbol{V}_{i,\mathcal{D}} = 0$. The objective is

$$\begin{aligned} q(\boldsymbol{V}_{i,:}) = (\boldsymbol{V}_{i,\mathcal{U}} - \boldsymbol{W}_{i,\mathcal{U}}) \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{U}} \cdot (\boldsymbol{V}_{i,\mathcal{U}} - \boldsymbol{W}_{i,\mathcal{U}})^\top \\ - 2 \cdot (\boldsymbol{V}_{i,\mathcal{U}} - \boldsymbol{W}_{i,\mathcal{U}}) \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{D}} \cdot \boldsymbol{W}_{i,\mathcal{D}}^\top + \boldsymbol{W}_{i,\mathcal{D}} \cdot \boldsymbol{H}_{\mathcal{D},\mathcal{D}} \cdot \boldsymbol{W}_{i,\mathcal{D}}^\top. \end{aligned} \tag{13}$$

Its gradient with respect to $\boldsymbol{V}_{i,\mathcal{U}}$ is $2 \cdot (\boldsymbol{V}_{i,\mathcal{U}} - \boldsymbol{W}_{i,\mathcal{U}}) \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{U}} - 2 \cdot \boldsymbol{W}_{i,\mathcal{D}} \cdot \boldsymbol{H}_{\mathcal{D},\mathcal{U}}$. Positive definiteness of $\boldsymbol{H}$ implies positive definiteness of its principal submatrix $\boldsymbol{H}_{\mathcal{U},\mathcal{U}}$; hence the stationary point in Equation (2) is the unique minimizer. If $\mathcal{U} = \emptyset$, the row is zero and no solve is needed. Rows with different $\mathcal{U}$ need different submatrices $\boldsymbol{H}_{\mathcal{U},\mathcal{U}}$; hence one factorization of $\boldsymbol{H}$ does not serve all rows.

Row-wise minimization therefore solves the whole layer: Q is the sum of the row objectives q, with no cross-row terms, and the mask constraints separate by row. Minimizing each row thus gives the unique minimizer of Q under the fixed mask $\boldsymbol{M}$.

B.2 Proof of Proposition 6

The proposition follows from Lagrangian stationarity [Lagrange, 1788, Boyd and Vandenberghe, 2004] and strict convexity. The objective of Equation (4) is $q(\boldsymbol{V}_{i,:}^{(r-1)})$ plus a quadratic form in $\boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}} - \boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}}^{(r-1)}$ with matrix $\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}}$. We let $\boldsymbol{\mu} \in \mathbb{R}^{1\times|\mathcal{D}_\mathcal{B}|}$ be the constraint multiplier, using one half of the objective in the Lagrangian. Stationarity gives $\boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}} = \boldsymbol{V}_{i,\mathcal{B}\cup\mathcal{R}}^{(r-1)} - \boldsymbol{\mu} \cdot \left((\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}\right)_{\mathcal{D}_\mathcal{B},\mathcal{B}\cup\mathcal{R}}$, whose matrix equals $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}\cup\mathcal{R}}$ because $\boldsymbol{F}$ is upper triangular, and the constraint $\boldsymbol{V}_{i,\mathcal{D}_\mathcal{B}} = 0$ gives $\boldsymbol{\mu} = \boldsymbol{V}_{i,\mathcal{D}_\mathcal{B}}^{(r-1)} \cdot \left((\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}\right)^{-1}$, which is Equation (5). For any feasible perturbation $\boldsymbol{h} \in \mathbb{R}^{1\times|\mathcal{B}\cup\mathcal{R}|}$ of this solution, the cross term vanishes by stationarity and the objective increases by $\boldsymbol{h} \cdot \boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}} \cdot \boldsymbol{h}^\top$, which is strictly positive unless $\boldsymbol{h} = 0$.

This proof holds $\boldsymbol{V}^{(r-1)}$, $\boldsymbol{H}$, and $\mathcal{D}_\mathcal{B}$ fixed. The complete fixed-mask optimum is given by Equation (2); a finite sequence of updates of the form in Equation (5) need not reach it. GS closes this gap, as proved next.

B.3 Proof of Theorem 7

Each GS update is an exact line minimization, which yields monotone descent. We fix row i with retained set $\mathcal{U}$. As a function of $\boldsymbol{V}_{i,\mathcal{U}}$, the row objective is a quadratic with Hessian $2 \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{U}}$ and

gradient $2 \cdot \boldsymbol{R}_{\mathcal{U},i}^{\top}$, where $\boldsymbol{R}_{:,i} = \boldsymbol{H} \cdot (\boldsymbol{V}_{i,:} - \boldsymbol{W}_{i,:})^{\top}$ is the ith column of the GS residual $\boldsymbol{R}$ defined in Section 3.2. Changing $V_{i,j}$, $j \in \mathcal{U}$, by $-R_{j,i}/H_{j,j}$ changes q by

$$-2 \cdot R_{j,i}^2/H_{j,j} + H_{j,j} \cdot (R_{j,i}/H_{j,j})^2 = -R_{j,i}^2/H_{j,j}. \tag{14}$$

Thus each update is an exact line minimization, and a complete sweep is nonincreasing. If an entire sweep leaves the objective unchanged, every update is zero; hence the original residual is zero in every coordinate. The starting point is then the unique optimum.

Strict descent per sweep becomes a uniform geometric rate by compactness. We let $\boldsymbol{G} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{U}|}$ be the matrix that maps the error before one complete sweep to the error after it; hence $\boldsymbol{e}^{(K)} = \boldsymbol{G}^K \cdot \boldsymbol{e}^{(0)}$. By the previous paragraph, $\|\boldsymbol{G} \cdot \boldsymbol{e}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}} < \|\boldsymbol{e}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}}$ for every $\boldsymbol{e} \neq 0$. Continuity and compactness of the unit sphere of $\|\cdot\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}}$, with the extreme value theorem [Rudin, 1976, Thm. 4.16], therefore give $\gamma \triangleq \|\boldsymbol{G}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}} = \max_{\|\boldsymbol{e}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}}=1} \|\boldsymbol{G} \cdot \boldsymbol{e}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}} < 1$. Submultiplicativity yields Equation (7). Completing the square gives the corresponding objective-gap bound:

$$q(\boldsymbol{V}_{i,\mathcal{U}}^{(J+K)}) - q(\boldsymbol{V}_{i,\mathcal{U}}^{*}) = \|\boldsymbol{e}^{(K)}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}}^2 \leq \gamma^{2 \cdot K} \cdot \big(q(\boldsymbol{V}_{i,\mathcal{U}}^{(J)}) - q(\boldsymbol{V}_{i,\mathcal{U}}^{*})\big). \tag{15}$$

Here $\|\boldsymbol{G}\|_{\boldsymbol{H}_{\mathcal{U},\mathcal{U}}} \triangleq \|\boldsymbol{H}_{\mathcal{U},\mathcal{U}}^{1/2} \cdot \boldsymbol{G} \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{U}}^{-1/2}\|_2$, whereas the spectral radius $\rho(\boldsymbol{G})$ is the largest eigenvalue magnitude. The spectral radius $\rho(\boldsymbol{G}) \leq \gamma$ governs the asymptotic rate, and both depend on $\boldsymbol{H}_{\mathcal{U},\mathcal{U}}$ and the coordinate order. Standard stationary-iteration theory provides the same SPD convergence result [Saad, 2003].

The implemented chunked update has the same exact-arithmetic recurrence. Within a chunk with column set $\mathcal{C}$, the entries $\boldsymbol{R}_{\mathcal{C},:}$ are updated after each column, as Equation (6) requires. The changes $\boldsymbol{V}_{:,j}^{(t)} - \boldsymbol{V}_{:,j}^{(t-1)}$ of a chunk's columns, stacked as the columns of one matrix, therefore equal the sequence of scalar updates in Equation (6), and one product of $\boldsymbol{H}_{:,\mathcal{C}}$ with the transpose of that matrix applies their sum to all rows of the residual. This proves the recurrence's equivalence, not bitwise floating-point equality. If finite-precision factorization fails or a nonfinite value appears, the run is rejected.

What converges. Theorem 7 concerns the fixed-mask quadratic, not mask selection or model perplexity, and it holds for any starting point, including the first-stage output (Appendix B.4). The gap $q(\boldsymbol{V}_{i,:}^{(J+K)}) - q(\boldsymbol{V}_{i,:}^{*})$ contracts by at most $\gamma^{2 \cdot K}$ relative to $q(\boldsymbol{V}_{i,:}^{(J)}) - q(\boldsymbol{V}_{i,:}^{*})$ (Equation (15)). A factor near one permits slow convergence.

B.4 Initialization and the role of block reconstruction

For a fixed mask, the damped quadratic is strictly convex in the retained weights. The limit in Theorem 7 is therefore its unique global minimum, not merely a local minimum. One could instead set the selected weights to zero, leave the others unchanged, and apply GS: repeated sweeps would converge to the same optimum for that same mask. The first stage (Equation (5)) is not necessary for this asymptotic guarantee. It supplies the starting point $\boldsymbol{V}^{(J)}$ for a finite number K of sweeps ($K = 1$ by default) and changes the weights used to select zeros in later blocks. Skipping it can therefore change both the starting point and the final mask. The theorem does not guarantee that its initialization is better than every alternative after a fixed number of sweeps; that comparison requires an ablation with the mask held fixed.

B.5 Why finite sweeps need not dominate another method

Example 8 (Finite sweeps and different initializations)**.** We take one row without deletions, $\boldsymbol{H} \triangleq \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$, and $\boldsymbol{W}_{i,:} \triangleq (1/3, 1/3)$, which is also the optimum. A GS sweep from zero gives $(1/2, 1/4)$, which has positive error while a comparator initialized at the optimum has zero error. Descent from one's own initialization cannot order arbitrary comparators. Conversely, for the same $\boldsymbol{H}$ and $\boldsymbol{W}_{i,:} \triangleq (1, 0)$, a sweep from zero reaches $(1, 0)$ exactly despite nonzero off-diagonal coupling. Coupling alone therefore does not imply strict dominance. □

Table 6: Notation. $\mathcal{R}$ is a set of trailing columns; $\boldsymbol{R}$ is the GS residual.

Symbol	Meaning / domain	Symbol	Meaning / domain
m, n, T	Positive integers: output features, input features, calibration tokens	$\boldsymbol{x}, \boldsymbol{y}$	Input in $\mathbb{R}^n$; output in $\mathbb{R}^m$
$\boldsymbol{W}, \boldsymbol{V}$	Reference and candidate weights in $\mathbb{R}^{m \times n}$	$\boldsymbol{X}$	Calibration inputs in $\mathbb{R}^{n \times T}$
$\boldsymbol{H}_0, \boldsymbol{H}$	Undamped and damped curvature in $\mathbb{R}^{n \times n}$	$\boldsymbol{M}$	Mask in $\{0,1\}^{m \times n}$
b, K	Block width $1 \le b \le n$; integer sweep count $K \ge 0$	$\mathcal{B}, \mathcal{R}$	Current and trailing column sets
$\mathcal{U}, \mathcal{D}$	Retained and deleted columns of a row; $\mathcal{U}_\mathcal{B}, \mathcal{D}_\mathcal{B}$ their parts in $\mathcal{B}$	q	Row objective of Lemma 3
$\boldsymbol{F}$	Upper Cholesky factor of $\boldsymbol{H}^{-1}$	$\boldsymbol{R}$	GS gradient residual in $\mathbb{R}^{n \times m}$

B.6 Role of damping and damping perturbation

Damping makes $\boldsymbol{H}$ SPD, bounds its conditioning, and acts as a ridge penalty. The undamped curvature $\boldsymbol{H}_0$ is only positive semidefinite; it is singular when $T < n$ or an input feature is inactive. For any nonzero $\boldsymbol{u} \in \mathbb{R}^n$, $\boldsymbol{u}^\top \cdot \boldsymbol{H} \cdot \boldsymbol{u} = c \cdot \|\boldsymbol{X}^\top \cdot \boldsymbol{u}\|_2^2 + \lambda \cdot \|\boldsymbol{u}\|_2^2 \ge \lambda \cdot \|\boldsymbol{u}\|_2^2 > 0$; hence $\boldsymbol{H}$ is SPD. For any nonempty retained set $\mathcal{U} \subseteq [n]$, the principal submatrix $\boldsymbol{H}_{\mathcal{U},\mathcal{U}}$ used in Lemma 3 has smallest eigenvalue at least λ, and its condition number is at most $1 + \lambda_{\max}(\boldsymbol{H}_0)/\lambda$. This bound stabilizes the FP32 Cholesky factorizations. Because $Q(\boldsymbol{V}) = c \cdot \|(\boldsymbol{V} - \boldsymbol{W}) \cdot \boldsymbol{X}\|_F^2 + \lambda \cdot \|\boldsymbol{V} - \boldsymbol{W}\|_F^2$, damping is a ridge penalty that keeps retained weights near their original values and limits overfitting to the calibration inputs [Hoerl and Kennard, 1970]. The two limits delimit the trade-off. As $\lambda \to 0$, the damped minimizer approaches the undamped one when $(\boldsymbol{H}_0)_{\mathcal{U},\mathcal{U}} \succ 0$, as quantified below. As $\lambda \to \infty$, $\boldsymbol{H}_{\mathcal{U},\mathcal{U}}^{-1} \cdot \boldsymbol{H}_{\mathcal{U},\mathcal{D}} \to 0$ in Equation (2); hence retained weights stay at $\boldsymbol{W}_{i,\mathcal{U}}$ and reconstruction reduces to zeroing without compensation.

Small damping perturbs the undamped minimizer by $O(\lambda)$. We write $\boldsymbol{V}_{i,\mathcal{U}}^*(\lambda)$ for the minimizer in Equation (2) with damping λ, and assume $(\boldsymbol{H}_0)_{\mathcal{U},\mathcal{U}} \succ 0$; then $\boldsymbol{V}_{i,\mathcal{U}}^*(0)$ exists. Since $\boldsymbol{H}_{\mathcal{D},\mathcal{U}} = (\boldsymbol{H}_0)_{\mathcal{D},\mathcal{U}}$, the resolvent identity gives

$$\begin{aligned} \boldsymbol{V}_{i,\mathcal{U}}^*(\lambda) - \boldsymbol{V}_{i,\mathcal{U}}^*(0) &= -\lambda \cdot \big(\boldsymbol{V}_{i,\mathcal{U}}^*(0) - \boldsymbol{W}_{i,\mathcal{U}}\big) \cdot \big((\boldsymbol{H}_0)_{\mathcal{U},\mathcal{U}} + \lambda \cdot \boldsymbol{I}\big)^{-1}, \\ \|\boldsymbol{V}_{i,\mathcal{U}}^*(\lambda) - \boldsymbol{V}_{i,\mathcal{U}}^*(0)\|_2 &\le \frac{\lambda}{\lambda_{\min}\big((\boldsymbol{H}_0)_{\mathcal{U},\mathcal{U}}\big) + \lambda} \cdot \|\boldsymbol{V}_{i,\mathcal{U}}^*(0) - \boldsymbol{W}_{i,\mathcal{U}}\|_2. \end{aligned} \tag{16}$$

This is an $O(\lambda)$ bound for fixed well-conditioned $(\boldsymbol{H}_0)_{\mathcal{U},\mathcal{U}}$, not a guarantee for a singular undamped system.

C Implementation correspondence and operation counts

This appendix maps the equations to the code and counts their operations. Table 7 separates the three decisions in which our method differs from prior rules. The same deletion score does not ensure identical masks, because earlier weight adjustments change the weights scored in later blocks; Proposition 6 therefore compares reconstructions only for the same current weights and deleted entries.

Factor construction. The original and the memory-bounded implementations both compute the blocks of $(\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}$ needed by Equation (5). The original implementation forms the upper Cholesky factor $\boldsymbol{F}$ of $\boldsymbol{H}^{-1}$ once. Since $\boldsymbol{F}$ is upper triangular, $\big((\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}\big)_{\mathcal{B},\mathcal{B}\cup\mathcal{R}} = \boldsymbol{F}_{\mathcal{B},\mathcal{B}}^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}\cup\mathcal{R}}$. The memory-bounded large-model implementation obtains the needed blocks of the tail inverse $(\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}$ by reverse-Cholesky solves; it takes $\boldsymbol{F}_{\mathcal{B},\mathcal{B}}$ as the upper Cholesky factor of $\big((\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}\big)_{\mathcal{B},\mathcal{B}}$ and solves $\boldsymbol{F}_{\mathcal{B},\mathcal{B}}^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{R}} = \big((\boldsymbol{H}_{\mathcal{B}\cup\mathcal{R},\mathcal{B}\cup\mathcal{R}})^{-1}\big)_{\mathcal{B},\mathcal{R}}$. Both therefore use blocks of the inverse, not the diagonal block $\boldsymbol{H}_{\mathcal{B},\mathcal{B}}$.

Table 7: Selection and reconstruction rules of the published designs.

Method	Selection rule	Reconstruction during pruning	Full fixed-mask refinement
Wanda	Weight magnitude times activation scale	None	None
SparseGPT	Inverse-factor saliency	Ordered column compensation and trailing propagation	None
Thanos	Blockwise score and adaptive mask	Joint block compensation and subsequent weight updates	Not the GS sweep studied here
Block-OBS-GS	SparseGPT-style saliency	Conditional joint solve and factor-space trailing propagation	Cyclic coordinate GS

Table 8: Comparison by optimization choice, not by performance.

Method	Selection rule / target	Which values are adjusted?	Relation to full-mask refinement
ExactOBS (OBC)	Greedy deletion cost	All remaining weights in a row after each deletion	Not forward-only; no separate stage
Wanda	Magnitude · activation norm	No retained-weight updates	No reconstruction
SparseGPT	Estimated deletion cost	Later columns; earlier columns frozen	No separate stage
Thanos	Magnitude · activation norm; adaptive mask	Joint block correction propagated forward	No separate stage
Block-OBS-GS	SparseGPT-style score	Joint row-wise block solve, then later columns	**GS revisits all retained weights**; zeros fixed
SparseLLM	Masks via local pruning solvers	Weights and auxiliary activations across coupled layers	Different, cross-layer objective
OWL	Layer sparsity budgets	Determined by underlying pruner	Complementary allocation rule
MaskLLM	Learned $N{:}M$ masks	Mask parameters; model weights frozen	Mask learning, not weight refinement
FLAP	Structured components	Bias compensation	Different sparsity structure and recovery
GPTQ	Quantized weight values	Later, not-yet-quantized weights	Precision reduction, not zero selection

Input handling. Numerical preparation precedes the exact-arithmetic analysis. The reference code copies $\boldsymbol{W}$ and $\boldsymbol{H}_0$ into FP32, handles columns with $(\boldsymbol{H}_0)_{j,j} = 0$ as in Appendix A, adds the damping λ of Definition 2, and keeps $\boldsymbol{W}$ as the reference for GS. It checks that $\boldsymbol{H}$ is symmetric and finite and rejects the run otherwise; the SPD assumption of Section 4 applies after these steps. Converting the result back to BF16 adds rounding that the exact-arithmetic proofs do not cover.

Selection. Inclusive block thresholding explains the achieved sparsities. For the $N_{\mathcal{B}} \triangleq m \cdot |\mathcal{B}|$ entries of a block, scores are $(V_{i,j}^{(r-1)})^2 / F_{j,j}^2$ with $j \in \mathcal{B}$, as in Step 1 of Section 3. The implementation sorts all scores in the block and deletes entries at or below the selected threshold. With zero-based index $\lfloor N_{\mathcal{B}} \cdot s \rfloor$, inclusive thresholding can delete one more than the nominal count in the absence of ties; ties can add more. This explains achieved sparsities such as 0.5000024 rather than exactly 0.5. The experiments record the achieved counts of each method.

Implementation variants. The engineering variant changes execution, not mathematics. The original inverse-factor reconstruction and dense-residual GS define the quality and mechanism studies. The separately evaluated optimized implementation (Section 5.2) uses stable grouping by deletion count, one stable mask partition, deferred Cholesky-status aggregation, direct scatter, and

Table 9: Operation shapes and GPU execution for an $m \times n$ layer; q is a row-chunk size and $\ell \leq b$ a common deletion count.

Operation	SparseGPT	Block-OBS-GS	GPU execution
Shared preparation	Dense curvature and inverse-factor preparation	Same type of dense preparation	Large dense kernels in both methods
Local compensation	Ordered column errors; dense suffix updates	Row-dependent $\ell \times \ell$ factorizations and $q \times (b - \ell) \times \ell$ batched corrections	SparseGPT: regular updates in a column loop; Block-OBS-GS: small kernels with gathers and scatters
Trailing propagation	Dense block error times a $b \times \lvert\mathcal{R}\rvert$ factor slice	$m \times b$ error times the same-shaped trailing factor slice	One large dense product per block in both methods
Refinement	No separate complete fixed-mask sweep	Ordered scalar steps; dense $n \times b$ by $b \times m$ residual updates	Sequential over chunks; one dense product per chunk

row-batched triangular GS. Rows in one batch share the system size $\ell_{i,r}$, not the matrix itself. Every row still evaluates Equation (5) with its own $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}$. For a fixed mask, row-batched triangular GS solves the same lower-triangular recurrence as the scalar-coordinate sweep; it does not replace GS by a simultaneous Jacobi update. The objective, score, inclusive threshold, block order, damping, Equation (5), and sweep count are unchanged. Grouping and batching target repeated indexing, status synchronization, and launches, but the time saved by each change is not separately isolated. Different floating-point reduction orders can change later activations and threshold decisions. Table 12 therefore tests end-to-end time and PPL agreement, not bitwise or fixed-support identity.

Detailed cost. The detailed operation counts confirm the class result of Section 4.2. For the J blocks and m rows, Cholesky factorizations of the $\ell_{i,r} \times \ell_{i,r}$ matrices $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}}$ cost $O(\sum_{i,r} \ell_{i,r}^3)$, and forming these matrices and the triangular solves add at most $O(m \cdot n \cdot b^2)$. The products with $(\boldsymbol{F}_{\mathcal{B},\mathcal{D}_\mathcal{B}})^\top \cdot \boldsymbol{F}_{\mathcal{B},\mathcal{B}}$ cost $O(m \cdot n \cdot b)$, and the trailing products with $\boldsymbol{F}_{\mathcal{B},\mathcal{R}}$ sum to $O(m \cdot n^2)$. Sorting costs $O\big(m \cdot n \cdot \log(m \cdot b)\big)$ for the $m \cdot b$ scores of each block, in addition to row indexing. When $K > 0$, initializing $\boldsymbol{R}$ costs $O(m \cdot n^2)$ once; the chunked residual updates cost $O(K \cdot m \cdot n^2)$, and the updates within chunks cost $O(K \cdot m \cdot n \cdot b)$. For transformer layers with comparable m, n and fixed b, K, sorting does not change the stated asymptotic class. Constant factors depend on the libraries and hardware and do not affect the class.

Workspace is not an inherent advantage over Thanos. Storing $\boldsymbol{H}$ or $\boldsymbol{F}$ requires $O(n^2)$ memory. Batched row-local systems for a row chunk of size q require $O(q \cdot b^2)$ workspace, in addition to weights and the $O(m \cdot n)$ dense residual. Solving instead of explicitly inverting a local system can reduce work and live temporaries, but both store matrices of size quadratic in b. This does not prove an order-of-growth memory advantage over every Thanos implementation. Activation residency, full-model placement, and concurrent tensors can dominate process memory.

Table 9 distinguishes shared dense products from row-specific indexing and solves. Our profiler timed only the stages listed in Table 3; it did not measure lower-level GPU costs, such as the number of kernel launches, memory bandwidth, or occupancy; hence Phase 1 cannot be split further.

D Full mechanism and downstream results

The full fixed-mask grid shows a GS gain in every cell but no uniform gain from the block solve alone. The block solve alone ($K = 0$) does not beat SparseGPT reconstruction in every cell: the 3B 62.5% and 87.5% cells and the 8B 87.5% cell provide counterexamples in this grid. GS reduces Q from its starting value in all 12 cells. The difference between this grid's 8B/75% $K = 1$ PPL of 32.1759 and the end-to-end result of 31.6882 is not rounding: the fixed-mask study keeps the recorded layer inputs, whereas the end-to-end run recomputes the inputs of later layers after earlier layers are reconstructed. The two families are therefore reported separately.

Table 10: PPL for all 12 model–sparsity cells with the mask fixed (lower is better).

Model	Sparsity	SparseGPT	$K = 0$	$K = 1$	$K = 2$	$K = 4$
3B	25%	8.0679	8.0657	8.0471	8.0440	8.0394
3B	37.5%	8.6852	8.6802	8.6075	8.5922	8.5823
3B	50%	10.6299	10.6044	10.2975	10.2510	10.2284
3B	62.5%	18.8784	18.9004	17.1603	16.9743	16.8993
3B	75%	63.3754	63.1194	52.6043	51.3353	50.6087
3B	87.5%	351.0129	374.6063	313.1780	305.0771	297.3557
8B	25%	6.4881	6.4871	6.4764	6.4741	6.4740
8B	37.5%	7.0150	7.0117	6.9693	6.9630	6.9600
8B	50%	8.2735	8.2652	8.1122	8.0881	8.0749
8B	62.5%	13.3615	13.3127	12.5278	12.4170	12.4042
8B	75%	36.8552	36.6588	32.1759	31.6902	31.4683
8B	87.5%	256.7083	257.6679	202.6030	194.2640	190.5517

Table 11: Llama-3.2-3B pruned to 50% sparsity, then quantized by GPTQ to four bits (group size 128).

3B configuration	Before W4 PPL	Reported PPL
Dense BF16	–	7.8166
GPTQ-W4 only	–	13.4737
SparseGPT + GPTQ-W4	10.6299	16.1311
Block-OBS-GS + GPTQ-W4	10.3057	15.3665

The second-mask experiment. The second-mask comparison isolates reconstruction on a different fixed support. The experiment of Section 5.3 uses blockwise magnitude–activation masks generated with the Wanda-style rule [Sun et al., 2024] and keeps the masks and $\boldsymbol{H}$ fixed within each model; hence only the reconstruction differs. Its PPL values are larger than those in Table 1 and are not mixed with them. The experiment does not optimize the mask and does not test learned masks.

Breadth and exceptions. Our results support no universal ranking. Earlier experiments on Qwen and Llama-2 checkpoints and with 2:4 masks contained exceptions; they used a different checkpoint and evaluator setup and are not combined with the results here. Within the present results, the regression of the block solve alone on the second mask (Section 5.3) and the mixed downstream scores (Table 2) show the same limit.

E Runtime profiles and large-model observations

The phase profile compares magnitudes; it does not attribute time exclusively. Table 3 reports synchronized totals per stage; computing $\boldsymbol{F}$ is timed separately from the per-row factorizations of Equation (5), which belong to Phase 1. The total pruning time includes calibration and reconstruction, not the subsequent PPL evaluation. The 435.927-second Phase-1 difference is 91.2% of the 478.051-second wall difference as a descriptive comparison of magnitudes. It is not an exclusive attribution: other stages change in both directions, and some work falls outside the timed stages.

E.1 Repeated 3B optimization measurements

The optimized implementation saves about one quarter of the original 3B pruning time, while remaining slower than SparseGPT. Table 12 compares the original Block-OBS-GS, optimized Block-OBS-GS, SparseGPT, and per-row Wanda on the same Llama-3.2-3B checkpoint and evaluator within each sparsity. The reductions are 23.60% at 25% sparsity and 25.94% at 50%, computed as $1 - t_{\text{optimized}}/t_{\text{original}}$ from method-wise medians. Speedup means $t_{\text{original}}/t_{\text{optimized}}$, not speedup over SparseGPT. The optimized/SparseGPT median ratios are 1.760 and 2.056.

Agreement is numerical and benchmark-specific, not exact identity. The optimized and original implementations agree in PPL, not in pruned weights. At 25% sparsity, optimized PPL is slightly worse: 8.052227 versus 8.049091 for the original, an absolute difference of 0.003136 (0.03896%

Table 12: Llama-3.2-3B pruning on one GPU: median and sample standard deviation (SD) over three fresh-process trials, excluding model loading and PPL evaluation.

3B sparsity	Method	PPL	Median (s)	SD (s)	Range (s)
25%	Original Block-OBS-GS	8.0491	388.11	8.56	383.86–400.34
25%	Optimized Block-OBS-GS	8.0522	296.50	3.57	290.35–296.55
25%	SparseGPT	8.0699	168.44	4.40	166.92–175.19
25%	Wanda	8.1426	47.78	1.19	46.59–48.97
50%	Original Block-OBS-GS	10.2814	459.93	2.29	457.68–462.25
50%	Optimized Block-OBS-GS	10.2765	340.65	0.86	339.40–341.05
50%	SparseGPT	10.6255	165.71	1.31	164.42–167.04
50%	Wanda	12.5192	43.36	0.47	43.05–43.98

Table 13: Achieved sparsity (%) and Compute Unified Device Architecture (CUDA) memory peaks (GiB) for the runs in Table 12.

3B sparsity	Method	Achieved (%)	Allocated GiB	Reserved GiB
25%	Original Block-OBS-GS	25.00023	7.4234	7.4922
25%	Optimized Block-OBS-GS	25.00023	7.4840	7.5508
25%	SparseGPT	25.00023	7.1328	7.4902
25%	Wanda	25.00000	6.9424	6.9980
50%	Original Block-OBS-GS	50.00024	7.4234	7.4941
50%	Optimized Block-OBS-GS	50.00024	7.4840	7.5527
50%	SparseGPT	50.00024	7.1328	7.4902
50%	Wanda	50.00000	6.9424	6.9980

relative). At 50%, optimized PPL is slightly better: 10.276547 versus 10.281396, an absolute difference of 0.004849 (0.04717% relative). Both comparisons meet our acceptance limits of 0.01 absolute PPL and 0.2% relative PPL difference, including each paired measured trial. These are acceptance tolerances, not statistical equivalence tests. Within each implementation, the three repetitions reproduce the same recorded masks and final tensor hashes. Across original and optimized implementations, however, only 7 of 196 prunable-tensor masks match exactly, and none of the 196 final tensor hashes matches, at either sparsity. Thus the end-to-end experiments do not hold the final support fixed. They support nearby WikiText-2 PPL for the tested checkpoint, calibration seed, and sparsities, not identical pruned models, guaranteed weightwise equivalence, or preservation of downstream accuracy.

The timing protocol separates repeatability from calibration robustness. Both suites share one protocol, which measures runtime repeatability at one seed. Both suites use BF16 model weights, FP32 statistics and solves, 128 calibration sequences, length 2,048, seed 0, damping fraction 0.01, and synchronous Compute Unified Device Architecture (CUDA) launches. Both Block variants use width 128 and one GS sweep; the reference row chunk is 4,096, and the optimized GS row chunk is 2,048, with explicit local-solve and triangular-workspace caps of 256 and 128 MiB. Each method receives one excluded whole-process warm-up. The measured orders are original/optimized/SparseGPT/Wanda, Wanda/SparseGPT/optimized/original, and SparseGPT/Wanda/original/optimized. Every measured trial starts a fresh process; hence process-local first-use costs remain inside the stated timing boundary. PPL is recomputed from 141 raw negative-log-likelihood (NLL) segments containing 288,627 predicted tokens. Three trials estimate runtime variation at one fixed calibration seed; they do not estimate seed variation or statistical significance of the PPL differences. The active benchmark uses GPU 0 with no recorded competing compute process on that device; the host's other physical GPUs are not pooled into this timing result.

The optimization does not show a pruning-memory advantage. The optimized implementation allocates slightly more memory than the original. Table 13 reports the achieved sparsities and maximum allocated/reserved peaks over the three measured trials, after the post-model-load allocator reset and within the pruning timer. These are allocator peaks that include the resident model and the temporary pruning arrays, not checkpoint sizes or serving memory. All timing samples, configurations, and raw NLL values are kept with the experiment records.

Table 14: Llama-3.3-70B-Instruct at 50% sparsity; one run on six GPUs.

70B / 50% method	Block width	PPL	Pruning (s)
dense	–	3.8623	–
SparseGPT	32	6.0800	4247.8
Thanos (solve-based)	32	6.1134	18200.3
Wanda	–	6.3981	1550.1
Block-OBS-GS	128	6.0389	4313.3

Table 15: Memory at 70B/50%: largest per-GPU CUDA peak over six GPUs (GiB).

70B / 50% method	Allocated (GiB)	Reserved (GiB)
SparseGPT	38.8598	40.7539
Wanda	34.2492	35.0625
Block-OBS-GS	40.8522	42.0410

E.2 Separate 70B reference observations

The 70B run is a single reference observation, not a scaling result. All runs used the same checkpoint and PPL evaluation, BF16 weights, and FP32 reconstruction, with synchronous execution on six GPUs. The time ratio of 1.015 between Block-OBS-GS and SparseGPT may reflect amortized fixed overhead, but one run cannot establish a scaling law. The table does not include the original Thanos code at larger block sizes or the optimized Block-OBS-GS at 70B.

An earlier 70B run at 75% sparsity, with a different execution configuration, was substantially slower for Block-OBS-GS; the 50% result here does not show that this gap is closed.

F Memory and inference measurement boundaries

Memory comparisons require the same model placement, precision, workload, device aggregation, and allocator reset. An earlier 70B comparison (70.2, 77.4, and 22.1 GiB) violated these conditions: the 22.1 GiB value for Block-OBS-GS covered one active pruning block while the rest of the model was offloaded to central processing unit (CPU) memory, whereas the baseline values used other residency, GPU aggregation, allocator resets, and measurement intervals. Its differences therefore reflect the measurement setup, not the algorithms, and Table 15 replaces it with matched measurements.

We infer no 70B/75% memory result from these 50% records, and no comparable 75% measurement is available. The 8B peak of 3.7355 GiB, measured during pruning only, is not directly comparable with Table 15, because its model-residency and reset boundaries differ. Reserved memory includes allocator caching and need not equal allocated tensor memory.

Table 16 reports actual CUDA CSR execution and its workload-dependent outcomes. The CSR conversion is exact up to small logit differences. All 224 pruned linear layers passed exact dense-roundtrip checks. A 16-token test gave complete top-1 agreement, with maximum absolute logit differences about 1.17×10^{-4} for SparseGPT and 5.20×10^{-5} for Block-OBS-GS. Prefill used five warm-ups and 20 measurements at each length 128, 512, and 2,048; fixed-token cached decode used an initial 512-token context, eight warm-up steps, and 32 measured steps. Each method ran dense before CSR in one process. There is no randomized outer-run order or uncertainty estimate across processes.

The matched FP32 payload reduction is 43.44%, and the peak allocated memory is reduced by 40.84%. The FP32 CSR payload is nevertheless 13.12% larger than BF16 dense. These byte counts are not serialized checkpoint sizes. Both methods thus yield usable sparse representations; the result is not a memory advantage of Block-OBS-GS.

Table 16: FP32 weights at 8B/75% in dense and CSR format: payload (GiB), decode (ms/token), and prefill (ms for 2,048 tokens).

Method	Dense GiB	CSR GiB	Dense decode	CSR decode	Dense prefill	CSR prefill
SparseGPT	29.915	16.920	26.520	20.654	548.7	19152.6
Block-OBS-GS	29.915	16.920	26.516	20.662	548.7	19154.5

G Experimental protocols and reproducibility boundaries

Checkpoints and preprocessing. The core experiments used validated ModelScope mirrors of Llama-3.2-3B and Meta-Llama-3.1-8B. The 70B experiment used Llama-3.3-70B-Instruct and is an instruction-tuned, not a base, checkpoint. Calibration and PPL used WikiText-2 with the recorded tokenizer and dataset versions, 128 calibration sequences, length 2,048, and seed zero. The model sources and tokenizer versions are kept with the experiment records. The Llama checkpoints [Grattafiori et al., 2024] are used under the Llama 3.1, Llama 3.2, and Llama 3.3 Community License Agreements, WikiText-2 [Merity et al., 2017] under the Creative Commons Attribution-ShareAlike (CC BY-SA) license, and lm-evaluation-harness [Gao et al., 2024] under the MIT license; all are used for research evaluation only.

Evaluator families. Two evaluator families are reported separately. The quality and mechanism results use an AutoTokenizer-based evaluator and its full-model PPL. The repeated timing runs and the 70B runs also record the token-level negative log likelihood over exactly 288,627 predicted tokens. They stay separate even when rounded PPL values are similar. The latter suites compute PPL as the exponential of total negative log likelihood divided by the number of predicted tokens; padding and an unpredicted first token are excluded from that denominator.

Timing. Each timing family has its own measurement boundary. The 8B profile uses one synchronized measurement of the total pruning time per method on an NVIDIA RTX PRO 6000 Blackwell Server Edition GPU. The separate 70B runs use six such GPUs and synchronous launch settings. Pruning time includes calibration and reconstruction and excludes final PPL evaluation. Neither family estimates between-run variability. The distinct 3B suites in Appendix E.1 provide three fresh-process measurements per method at each sparsity; Table 12 reports their range and sample standard deviation. Their numerical acceptance concerns WikiText-2 PPL, not exact masks or final weights. BF16 weights and FP32 reconstruction remain distinct from the FP32 serving control.

Software and downstream evaluation. The repeated 3B suites, 8B profile, and CSR study record Python 3.12.3, PyTorch 2.8.0+cu128, CUDA runtime 12.8, Transformers 4.46.3, and Tokenizers 0.20.3. Downstream evaluation uses lm-evaluation-harness 0.4.11 [Gao et al., 2024]. Avg7 is the unweighted mean over the Challenge and Easy sets of the AI2 Reasoning Challenge (ARC), BoolQ, HellaSwag, OpenBookQA, Physical Interaction Question Answering (PIQA), and WinoGrande, using normalized accuracy where the recorded task supplies it. The dense 8B reference is 71.4616% Avg7 and 65.3112% five-shot MMLU; MMLU uses the sample-weighted aggregate over its 57 subjects.

The GPTQ pipeline uses a separate quantization environment: GPTQModel 7.3.1, Transformers 5.6.0, and PyTorch 2.8.0+cu128. Its W4 configuration uses group size 128, symmetric quantization, no activation ordering, sequential quantization, damping fraction 0.01, and batch size one. Calibration uses 128 sequences of length 2,048 and seed zero. The pre-W4 values measure the exported pruned checkpoints before quantization; the four final endpoints use the quantization evaluator.

Mechanism controls. Fixed-mask ablations use the same starting weights, masks, and $\boldsymbol{H}$ within each cell, and their times cover only the reconstruction. End-to-end runs instead let later inputs and masks depend on earlier reconstructed layers. This difference explains why fixed-mask and end-to-end PPL values differ (Appendix D).

Reproducibility. Appendices B–G document the assumptions, implementation, precision, calibration, model placement, and timing protocol, and all tables use archived experiment records. Comparisons reported from a single run carry no between-run uncertainty estimate.

Broader impact. Pruning lowers the storage and computation needed to deploy existing language models. It does not remove their biases or unsafe behavior, and a lower perplexity does not establish safety. The evaluation uses existing benchmarks and no newly collected personal data.

Limitations of the evidence. The experiments are concentrated on Llama checkpoints, one calibration seed, and WikiText-2, with selected downstream tests. They do not establish calibration-seed robustness, learned-mask transfer, optimized-runtime scaling, or production-serving performance. Matched end-to-end timing across all GS sweep counts and a validated predictor of their PPL benefit are also unavailable. The 70B memory comparison omits a matched Thanos peak and a comparable 75% study; event-boundary host samples do not establish a host lifetime peak or a simultaneous aggregate GPU peak. The serving study measures parameter buffers, not serialized checkpoint size, and does not test a multi-request workload.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction distinguish conditional optimality, fixed-mask guarantees, and empirical results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Conclusion (Section 7) and Appendix G discuss model coverage, calibration, downstream accuracy, runtime, and deployment.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Section 4 states assumptions; Appendix B provides the derivations.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix G records calibration, precision, evaluator, and measurement boundaries; the algorithm and proofs specify the method.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: A public archival release of all experiment code and records has not been established.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 5.1 and Appendix G describe the experimental settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The 3B timing suites report three-trial ranges and sample standard deviations at a fixed calibration seed. Other quality and 70B comparisons remain single-run; no confidence intervals or significance claim is made for the PPL differences.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix E identifies single-GPU profiling and six-GPU execution; the serving protocol is specified in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics. The work uses public models and datasets under their licenses, involves no human subjects or new personal data, and releases no new model or dataset.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader-impact paragraph in Appendix G describes both the deployment benefits of compression and the persistence of model biases and unsafe behavior.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No new generative model checkpoint or unrestricted service is released.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Appendix G credits and cites all assets and states their licenses: the Llama 3.1, 3.2, and 3.3 Community License Agreements for the checkpoints, CC BY-SA for WikiText-2, and the MIT license for lm-evaluation-harness.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new dataset or model checkpoint is introduced by this manuscript package.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or newly collected human-subject data is used.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human-subject experiment is performed.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The AI-use statement discloses manuscript, analysis, and tooling assistance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.