k-SUPPORT PRUNING: TIGHT CONVEX RELAXATION AND PRINCIPLED OPTIMIZATION FOR N:M LLM PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Semi-structured (N:M) sparsity — keeping N of every M weights, the form that supported sparse kernels can accelerate — is the dominant hardware-realizable way to prune large language models. Choosing *which* N survive is combinatorial, and current methods approximate that choice rather than characterize it: one-shot pruners commit to a ranking fixed before any optimization, and learnable-mask methods descend a nonconvex surrogate at the cost of gradients through the model. We show that the choice has a convex geometry of its own. Per block, the convex hull of the N-sparse unit-ℓ_2 ball is exactly the unit ball of the k-support norm at $k = N$: the tight per-block relaxation, and, among the classical structured penalties we consider, the one that carries the budget N itself — for $N \geq 2$ the group and exclusive lasso do not. The relaxation is constructive. Its exact closed-form proximal operator is governed by one *water level* per block, found in a single sorted pass: what stands above it survives, shrunk, and the rest is exactly zero, with a least-squares refit then restoring the survivors. It turns the layerwise pruning objective into a convex program solved in the one-shot post-training setting, and certifies per block that rounding the relaxed solution back to N:M costs no additional reconstruction loss. Magnitude and activation-scaled saliency reappear as its zero-iteration gates, and under diagonal curvature with a separation condition, hardening and refitting it returns the combinatorial optimum. Read across matrices rather than within a block, that same water level prices the sparsity budget, so one object decides both which weights to keep and *where* sparsity is spent. Across seven models from four architecture families, KSP takes the best accuracy in six of seven columns under $2{:}4$ and eight of ten cells on each metric at 75% sparsity; on a 70B model, allocating the budget beats uniform N:M at matched measured latency, and pricing it by latency adds further gains.

1 INTRODUCTION

Large language models (LLMs) have advanced largely through scale (Brown et al., 2020; Touvron et al., 2023a; Kaplan et al., 2020), and for a widely deployed model the lifetime cost is dominated by inference. In memory-bound serving that cost is set by moving weights rather than by arithmetic, so storing fewer weights is a direct lever on it; since retraining after pruning is out of reach for most practitioners, the dominant paradigm is *one-shot post-training pruning*. Among sparsity patterns, semi-structured N:M sparsity — at most N nonzeros in every block of M consecutive weights — is the one with first-class hardware support, $2{:}4$ mapping onto sparse tensor cores (Mishra et al., 2021). The problem is concrete: choose, for every block of every weight matrix, which N of M weights to keep, and what the survivors should be.

Three families solve it well enough to be deployed. Saliency pruners rank weights by a closed-form score, typically weight magnitude scaled by activation norms, and keep the top N per block (Sun et al., 2024; Zhang et al., 2024a). Solvers on the layerwise objective (Frantar & Alistarh, 2023; Meng et al., 2024) update the survivors as well, either by sequential elimination with compensation or by alternating a regularized solve with a projection onto the nonconvex N:M set. A third family *learns* the mask through the model by gradient descent on a relaxed surrogate (Fang et al., 2024; Sun et al., 2026). Together they made 50%-sparse LLMs practical on a single GPU (App. A).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

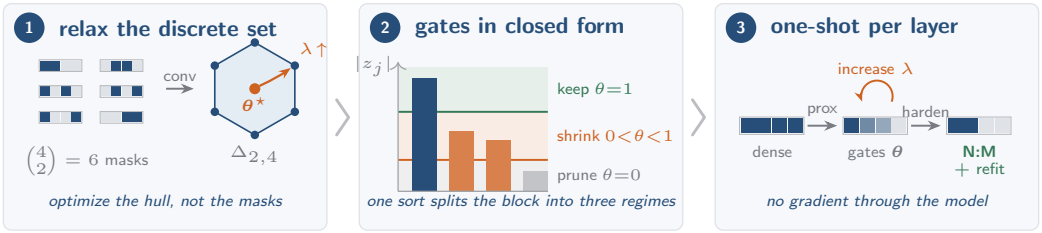


Figure 1: **KSP at a glance.** A block of M weights keeps at most N. (1) Rather than search the $\binom{M}{N}$ full-budget masks, KSP optimizes over the mask polytope $\Delta_{N,M}$, whose maximal-budget vertices are exactly those masks. (2) One sort per block splits its gates into full, fractional and zero in closed form (Thm. 1). (3) One pass per layer rounds the gates to an admissible mask and refits the survivors. Read across matrices, the same construction prices the budget (§4). Symbols: §3.

What none has is a characterization of the selection problem being solved: a fixed ranking cannot say what it is optimal for, a greedy or nonconvex-projection solver inherits the guarantees of its heuristic rather than of the problem, and a learned mask descends a surrogate whose stationary points are not described. The natural remedy — relax the combinatorial constraint convexly — has been unavailable, because the penalties inherited from structured sparsity relax the *wrong* discrete set: none carries the cardinality N that the hardware fixes (§3). What that costs is easiest to see beyond 2:4, where saliency rankings collapse by orders of magnitude. Nor does any of the three price a unit of sparsity, so all spend the same pattern on every matrix, while the speedup a sparse kernel delivers is strongly shape- and phase-dependent.

Our starting point is that the N:M constraint is not an arbitrary structured-sparsity pattern. Inside each hardware block it is the convex-hull geometry of an N-sparse ℓ_2 ball, and that geometry is exactly the *k-support norm* (Argyriou et al., 2012) — an object that has not, to our knowledge, been brought to bear on semi-structured pruning. Everything else follows, and Fig. 1 traces the chain. The norm admits a variational representation by multiplicative gates on the mask polytope, whose maximal-budget vertices are exactly the N:M masks; the optimal gates have a closed-form water-filling solution, and the proximal operator of the squared norm is closed-form and parallel over blocks, so the relaxed layerwise objective becomes a convex program whose reference solver costs one matrix product per iteration. The geometry then pays for itself three times: it certifies per block when rounding to N:M costs nothing, it recovers magnitude and activation-scaled saliency as its own zero-iteration limit, and — because the squared water level producing the gates is, read across matrices, the marginal price of a unit of budget in the penalty — it turns allocation into a second convex program with a closed-form optimality condition (§4). One geometry answers both which weights to keep and where sparsity should be spent, and it lets us separate four levers practice conflates — selector, search, compensation and allocation — which our controlled experiments then size against one another.

Contributions. *(i) A cardinality-correct convex relaxation.* The per-block k-support norm is the tight convex-gauge relaxation of the N:M constraint under the ℓ_2 scale (Prop. 1), and its square is our regularizer. *(ii) A constructive framework.* The relaxation admits closed-form gates and an exact weighted prox (Thm. 1), giving a deterministic one-shot solver with no backpropagation through the model, together with a lossless-hardening certificate (Cor. 1), exact recovery under diagonal curvature (Prop. 2), and magnitude and activation-scaled saliency as zero-iteration gates (Prop. 6). *(iii) One object for selection and allocation.* The norm at fractional budget (§4) supplies a closed-form price for budget — the squared water level — and an equalized-price characterization of the optimal allocation across matrices (Thm. 2), making the two decisions consequences of one object rather than of two procedures.

2 Related Work

Three literatures on N:M pruning meet here, and one on convex sparsity penalties; in each, the ingredient we replace is a heuristic surrogate for the underlying selection problem. **One-shot LLM pruners** choose or refine a support by ranking (Sun et al., 2024; Zhang et al., 2024a;b), by greedy

elimination (Frantar & Alistarh, 2023) or by projection onto the nonconvex N:M set (Meng et al., 2024), and inherit whatever guarantee the rule carries. We replace it with a convex program tight for the constraint the hardware imposes, which lets us say what a support is optimal for, certify when rounding it costs nothing, and recover the ranking rules as the zero-iteration case (Prop. 6) rather than as rivals; sharpening the score (Yang et al., 2025) or preconditioning the weights first (Gu et al., 2025; Zou et al., 2025) changes what a selector is handed, not how it chooses, and composes with ours. **Learnable-mask methods** (Fang et al., 2024; Sun et al., 2026; Sun & Sakuma, 2026; Liu et al., 2025) also relax the mask polytope, but through smoothings chosen so that gradients flow, or exact penalties carrying critical-point guarantees only, so the mask is learned by gradient descent, end to end or layer by layer, over GPU-hours to days. Ours is the tight convex gauge of the constraint set itself, with a closed-form per-block prox in one post-training pass, and tight for every N:M rather than for 2:4 alone. **Non-uniform sparsity budgets** (Yin et al., 2024; Chen et al., 2025; Yang et al., 2026; Lu et al., 2025; Zhao et al., 2025) ask the question of §4 — how much budget each matrix should get — but each invents or learns a proxy for the value of budget and then tunes it. We introduce no separate proxy: the squared water level of each block's solution *is* that value in the penalty. Among **structured-sparsity penalties** (Yuan & Lin, 2006; Zhou et al., 2010; Lin et al., 2023; Tibshirani & Taylor, 2011; Fei et al., 2023; Bian et al., 2024; Fei et al., 2025), none relaxes the right combinatorial set for $N \geq 2$ (§3) while the k-support norm (Argyriou et al., 2012) does. What is new here is the identification — that N:M hardware sparsity has precisely this convex geometry — and what one object then yields: a closed-form gate, an exact prox, a hardening certificate, exact recovery in a tractable regime, and a price for the budget. To our knowledge no prior N:M method derives selection and allocation from one relaxation. App. A treats each literature in full.

3 The Blockwise k-Support Relaxation, and What It Gives

We fix the objective that one-shot pruners share — so that every comparison in this paper is one of mechanism alone — then show that the N:M constraint attached to it has a convex geometry of its own, and turn that geometry into a solver with guarantees.

3.1 Setting, and a cardinality-correct relaxation

We prune a frozen pre-trained LLM one layer at a time. For a linear layer with dense weights $\boldsymbol{W}_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ (d_{out} output and d_{in} input channels) and n_{cal} rows of calibration input $\boldsymbol{X} \in \mathbb{R}^{n_{\text{cal}} \times d_{\text{in}}}$, a pruned $\boldsymbol{W}$ is good to the extent that it reproduces the layer's outputs on that data. Writing $\boldsymbol{H} := \boldsymbol{X}^\top \boldsymbol{X}$, that reconstruction error is a quadratic in the weight perturbation,

$$f(\boldsymbol{W}) = \tfrac{1}{2} \operatorname{tr}\big[(\boldsymbol{W} - \boldsymbol{W}_0)\boldsymbol{H}(\boldsymbol{W} - \boldsymbol{W}_0)^\top\big], \tag{1}$$

so the data enters only through the $d_{\text{in}} \times d_{\text{in}}$ Gram matrix, accumulated in one streaming pass and damped as in Frantar & Alistarh (2023) so that f is strongly convex. Fixing a block size M and a budget $1 \leq N \leq M$ (with $N < M$ in every uniform pattern) and partitioning each row i into $B = d_{\text{in}}/M$ consecutive blocks $\{g_b\}$, with entries $\boldsymbol{w}_{i,g_b}$, the N:M feasible set constrains each block,

$$\mathcal{C}_{N:M} \;=\; \{\boldsymbol{W} : \|\boldsymbol{w}_{i,g_b}\|_0 \leq N \;\; \forall i, b\}. \tag{2}$$

The per-block quota is what makes N:M harder than a global budget and what the sparse tensor core requires: minimizing (1) over (2) is combinatorial, and the blocks couple through the off-diagonal entries of $\boldsymbol{H}$. The relaxation needs the at-most-N set; every reported mask keeps exactly N per block.

Relaxing that constraint convexly is where the classical structured penalties fall short, for a precise reason: they do not carry the cardinality N. Group lasso drives whole blocks to zero, the opposite of what N:M needs; the exclusive lasso promotes competition inside a block, which looks right, but its gauge $(\sum_b \|\boldsymbol{w}_{i,g_b}\|_1^2)^{1/2}$ is provably the tightest convex relaxation of *at most one nonzero per block* (Obozinski & Bach, 2012, §4.2) — of 1:M, not N:M (App. A, Tab. 2). What does carry N is the k-support norm. For $\boldsymbol{v} \in \mathbb{R}^M$ and $k = N$, written $\|\cdot\|_{(N)}$, it is the gauge of $K_N := \operatorname{conv}\{\boldsymbol{v} : \|\boldsymbol{v}\|_0 \leq N,\ \|\boldsymbol{v}\|_2 \leq 1\}$ (Argyriou et al., 2012); equivalently the latent overlapping group lasso over all $\binom{M}{N}$ size-N groups (Jacob et al., 2009). We propose

$$\Omega^{\boldsymbol{\beta}}_{N:M}(\boldsymbol{W}) = \sum_{i=1}^{d_{\text{out}}} \sum_{b=1}^{B} \big\| \sqrt{\boldsymbol{\beta}_{g_b}} \circ \boldsymbol{w}_{i,g_b} \big\|^2_{(N)}, \qquad \min_{\boldsymbol{W}} \; F_\lambda(\boldsymbol{W}) := f(\boldsymbol{W}) + \lambda\, \Omega^{\boldsymbol{\beta}}_{N:M}(\boldsymbol{W}), \tag{3}$$

with penalty parameter $\lambda > 0$, entrywise product $\circ$, and per-coordinate penalty weights $\boldsymbol{\beta} > 0$ (entries β_j, entrywise square root) fixed at $\beta_j \propto H_{jj}$ throughout, for the reason given in §3.2.

Proposition 1 (Per-block convex-hull tightness)**.** *Per block, the unit ball of $\|\cdot\|_{(N)}$ equals K_N, and any convex function lower-bounding the combinatorial gauge of the N-sparse ball is dominated by $\|\cdot\|_{(N)}$: it is the tight convex-gauge relaxation of the blockwise cardinality constraint under the ℓ_2 scale. (App. B.1, with the product-of-hulls statement.)*

The penalty is therefore built from the tightest gauge per block, and one integer covers the family — $N = 1$ gives ℓ_1 per block, $N = M$ gives ℓ_2 — which is what lets §4 interpolate between patterns.

3.2 Closed-form gates and an exact proximal operator

What makes the relaxation cheap at LLM scale is that its two working primitives are closed-form. Let $\Delta_{N,M} := \{\boldsymbol{\theta} \in [0,1]^M : \sum_j \theta_j \le N\}$ be the *mask polytope*, whose extreme points are the binary vectors with at most N ones and whose maximal-budget vertices are the full-budget N:M masks.

Theorem 1 (Gates and the weighted prox)**.** (i) Variational identity. *For every $\boldsymbol{v} \in \mathbb{R}^M$ (conventions $0/0 := 0$, $t/0 := \infty$ for $t > 0$),*

$$\|\boldsymbol{v}\|_{(N)}^2 \;=\; \min_{\boldsymbol{\theta} \in \Delta_{N,M}} \sum_{j=1}^{M} \frac{v_j^2}{\theta_j}, \tag{4}$$

minimized at $\theta_j^\star = \mathbf{1}\{v_j \neq 0\}$ when $\|\boldsymbol{v}\|_0 \le N$, and otherwise at the water-filling gate

$$\theta_j^\star = \min\big(|v_j|/\tau,\, 1\big), \quad \text{water level } \tau\text{: the root of } \textstyle\sum_j \min(|v_j|/\tau, 1) = N, \tag{5}$$

unique on $\operatorname{supp}(\boldsymbol{v})$ and computable in $O(M \log M)$ by one sort. (ii) Weighted prox. *For a prox input $\boldsymbol{z} \in \mathbb{R}^M$, loss weights $a_j > 0$ and penalty weights $\beta_j > 0$, write $c_j := 2\lambda\beta_j/a_j$ for the effective shrinkage and $\rho_j := a_j|z_j|/\sqrt{2\lambda\beta_j}$ for the* selection statistic. *Then $\min_{\boldsymbol{v}} \sum_j \frac{a_j}{2}(v_j - z_j)^2 + \lambda \left\|\sqrt{\boldsymbol{\beta}} \circ \boldsymbol{v}\right\|_{(N)}^2$ has the unique solution*

$$v_j^\star = \frac{\theta_j^\star}{\theta_j^\star + c_j}\, z_j, \qquad \theta_j^\star = \operatorname{clip}_{[0,1]}\Big(\frac{\sqrt{a_j c_j}\,|z_j|}{\tau} - c_j\Big), \tag{6}$$

where $\theta_j^\star = \mathbf{1}\{z_j \neq 0\}$ if $\|\boldsymbol{z}\|_0 \le N$, and otherwise τ solves $\sum_j \theta_j^\star(\tau) = N$ by a sorted scan of $2M$ breakpoints in $O(M \log M)$. The operator produces exact zeros*: in the binding case ($\|\boldsymbol{z}\|_0 > N$) $v_j^\star = 0$ iff $\rho_j \le \tau$, so selection is a threshold on ρ_j and signs are preserved. (App. B.1, App. B.2.1.)*

The prox costs $O(M \log M)$ per block and is parallel across the millions of blocks in a layer, so the per-iteration cost is set by the gradient matrix product alone. Read at the loss curvature $a_j = H_{jj}$, with $s_j := H_{jj}^{1/2}$, the statistic ρ_j ranks coordinates by $s_j|z_j|$ — the ranking the ℓ_0-optimal block solution uses — exactly under *matched scaling* $\beta_j \propto H_{jj}$, the unique weighting up to scale under the conditions of Prop. 5. It is the scaling we use everywhere; App. B.2 separates it from the isotropic majorization the solver's inner step uses.

The per-layer solver. KSP is one-shot post-training pruning in the standard sense: no retraining, no backpropagation through the LLM, each layer solved from calibration statistics. The loop is a short continuation path over (3) along an increasing schedule $\lambda_0 < \lambda_1 < \cdots$, stopped when the per-block supports freeze, followed by *hardening* (keeping each block's top-N gates) and a *refit* — the masked conjugate-gradient (CG) re-solve (Hestenes & Stiefel, 1952) of the survivors (Alg. 1). What the formulation *fixes* is the objective, the penalty weights, the prox and the refit; what a solver *chooses* is the step between prox evaluations. We use two: the proximal gradient step of Alg. 1 for the reduced-budget controls, and for all reported KSP numbers a *preconditioned* splitting step — one cached ridge solve in the loss's own metric alternating with the same prox, which at our iteration budgets reaches the better supports (App. C). Both drive the same strongly convex F_λ, with one minimizer, through the same prox, so they differ in cost and in the iterates they return, not in what they optimize.

Evaluating the gates once at $\boldsymbol{W}_0$, before any descent, already returns something familiar: under matched scaling, hardening the per-block top-N of $s_j|(\boldsymbol{W}_0)_{ij}|$ is Wanda's N:M mask (Sun et al., 2024) up to the Gram damping, and with all $s_j = 1$ it is blockwise magnitude pruning (Prop. 6, App. B.3). The activation-scaled saliency rules are therefore not competitors standing beside the formulation but its cheapest instance. That also separates two decisions the literature conflates: an N:M method is a *selection* — a map $\mathcal{A}$ from $(\boldsymbol{W}_0, \boldsymbol{H})$ to a binary mask $\boldsymbol{\Theta}^\star \in \{0,1\}^{d_{\text{out}} \times d_{\text{in}}}$ — followed by a *compensation* $\bar{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}:\, \text{supp}(\boldsymbol{W}) \subseteq \text{supp}(\boldsymbol{\Theta}^\star)} f(\boldsymbol{W})$. Saliency rules supply $\mathcal{A}$ and skip the compensation; KSP makes $\mathcal{A}$ a convex program and solves the compensation. The prox returns shrunken survivors, which the refit restores, so after it the support is the only remaining degree of freedom — which makes the mask swap of §5.2 a decisive test.

3.3 What the geometry guarantees

Three objects have to be kept apart: the convex optimum $\widehat{\boldsymbol{W}}$ of F_λ, which still carries shrinkage; the mask $\boldsymbol{\Theta}^\star$ obtained by hardening its gates; and the refit weights $\bar{\boldsymbol{W}}$ on its support. In one regime the chain is exact end to end, all three coinciding with the combinatorial optimum $\arg\min_{\mathcal{C}_{N:M}} f$: the diagonal geometry underlying magnitude and activation-scaled saliency, where blocks decouple and the relaxed and combinatorial solutions rank coordinates alike. Throughout, $(s|z|)_{(r)}$ and $\rho_{(r)}$ denote the r-th largest of $s_j|z_j|$ and of ρ_j in a block, the $(M+1)$-th being 0; a parenthesized subscript on a single bar is an order statistic, on a double bar the k-support norm.

Proposition 2 (Diagonal-curvature support recovery)**.** *Let $\boldsymbol{H} = \text{diag}(s_1^2, \ldots, s_{d_{\text{in}}}^2)$ with all $s_j > 0$, and run KSP with matched scaling; because $\boldsymbol{H}$ is diagonal, each block's $\boldsymbol{z}$ is that block of $\boldsymbol{W}_0$. If the separation condition $(s|z|)_{(N)} > (s|z|)_{(N+1)}$ holds in every block, then $\min_{\mathcal{C}_{N:M}} f$ has a unique optimal support — per-block top-N by $s_j|z_j|$ — the mask hardened from the minimizer of F_λ equals it for* every *$\lambda > 0$, and refitting that mask gives the combinatorial N:M optimum up to the CG tolerance. (App. B.3.)*

In that regime the relaxation *finds* the combinatorial optimum. On dense Grams we certify, block by block, that hardening — where a convex method usually loses its guarantees — costs nothing.

Corollary 1 (Lossless-hardening certificate)**.** *Assume the budget binds and $c_j \equiv c$ across the block, which holds in the plain case ($a_j = \beta_j = 1$) and under matched scaling in the loss metric ($a_j = \beta_j = s_j^2$). Then the prox of Thm. 1 returns an integral gate vector $\boldsymbol{\theta}^\star \in \{0,1\}^M$ iff*

$$\rho_{(N)} \big/ \rho_{(N+1)} \;\geq\; 1 + \tfrac{1}{c}, \tag{7}$$

with $c = 2\lambda$ in both named cases ($2\lambda\zeta$ for $\beta_j = \zeta s_j^2$), where $\rho_j \propto |z_j|$ in the plain case and $\rho_j \propto s_j|z_j|$ under matched scaling; the surviving support is the top-N by ρ_j, and hardening that block is lossless*. For dense $\boldsymbol{H}$, where c_j vary, it evaluates the general form of the test (App. B.2).*

The test costs one sort per block; elsewhere the projection error is controlled by the gate non-integrality, and the refit never increases the layerwise loss relative to naive projection (App. B.2.1).

4 From Local Selection to Global Allocation

Everything so far has fixed the pattern (N, M) and asked which weights survive inside each block. The same convex object answers a question one scale up — how much budget each matrix should receive — because the water level of §3.2, read across matrices rather than within a block, has as its square the exact *marginal value of budget* in the penalty of (3). Equalizing marginal value per unit cost is the classical condition for allocating a scarce resource, and the geometry delivers it for the relaxed objective. The target is quality at fixed hardware latency: *mixed* N:M lets N vary across a partition of the weights into *tiles* under a global budget, a Pareto move on the speed–accuracy frontier (App. D, Fig. 7). Nothing in Thm. 1 or Cor. 1 uses uniformity of N, so the penalty, prox, solver and certificate apply to any allocation unchanged.

The water-filling (5) never uses integrality of N. For real $n \in (0, M]$ the *fractional k-support norm* $\|\boldsymbol{v}\|_{(n)}$ is (4) with the budget relaxed to n; it is a norm, computed by the same water-filling, and its value function $V_{\boldsymbol{v}}(n) := \|\boldsymbol{v}\|_{(n)}^2$ is convex and nonincreasing with, on the binding range $n < \|\boldsymbol{v}\|_0$,

$$V'_{\boldsymbol{v}}(n) \;=\; -\tau(n)^2 \tag{8}$$

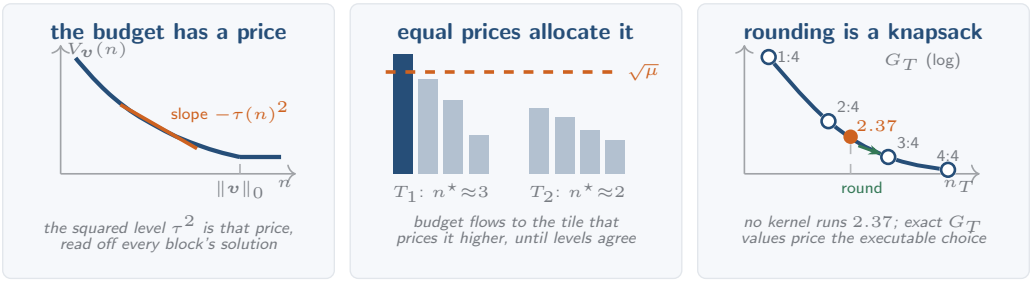


Figure 2: **The water level prices the sparsity budget of the relaxed objective** (3); symbols are defined in this section. *Left:* $V_{\boldsymbol{v}}(n)$ is convex and nonincreasing with slope $-\tau(n)^2$ on the binding range, so the squared water level is the marginal price of one more nonzero. *Middle:* the optimum equalizes the block-averaged squared water level across tiles, so the higher-energy tile T_1 takes more budget than the flatter T_2; at *equal* energy an outlier-dominated tile attracts *less* than a flat one, in contrast to heuristics that protect outlier-heavy layers. *Right:* a multiple-choice knapsack selects executable patterns, with the closed-form G_T values as coefficients (App. D.2).

(Prop. 8): the marginal value of one more nonzero in a block, in the penalty, is the square of its water level, a level the block prox computes up to a factor $\sqrt{2\lambda}$ (Fig. 2, left). Hardware constrains mixing to a set $\mathcal{T}$ of *tiles* — groups of blocks that must share one pattern — each declaring one budget $n_T \geq n_{\min}$, the smallest executable per-block budget. With encoding cost γ_T per unit of budget in tile T, global budget κ, and $\boldsymbol{n} = (n_T)_{T\in\mathcal{T}}$, the allocation problem for fixed weights is

$$\min_{\boldsymbol{n}\in[n_{\min},M]^{|\mathcal{T}|}} \sum_T G_T(n_T), \qquad G_T(n) := \sum_{b\in T} V_{\boldsymbol{v}_b}(n), \qquad \text{s.t. } \textstyle\sum_T \gamma_T\, n_T \leq \kappa, \tag{9}$$

a convex separable program, with $b = (i, g)$ ranging over the tile's row blocks and each $V_{\boldsymbol{v}_b}$ evaluated on the *scaled* block $\boldsymbol{v}_b = \sqrt{\boldsymbol{\beta}_g} \circ \boldsymbol{w}_{i,g}$, so water levels are commensurable across matrices of different curvature. The full mixed program over weights, gates and budgets is convex (App. D).

Theorem 2 (Water-level allocation)**.** *A feasible budget vector $\boldsymbol{n}^\star$ solves* (9) *iff there exists a price $\mu \geq 0$ for the global budget, with complementary slackness, such that for every tile*

$$\sum_{b\in T} \tau_b(n_T^\star)^2 \;=\; \mu\,\gamma_T \quad \textit{for interior } n_T^\star \qquad (\leq \textit{at the lower box; } \geq \textit{its left limit at the upper box}),$$

where $\tau_b(n)$ is the water level of block b at budget n ($:= 0$ for $n \geq \|\boldsymbol{v}_b\|_0$), with the subdifferential inclusion $\mu\gamma_T \in -\partial G_T(n_T^\star)$ at the finitely many kinks above $n_{\min}$. When γ_T is the number B_T of blocks in the tile, the block-averaged squared water level is constant across tiles with interior, non-kink budgets: mixed allocation is a water-filling over water-fillings. (App. D.)

The condition is a *pricing* statement: each tile reports one number, its summed squared water level per unit cost, and the allocator equalizes it across tiles — the number depth ramps, outlier ratios and fixed per-layer schedules guess (Fig. 2, middle). Solving (9) costs one scalar minimization per tile inside a scalar dual, with no solver reruns, and a multiple-choice knapsack selects executable patterns, with the closed-form G_T as coefficients (Fig. 2, right). As the water level is a property of the weights, not of the schedule, the allocation is cheap and transfers across scales.

5 Experiments

One-shot N:M pruning makes four decisions that the literature usually conflates, and the framework of §3–§4 separates them: the *selector* that chooses a support, the *search* used to make that choice, the *compensation* applied to the survivors, and the *allocation* of budget across matrices. They are orthogonal experimental axes — any selector runs under any allocation, and any mask under any compensation — so each can be varied with the others held fixed, which is the design of everything below. We first place KSP against the field under fixed patterns (§5.1), then size the four decisions against one another (§5.2), then hold measured hardware latency fixed and ask what allocation is worth, and what it preserves on benchmarks a perplexity number cannot see (§5.3).

Table 1: **Uniform** 2:4 **(50% sparsity).** PPL is WikiText-2 perplexity (↓), Acc the seven-task zero-shot mean (↑); best pruned entry per column in **bold**, second underlined. † seed means (counts and exceptions in Tab. 5); ‡ the authors' released checkpoints at Llama-2-7B, Mistral-7B and Llama-2-13B, their code run by us elsewhere; — not run. KSP is at the reported configuration of App. C, tuned with ALPS by one pre-registered rule on a slice disjoint from every reported column.

	Llama-3.2-1B		Llama-3.2-3B		Llama-2-7B		Qwen2.5-7B		Mistral-7B		Llama-2-13B		Llama-2-70B	
Method	PPL	Acc	PPL	Acc	PPL	Acc	PPL	Acc	PPL	Acc	PPL	Acc	PPL	Acc
Dense	9.75	50.70	7.81	57.42	5.47	60.17	6.85	65.80	5.32	64.87	4.88	63.19	3.32	66.53
***saliency**: a ranking fixed before any optimization*														
Magnitude	2919	32.95	387.5	35.26	37.29	47.25	2467	40.42	13.08	50.33	8.86	49.06	6.76	58.39
Wanda	79.49	37.78	35.35	43.15	12.05	48.84	14.84	56.07	11.95	50.79	8.99	54.05	5.47	62.87
RIA	88.85	37.88	35.19	43.13	11.30	48.34	13.84	57.43	11.10	52.17	8.41	54.71	**5.36**	62.45
DSnoT	83.75	37.20	31.37	43.46	11.96	48.84	14.00	56.50	12.46	50.72	8.84	54.08	5.49	62.34
Pruner-Zero	106.0	37.13	35.98	43.72	11.25	47.59	12.36	57.74	9.49	51.90	8.03	53.74	—	—
***learned masks**: gradients through the model*														
MaskPro	79.32	37.85	35.30	43.28	10.73	49.63	13.38	56.56	11.75	51.17	8.61	54.37	—	—
ProxSparse‡	67.75	37.76	25.80	43.20	**9.17**	51.44	15.50	54.59	9.35	54.00	**7.10**	56.04	—	—
***solvers on the same layerwise objective**: the comparison that isolates the formulation*														
SparseGPT†	33.99	41.15	21.11	47.07	11.06	50.70	11.29	58.78	10.34	54.19	8.79	56.19	5.59	63.69
ALPS†	26.50	42.47	18.20	49.02	10.09	51.98	10.58	57.93	9.63	54.81	8.21	**56.70**	—	—
KSP† (ours)	**26.34**	**42.49**	**17.93**	**50.26**	9.70	**52.03**	**10.17**	**59.65**	**9.32**	**55.04**	7.99	56.55	5.37	**64.94**

Setup. We prune seven models from four architecture families (Touvron et al., 2023b; Grattafiori et al., 2024; Yang et al., 2024; Jiang et al., 2023) spanning a 70× size range, plus Llama-3.1-70B for the deployment check, against magnitude, Wanda (Sun et al., 2024), RIA (Zhang et al., 2024a), DSnoT (Zhang et al., 2024b), Pruner-Zero (Dong et al., 2024), SparseGPT (Frantar & Alistarh, 2023) and ALPS (Meng et al., 2024), together with MaskPro (Sun et al., 2026) (re-run) and ProxSparse (Liu et al., 2025), and OWL (Yin et al., 2024) and a random dense-selection control for the allocation experiments. Calibration is C4 (Raffel et al., 2020), matched across one-shot methods in every comparison. We report WikiText-2 perplexity (Merity et al., 2017) and mean zero-shot accuracy over seven tasks (Gao et al., 2023), adding PTB and C4 on the corpus-level comparisons, and measured latency. Every run, including the 70B models, fits on a single NVIDIA RTX PRO 6000 Blackwell (96 GiB). Seed counts are stated on each results table, effect sizes are in units of the pooled per-seed standard deviation, and the full protocol is in App. E.

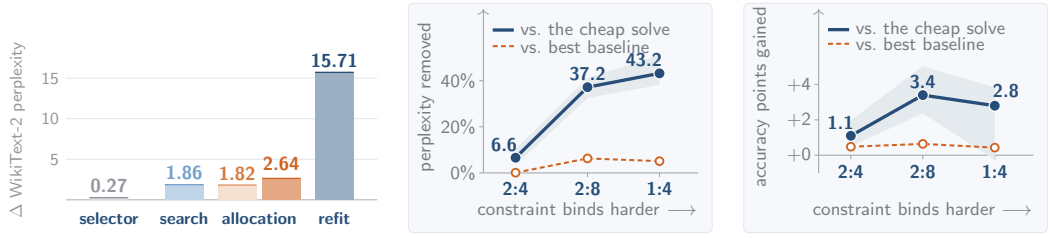
5.1 Comparison under a fixed pattern

Under uniform 2:4 — the one N:M pattern with a deployable kernel at every scale — KSP takes the best accuracy on six of the seven models and the best perplexity on four (Tab. 1). The comparison that isolates the formulation is against ALPS, which minimizes the same layerwise reconstruction objective over the nonconvex N:M set and is given the same tuning courtesy; there KSP wins perplexity on all six models where ALPS runs and both metrics on five of them. The learned-mask methods, which spend GPU-hours to GPU-days, do not reach KSP on accuracy at any scale.

The picture separates when the per-block budget binds harder. At 75% sparsity every ranking-only rule collapses to three- to five-digit perplexity and to the accuracy floor, while every method that updates the survivors stays up to two orders of magnitude below them, and nearly a factor of five even in the closest cell. Among those solvers KSP leads ALPS in eight of ten perplexity and nine of ten accuracy cells, and beats SparseGPT on perplexity in all ten (Tab. 10). This is the regime the formulation was built for, and §5.2 shows it is also where *how* the support is searched pays most.

5.2 Sizing the four decisions

Fig. 3a sizes all four decisions on one model, everything else held fixed, and the ordering inverts the one the literature's effort implies. On these controlled comparisons, the measured refit effect is roughly six times as large as the largest allocation effect and eight times as large as the search effect;



(a) Four decisions, one model. **(b)** Search value, perplexity. **(c)** Search value, accuracy.

Figure 3: **Sizing the decisions.** a Llama-3.2-3B, one decision varied at a time, each bar a difference between two reported arms: the selector against ALPS at the reported solver budget; the search; the allocation against a random dense-selection control under two protocols (pale, matched latency; solid, matched parameter budget at 62.5% density); and the refit switched on or off, measured on RIA's mask; each bar is a separate comparison (App. E.2). b, c Mean over five models, band the per-model spread: what the reported solver budget buys over the same objective solved cheaply (solid) and over the strongest baseline in each cell (dashed).

search and allocation come next, within a perplexity point of each other; and the selector, the axis most of this literature optimizes, is at the reported solver budget the smallest of the four (a). The stronger *search* is worth several times that selector gap at 2:4 and far more again at 75% (b), with the same jump visible in accuracy (c), as a finer pattern makes the support decision harder.

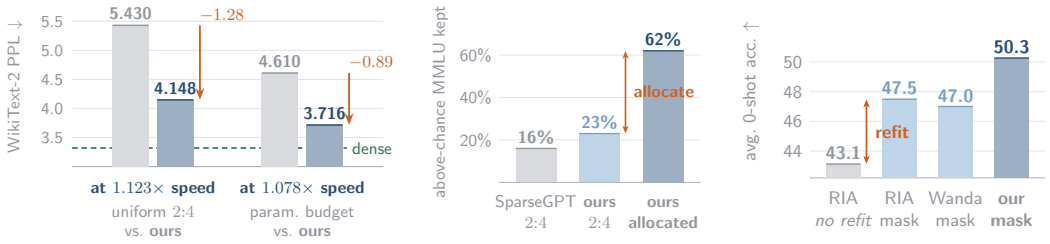
Two controls establish the refit and allocation effects. For the refit, we give every arm KSP's masked-CG re-solve and vary only the mask (Fig. 4c, Tab. 11, App. E.2): adding the refit to a fixed mask is worth several times more perplexity than the widest difference between masks, while on accuracy the masks do separate, and ours is best on both metrics at both scales. For allocation, we run the strongest competing selector under *KSP's own allocation*, our arm at the reduced solver budget, and ask how much of its uniform-pattern advantage survives: at 3B almost all of it disappears, and on accuracy KSP moves ahead (Tab. 14). Run against the saliency and reconstruction selectors across five models under matched allocation, the same design leaves KSP first on perplexity everywhere (Fig. 8). Allocating the budget absorbs most of what a selector is worth under a uniform pattern — the claim §5.3 tests directly, at fixed hardware latency.

5.3 Allocation at a fixed hardware budget

Allocation can be matched on the *parameter budget* of retained weights or on the *hardware budget* of measured latency; the cleanest test fixes the latter and varies only where the budget goes. On Llama-3.2-3B, with selector and compensation identical, the water-level allocation beats a random dense-selection control at matched measured latency on both metrics over three seeds. At matched parameter budget, the outlier-ratio allocator OWL does no better than that control, supporting the interpretation that the gain comes from pricing the budget rather than simply spreading it unevenly (Tab. 13). On Llama-2-70B (Fig. 4a), graph-captured 2:4 speedups differ sharply between projections, so sparsifying a matrix whose kernel provides essentially no speedup can incur quality loss without a corresponding latency benefit. Keeping them dense (*free-dense*), one line and no allocator, beats uniform 2:4 at identical speed by 1.28 perplexity, and at a matched 1.078× pricing the budget by latency rather than parameters is worth a further 0.89. At 0.95 of dense latency the allocation recovers close to nine tenths of the perplexity uniform 2:4 gives up (Tab. 15).

Three properties make the allocation practical. It *transfers*: mapped by relative depth and matrix name, an allocation computed on a 7B model is applied unchanged at 70B, where it delivers the latency-priced gain above with no 70B allocator run (Fig. 9). It is *stable under calibration*: nearly nine matrices in ten keep the identical budget when the corpus changes (App. E.2). And the free-dense gain *grows with model quality*: transferred unchanged to Llama-3.1-70B, trained on far more tokens per parameter, it more than doubles its Llama-2-70B WikiText-2 gain and holds on every corpus and on accuracy (Fig. 5). It composes with quantization and survives LoRA recovery (Tab. 16).

The decisions separate more sharply on knowledge and reasoning benchmarks (Fig. 4b). At 62.5% density against the uniform arms' 50%, the allocated configuration retains about 2.7 times as much



(a) Matched latency, Llama-2-70B. **(b)** Capability retained. **(c)** Mask against refit.

Figure 4: **What the decisions buy.** a Llama-2-70B, each pair at the *same* measured prefill speed: free-dense vs. uniform 2:4 at 1.123×, latency- vs. parameter-priced allocation at 1.078×; dashed: dense. b Llama-3.2-3B, above-chance MMLU kept; allocated arm at 62.5% density, uniform arms at 50%. c Llama-3.2-3B at 2:4, mean zero-shot accuracy; every bar but the leftmost carries KSP's refit. a, b at the reduced solver budget, c at the reported one (y axis truncated); numbers in App. E.2.

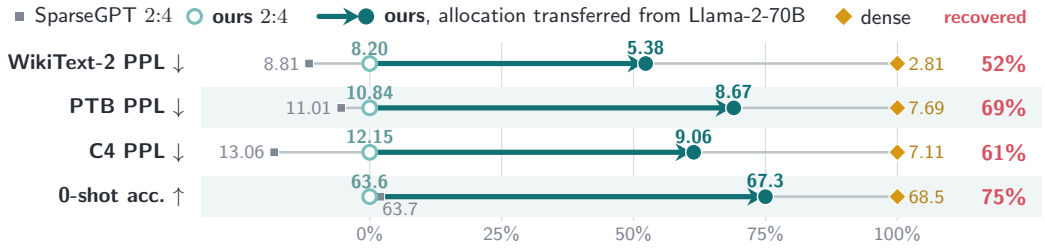


Figure 5: **The allocation transfers to a newer model.** Llama-3.1-70B at 2:4 with the Llama-2-70B free-dense rule applied unchanged; the models share every decoder-layer linear shape, so its latency match carries over. Each row runs from our uniform-2:4 model (0%) to the dense model (100%); the transferred allocation closes 52–75% of that gap on every metric (values in Tab. 15).

of the dense model's above-chance MMLU as our uniform arm and close to four times as much as SparseGPT's, on a benchmark where the zero-shot mean separates the three less sharply; on GSM8K it retains more than twice as much of the dense score as either uniform arm. An averaged zero-shot score thus understates what allocation preserves. The ordering of §5.2 thus holds at deployment scale: at its inherited latency match on Llama-3.1-70B, the transferred free-dense rule is worth three to thirteen times the perplexity separating SparseGPT from our selector at 2:4 (Fig. 5).

6 Conclusion and Future Work

N:M pruning has meant choosing between heuristics that never revisit their ranking and mask optimization that descends a nonconvex surrogate. We showed that the geometry itself admits a tractable convex relaxation: per block, the convex hull of the N-sparse unit-ℓ_2 ball is the unit ball of the k-support norm, whose square is our regularizer. That relaxation is constructive — closed-form gates and weighted prox, a lossless-hardening certificate, exact recovery under diagonal curvature — so the layerwise objective becomes a convex program solved one-shot, with magnitude and activation-scaled saliency its zero-iteration limit. Read across matrices, the same water level prices the budget, so one object decides both which weights to keep and where sparsity is spent.

Separating the decisions one-shot pruning conflates is what the framework buys empirically: in our controlled comparisons compensation produces the largest measured change, allocation and search come next, the value of a stronger search growing from 2:4 to 75% sparsity, and the within-block selector, the axis most of this literature optimizes, is the smallest. In practice this reverses the usual priority: refitting the survivors and pricing the budget in measured latency matter before the choice of selector, and both come from the same convex object at one-shot cost, with no backpropagation. Two directions follow: extending the support-recovery guarantee to block-diagonal curvature, with the coupling strength entering the separation condition; and making the allocator quantization-aware.

AI USE STATEMENT

Generative AI tools were used in four ways, disclosed here and in the submission form. They edited author-written drafts for clarity and grammar; they produced preliminary drafts of some expository passages — parts of the related work and of the experimental descriptions — which the authors then rewrote and checked; they assisted with parts of the proofs of the theorems, each of which the authors then verified line by line; and they served as a coding assistant while implementing the method and the baselines and for experiment execution and analysis, launching and monitoring runs, collecting logs, and tabulating and plotting the recorded measurements. All mathematical statements, proofs, experimental designs and numerical results are the authors' own, and every claim, citation and number was verified by the authors against the proofs in the appendix and the recorded run logs. The authors take full responsibility for the entire content of the paper.

ETHICS STATEMENT

This study does not involve human subjects or personally identifiable data. It uses publicly released pre-trained language models and public evaluation corpora, and concerns the efficiency of existing models: the method reduces deployment cost and energy. The main ethical consideration is that compression can amplify or attenuate model behaviour non-uniformly, and an averaged benchmark score can hide heterogeneous capability loss. We therefore report capability benchmarks (§5.3) that make the loss under uniform sparsity explicit rather than averaged away, and the released logs carry the recorded per-task zero-shot scores behind the averages in the paper, with the largest single-task degradation marked for every run in the record.

REPRODUCIBILITY STATEMENT

All theoretical claims are proved in App. B–D, with all conventions stated; Algs. 1 and 2 (App. B.2) are fully specified by Thm. 1. The experimental protocol, baseline configurations, calibration data and seeds are in §5 and App. E, with the hyperparameter grids and the complete record in the released code, and the configuration-selection rule and its pre-registration in App. C. The code, including the scripts that regenerate every table from the raw run records, is provided as supplementary material for the reviewing process and will be made public if this work is accepted.

REFERENCES

Brandon Amos, Vladlen Koltun, and J. Zico Kolter. The limited multi-label projection layer. *arXiv preprint arXiv:1906.08707*, 2019.

Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse prediction with the k-support norm. In *Advances in Neural Information Processing Systems*, 2012.

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*, 2024.

Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009. doi: 10.1109/TIP.2009.2028250.

Ke Bian, Lu Sun, and Dengji Zhao. Learning compact neural networks via generalized structured sparsity. In *European Conference on Artificial Intelligence (ECAI)*, pp. 2605–2612, 2024.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, 2020.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Roberto L. Castro, Andrei Ivanov, Diego Andrade, Tal Ben-Nun, Basilio B. Fraguela, and Torsten Hoefler. VENOM: A vectorized N:M format for unleashing the power of sparse tensor cores. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2023.

Yuli Chen, Bo Cheng, Jiale Han, Yingying Zhang, Yingting Li, and Shuhao Zhang. DLP: Dynamic layerwise pruning in large language models. In *International Conference on Machine Learning*, volume 267, pp. 7934–7956. PMLR, 2025.

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

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*, 2024.

Luhuan Fei, Lu Sun, Mineichi Kudo, and Keigo Kimura. Structured sparse multi-task learning with generalized group lasso. In *European Conference on Artificial Intelligence (ECAI)*, pp. 692–699, 2023.

Luhuan Fei, Xinyi Wang, Jiankun Wang, Lu Sun, and Yuyao Zhang. Multi-level sparse network lasso: Locally sparse learning with flexible sample clusters. *Neurocomputing*, 635:129898, 2025.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, 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. A framework for few-shot language model evaluation. Zenodo, version v0.4.0, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tianteng Gu, Bei Liu, Bo Xiao, Ke Zeng, Jiacheng Liu, and Yanmin Qian. DenoiseRotator: Enhance pruning robustness for LLMs via importance concentration. In *Advances in Neural Information Processing Systems*, 2025.

Babak Hassibi, David G. Stork, and Gregory J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pp. 293–299. IEEE, 1993.

Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

Weiyu Huang, Yuezhou Hu, Guohao Jian, Jun Zhu, and Jianfei Chen. Pruning large language models with semi-structural adaptive sparse training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24167–24175, 2025.

Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *International Conference on Machine Learning*, pp. 433–440, 2009.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, et al. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jonas M. Kübler, Yu-Xiang Wang, Shoham Sabach, Navid Ansari, Matthäus Kleindessner, Kailash Budhathoki, Volkan Cevher, and George Karypis. A proximal operator for inducing 2:4-sparsity. *Transactions on Machine Learning Research*, 2025.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, 2020.

Weijia Lin, Jiankun Wang, Lu Sun, Mineichi Kudo, and Keigo Kimura. Multi-label personalized classification via exclusive sparse tensor factorization. In *IEEE International Conference on Data Mining (ICDM)*, pp. 398–407, 2023.

Hongyi Liu, Rajarshi Saha, Zhen Jia, Youngsuk Park, Jiaji Huang, Shoham Sabach, Yu-Xiang Wang, and George Karypis. ProxSparse: Regularized learning of semi-structured sparsity masks for pretrained LLMs. In *International Conference on Machine Learning*, volume 267, pp. 39405–39419. PMLR, 2025.

Mingge Lu, Jingwei Sun, Junqing Lin, Zechun Zhou, and Guangzhong Sun. Lua-LLM: Learning unstructured-sparsity allocation for large language models. In *Advances in Neural Information Processing Systems*, 2025.

Andrew M. McDonald, Massimiliano Pontil, and Dimitris Stamos. Spectral k-support norm regularization. In *Advances in Neural Information Processing Systems*, 2014.

Andrew M. McDonald, Massimiliano Pontil, and Dimitris Stamos. New perspectives on k-support and cluster norms. *Journal of Machine Learning Research*, 17(155):1–38, 2016.

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

Xiang Meng, Mehdi Makni, and Rahul Mazumder. TSENOR: Highly-efficient algorithm for finding transposable N:M sparse masks. In *Advances in Neural Information Processing Systems*, 2025.

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

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

Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013. doi: 10.1007/s10107-012-0629-5.

Vlad Niculae, André F.T. Martins, Mathieu Blondel, and Claire Cardie. SparseMAP: Differentiable sparse structured inference. In *International Conference on Machine Learning*, 2018.

Guillaume Obozinski and Francis Bach. Convex relaxation for combinatorial penalties. *arXiv preprint arXiv:1205.1240*, 2012.

Brendan O'Donoghue and Emmanuel Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015.

Jeff Pool and Chong Yu. Channel permutations for N:M sparsity. In *Advances in Neural Information Processing Systems*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Lu Sun and Jun Sakuma. Learning semi-structured sparsity for LLMs via shared and context-aware hypernetwork. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lqjQs2lVNm.

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.

Yan Sun, Qixin Zhang, Zhiyuan Yu, Xikun Zhang, Li Shen, and Dacheng Tao. MaskPro: Linear-space probabilistic learning for strict (N:M)-sparsity on LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0R06BghLJX.

Ryan J. Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *The Annals of Statistics*, 39(3):1335–1371, 2011.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. In *Advances in Neural Information Processing Systems*, 2020.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus Müller, Jonas M. Kübler, Rupak Vignesh Swaminathan, Athanasios Mouchtaris, Sravan Babu Bodapati, Nathan Susanj, Zheng Zhang, Jack FitzGerald, and Abhishek Kumar. Wanda++: Pruning large language models via regional gradients. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4321–4333, 2025.

Zhiguo Yang, Changjian Deng, Qinke Chen, Zijing Zhou, and Jian Cheng. LSA: Layer-wise sparsity allocation for large language model pruning based on minimal linear reconstruction error. In *International Conference on Learning Representations*, 2026.

Lu Yin, You Wu, Zhenyu Zhang, et al. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *International Conference on Machine Learning*, 2024.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *International Conference on Learning Representations*, 2024a.

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

Wenqian Zhao, Lancheng Zou, Zixiao Wang, Xufeng Yao, and Bei Yu. HAPE: Hardware-aware LLM pruning for efficient on-device inference optimization. *ACM Transactions on Design Automation of Electronic Systems*, 30(4):61:1–61:18, 2025. doi: 10.1145/3744244.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning N:M fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations*, 2021.

Yang Zhou, Rong Jin, and Steven C.H. Hoi. Exclusive lasso for multi-task feature selection. In *International Conference on Artificial Intelligence and Statistics*, pp. 988–995, 2010.

Lancheng Zou, Shuo Yin, Zehua Pei, Tsung-Yi Ho, Farzan Farnia, and Bei Yu. PermLLM: Learnable channel permutation for N:M sparse large language models. In *Advances in Neural Information Processing Systems*, 2025.

APPENDIX ROADMAP

App. A places KSP against the four literatures it draws on. App. B proves the per-block tightness, the gates and weighted prox, the certificate, matched scaling and the diagonal-curvature recovery of §3, together with the vectorized prox and its operation counts. App. C describes the support search, the configuration rule and the compute dial. App. D proves the allocation results of §4 and gives the allocator and the executable rounding. App. E records the experimental protocol and the results deferred from §5.

A RELATED WORK IN FULL

One-shot LLM pruners. SparseGPT (Frantar & Alistarh, 2023) performs greedy OBS-style elimination (Hassibi et al., 1993) with sequential compensation; Wanda (Sun et al., 2024) and RIA (Zhang et al., 2024a) rank by activation-scaled saliency and DSnoT (Zhang et al., 2024b) refines such a mask by prune-and-grow against reconstruction error; ALPS (Meng et al., 2024) optimizes the same layerwise objective we do under an ℓ_0 constraint via nonconvex ADMM (Boyd et al., 2011), which makes it the baseline against which a change of formulation is easiest to read. Wanda++ (Yang et al., 2025) adds decoder-block gradients to the saliency score and pairs it with a regional weight update, improving both halves of the selection–compensation split at once. DenoiseRotator (Gu et al., 2025) applies learned orthogonal transforms before pruning and PermLLM (Zou et al., 2025) learns a channel permutation that lowers the pruning error; both act on the *input* to a selector. SparseLLM (Bai et al., 2024) attacks the local–global gap of the layerwise objective itself. *Our position.* Every method here chooses its support by a rule — a ranking, a greedy elimination order, a projection onto the nonconvex set — and inherits whatever guarantee that rule carries. We replace the rule with a convex program whose relaxation is tight for the constraint the hardware imposes, which is what lets us say what the support is optimal for, certify when rounding it costs nothing, and recover the saliency rules as the zero-iteration case (Prop. 6) rather than as rivals. The preconditioning and local–global methods change what the selector is handed, or the objective it is handed, not how it chooses, and compose with ours.

Learnable-mask methods. MaskLLM (Fang et al., 2024) learns a categorical distribution over the $\binom{M}{N}$ block masks via Gumbel-Softmax (Jang et al., 2017); MaskPro (Sun et al., 2026) keeps that probabilistic view with a *linear-space* policy over strict N:M masks; Sun & Sakuma (2026) *amortize* selection through a shared hypernetwork; and adaptive sparse training (Huang et al., 2025) reaches good 2:4 masks by distillation-guided retraining. On the penalty side, Kübler et al. (2025) introduce the elementary symmetric polynomial $e_{N+1}(|\boldsymbol{w}_g|)$ — a nonconvex exact penalty vanishing precisely on blocks with at most N nonzeros, with a proximal operator derived for 2:4 — and ProxSparse (Liu et al., 2025) carries it into an end-to-end learning framework. TSENOR (Meng et al., 2025) relaxes the *transposable* mask search into a per-block entropy-regularized optimal-transport program, solved iteratively and rounded within one-shot pruning. *Our position.* This family and ours both relax the mask polytope; the difference is what the relaxation is *of* and how it is solved. The learned-mask methods use smoothings chosen so that gradients flow, or exact penalties carrying critical-point guarantees only, and learn the mask by gradient descent, end to end or layer by layer; TSENOR relaxes a saliency objective under a transposability constraint. Ours is the tight convex gauge of the constraint set itself, applied to the reconstruction objective with a closed-form per-block prox, in one post-training pass. Kübler et al. (2025) is the nearest methodological neighbour — a layerwise, post-training proximal method for 2:4 — but its regularizer is a nonconvex exact penalty chosen to exploit local correlations between input features rather than the gauge of the constraint set, its prox needs an iterative inner solver, and it has no analogue of the water level. The gauge is what generalizes: tight for every N:M, with a closed-form prox, and it is what carries the water level into the allocation problem.

Non-uniform sparsity budgets. These are the closest neighbours of our second contribution, and they differ in what they use as a proxy for the value of budget: outlier statistics (Yin et al., 2024), weight-and-activation statistics (Chen et al., 2025), a per-projection reconstruction proxy (Yang et al., 2026), or a learned soft top-k (Lu et al., 2025). HAPE (Zhao et al., 2025) prices the budget in measured on-device cost rather than in parameters, but by a heuristic search. *Our position.* Each of these must invent or learn a proxy, and then tune it, on top of whatever selector it wraps. We

introduce no separate proxy: the squared water level of each block's solution *is* the marginal value of budget in the penalty (8), so allocation and selection fall out of one convex object with a closed-form optimality condition. HAPE is closest in spirit; it obtains its price from a separate search rather than from the selection problem, so its allocator and its pruner remain two algorithms.

Structured-sparsity penalties. Group lasso (Yuan & Lin, 2006), exclusive lasso (Zhou et al., 2010; Lin et al., 2023), their generalized, structured and multi-level variants (Fei et al., 2023; Bian et al., 2024; Fei et al., 2025) and the generalized lasso (Tibshirani & Taylor, 2011) form the classical toolbox, alongside soft-threshold reparameterization (Kusupati et al., 2020), N:M training from scratch (Zhou et al., 2021) and channel permutation for N:M (Pool & Yu, 2021). As §3 establishes, that toolbox relaxes a different combinatorial set for $N \geq 2$; Tab. 2 states it in a line each.

Table 2: **Why the classical penalties relax a different set.** Only the k-support gauge carries the cardinality N.

Penalty	What it encodes per block	Correct for N:M?
Group lasso $\sum_b \|\boldsymbol{v}_{g_b}\|_2$	the whole block vanishes	no — the opposite pattern
Exclusive lasso $\sum_b \|\boldsymbol{v}_{g_b}\|_1^2$	competition, one survivor	no — provably $1{:}M$
k-support $\sum_b \|\boldsymbol{v}_{g_b}\|_{(N)}^2$	at most N survivors	yes, and tightly

Two details sharpen the table. Generalized group-lasso forms that place an operator inside the group norm (Fei et al., 2023) inherit the group-lasso failure mode: an operator changes *which* linear combination is driven to zero, not the fact that the whole group goes at once. And the exclusive-lasso reading is a theorem, not an observation — $(\sum_b \|\boldsymbol{v}_{g_b}\|_1^2)^{1/2}$, whose square is the exclusive-lasso penalty, is the tightest convex relaxation of *at most one nonzero per block* (Obozinski & Bach, 2012, §4.2) — which is why it remains the natural-looking candidate while being provably the wrong one for $N \geq 2$. *Our position.* The correct object is the k-support norm (Argyriou et al., 2012) in its variational box-norm form (McDonald et al., 2016; 2014), an infimal-convolution penalty in the family of the latent overlapping group lasso (Jacob et al., 2009) rather than a sum-form $\sum_\pi \|\boldsymbol{D}_\pi \boldsymbol{w}\|_2$, with the operator role of $\boldsymbol{D}$ taken over by *calibration statistics* (Prop. 5). The water-filling gate is functionally a soft top-k operator on the capped simplex, adjacent to optimal-transport soft top-k (Xie et al., 2020), limited multi-label projection (Amos et al., 2019), SparseMAP (Niculae et al., 2018) and differentiable sorting (Blondel et al., 2020); the distinction is provenance — ours is the exact variational form of a *norm* inside a convex program, which is what yields the closed-form prox and the certificate. What is new here is the identification — that N:M hardware sparsity has precisely this convex geometry — and what follows from one object: a closed-form N:M gate, an exact prox, a hardening certificate, exact recovery in a tractable regime, and a water-level price for the budget. Prior uses of the norm target regression and multi-task problems with no per-block hardware pattern, and, to our knowledge, no prior N:M method derives selection and allocation from one relaxation.

B THE RELAXATION, THE PROX AND THE GUARANTEES: FULL PROOFS

This appendix proves the statements of §3 in the order they are made: the variational identity and the gates, then the weighted proximal operator with its complexity and support regimes, then the matched scaling under which the recovery guarantee holds.

B.1 THE VARIATIONAL IDENTITY AND THE GATES

Conventions. $\frac{0}{0} := 0$, $\frac{t}{0} := +\infty$ $(t > 0)$; $|v|_{(1)} \geq \cdots \geq |v|_{(M)}$ sorted magnitudes, with $|v|_{(0)} := +\infty$ and $|v|_{(M+1)} := 0$; $\Delta_{N,M}$ as in §3.

Lemma 1 (Perspective representation)**.** *For $(v, \theta) \in \mathbb{R} \times [0, 1]$: $v^2/\theta = \sup_{t \in \mathbb{R}}(2tv - t^2\theta)$. Hence $(v, \theta) \mapsto v^2/\theta$ is jointly convex and lsc as a supremum of affine functions, and for fixed $\boldsymbol{v}$ the minimum of $\sum_j v_j^2/\theta_j$ over the compact $\Delta_{N,M}$ is attained and finite ($\boldsymbol{\theta} = \frac{N}{M}\mathbf{1}$ gives $\frac{M}{N}\|\boldsymbol{v}\|_2^2$).*

Proof. For $\theta > 0$ the supremum is at $t = v/\theta$; for $\theta = 0$ it is $+\infty$ unless $v = 0$, where it is 0. Convexity/lsc as a supremum of affine functions; attainment by Weierstrass. □

Lemma 2 (Extreme points of the mask polytope)**.** *The extreme points of $\Delta_{N,M}$ are exactly the binary vectors with at most N ones; those with exactly N ones are the binary N:M masks.*

Proof. Binary points with $\leq N$ ones are extreme because they are already extreme in the enclosing box $[0,1]^M \supseteq \Delta_{N,M}$, and an extreme point of a superset that lies in a subset is extreme there. Conversely, a feasible $\boldsymbol{\theta}$ with two coordinates in $(0,1)$ admits the feasible perturbations $\boldsymbol{\theta} \pm \epsilon(\boldsymbol{e}_i - \boldsymbol{e}_j)$ (budget unchanged, box satisfied for small ϵ), so it is not extreme; with exactly one fractional coordinate, either the budget is slack ($\boldsymbol{\theta} \pm \epsilon \boldsymbol{e}_i$ both feasible: not extreme) or the budget is tight with the rest binary, forcing the fractional coordinate to equal the integer $N - \#\{\text{ones}\}$ — a contradiction. □

Proof of Proposition 1. Tightness. Write g for the gauge of the N-sparse unit-ℓ_2 ball ($g(\boldsymbol{v}) = \|\boldsymbol{v}\|_2$ if $\|\boldsymbol{v}\|_0 \leq N$, $+\infty$ otherwise). $\|\cdot\|_{(N)}$ is the gauge of $K_N = \operatorname{conv}$ of that set, so $\|\cdot\|_{(N)} \leq g$ and $\|\cdot\|_{(N)}$ is convex. For maximality, let h be any convex $h \leq g$ (so $h(\mathbf{0}) \leq g(\mathbf{0}) = 0$), and let $\boldsymbol{v} = \sum_i \boldsymbol{u}_i$ with each $\boldsymbol{u}_i$ N-sparse and not all zero. Put $\Sigma := \sum_i \|\boldsymbol{u}_i\|_2 > 0$, $\lambda_i := \|\boldsymbol{u}_i\|_2 / \Sigma$ and $\boldsymbol{w}_i := \boldsymbol{u}_i/\lambda_i$ (for $\lambda_i > 0$), so that $\boldsymbol{v} = \sum_i \lambda_i \boldsymbol{w}_i$ is a convex combination and $\|\boldsymbol{w}_i\|_2 = \Sigma$ with $\boldsymbol{w}_i$ N-sparse. Then $h(\boldsymbol{v}) \leq \sum_i \lambda_i h(\boldsymbol{w}_i) \leq \sum_i \lambda_i g(\boldsymbol{w}_i) = \sum_i \lambda_i \Sigma = \Sigma$. Taking the infimum over such decompositions and using the infimal-convolution form $\|\boldsymbol{v}\|_{(N)} = \inf\{\sum_i \|\boldsymbol{u}_i\|_2 : \sum_i \boldsymbol{u}_i = \boldsymbol{v},\ \|\boldsymbol{u}_i\|_0 \leq N\}$ (Argyriou et al., 2012) gives $h \leq \|\cdot\|_{(N)}$. (The set is compact, so K_N is closed and the infimum is attained.)

Product of hulls. For arbitrary nonempty sets, $\operatorname{conv}(\prod_b A_b) = \prod_b \operatorname{conv}(A_b)$: "$\subseteq$" holds because $\prod_b \operatorname{conv}(A_b)$ is a convex superset of $\prod_b A_b$; for "$\supseteq$", given $x_b = \sum_i \lambda_i^b a_i^b$ per factor, the product distribution $\prod_b \lambda^b$ exhibits $(x_1, \ldots, x_B)$ as a convex combination of points of $\prod_b A_b$. No compactness is needed. This is a statement about the feasible set, not a claim that the aggregated penalty (3) is the joint gauge (Rem. 1).

Endpoint cases. For $N = 1$, K_1 is the cross-polytope, whose gauge is ℓ_1; for $N = M$, K_M is the ℓ_2 ball. □

Proof of Theorem 1(i). Let $\Omega(\boldsymbol{v})^2 := \min_{\boldsymbol{\theta} \in \Delta_{N,M}} \sum_j v_j^2/\theta_j$ (well-defined by Lemma 1).

Ω is a norm. Homogeneity is clear ($\Delta_{N,M}$ does not depend on $\boldsymbol{v}$). Since $\theta_j \leq 1$, $\Omega \geq \|\cdot\|_2 > 0$ off the origin. Convexity of $\Phi := \Omega^2$: joint convexity is preserved under partial minimization over the convex $\Delta_{N,M}$. Subadditivity from convexity plus 2-homogeneity: for $t \in (0,1)$, $\Phi(\boldsymbol{u} + \boldsymbol{v}) \leq \Phi(\boldsymbol{u})/t + \Phi(\boldsymbol{v})/(1-t)$, minimized at $t^\star = \sqrt{\Phi(\boldsymbol{u})}/(\sqrt{\Phi(\boldsymbol{u})} + \sqrt{\Phi(\boldsymbol{v})})$ to give $\Phi(\boldsymbol{u} + \boldsymbol{v}) \leq (\sqrt{\Phi(\boldsymbol{u})} + \sqrt{\Phi(\boldsymbol{v})})^2$.

$\Omega = \|\cdot\|_{(N)}$ by duality. The unit ball of Ω is $\bigcup_{\boldsymbol{\theta} \in \Delta_{N,M}} E_{\boldsymbol{\theta}}$, $E_{\boldsymbol{\theta}} = \{\boldsymbol{v} : \sum_j v_j^2/\theta_j \leq 1\}$ (an ellipsoid, degenerate to $v_j = 0$ where $\theta_j = 0$). Hence $\Omega^*(\boldsymbol{u}) = \sup_{\boldsymbol{\theta} \in \Delta_{N,M}} \left(\sum_j \theta_j u_j^2\right)^{1/2}$ (support function of an ellipsoid, attained at $v_j \propto \theta_j u_j$). The inner expression is linear in $\boldsymbol{\theta}$, so the supremum is attained at an extreme point (Lemma 2); maximizing over binary supports of size $\leq N$ selects the top-N of u_j^2: $\Omega^*(\boldsymbol{u}) = \left(\sum_{j \leq N} |u|_{(j)}^2\right)^{1/2}$. Independently, the support function of K_N is $\max_{|S| \leq N} \|\boldsymbol{u}_S\|_2$ — the same expression. Two closed norms with equal duals coincide, so $\Omega = \|\cdot\|_{(N)}$.

Minimizer. If $\|\boldsymbol{v}\|_0 \leq N$: $\theta_j = \mathbf{1}\{v_j \neq 0\}$ is feasible with value $\|\boldsymbol{v}\|_2^2$, matching the global lower bound $\sum_j v_j^2/\theta_j \geq \|\boldsymbol{v}\|_2^2$. If $\|\boldsymbol{v}\|_0 > N$: the problem is convex with Slater's point $\frac{N}{2M}\mathbf{1}$, so KKT is necessary and sufficient. If the budget were slack ($\nu = 0$), stationarity $-v_j^2/\theta_j^2 + \mu_j = 0$ with μ_j active only at the cap would force $\theta_j = 1$ on $\operatorname{supp}(\boldsymbol{v})$, contradicting the budget. With $\nu > 0$: interior coordinates give $\theta_j = |v_j|/\sqrt{\nu}$, capped ones $|v_j| \geq \sqrt{\nu}$; set $\tau := \sqrt{\nu}$ to obtain (5). The budget function $\varphi(\tau) = \sum_j \min(|v_j|/\tau, 1)$ is continuous, nonincreasing, equal to $\|\boldsymbol{v}\|_0 > N$ as $\tau \downarrow 0$ and to 0 as $\tau \to \infty$, and strictly decreasing wherever a coordinate is strictly interior (local form $r + S_r/\tau$ with $S_r > 0$); a plateau *at level* N would force $r = N$ and $S_r = 0$, i.e. $\|\boldsymbol{v}\|_0 = N$,

Table 3: Existing methods as corners of the design space. The last three rows are the same KSP objective at three compute budgets: no iterations, a reduced budget used by several controls, and the configuration we report, which preconditions the step. The last two differ from ALPS in what the loop is split around — the convex penalty rather than the nonconvex set (App. C).

Method	Geometry	Sparsity mechanism	Iters	Weight update
Magnitude	$\boldsymbol{I}$	block top-N (ℓ_0 proj.)	0	none
Wanda	$\mathrm{diag}(\boldsymbol{H})$	block top-N (ℓ_0 proj.)	0	none
SparseGPT	$\boldsymbol{H}$	greedy ℓ_0 + OBS	0	sequential
ALPS	$\boldsymbol{H}$	splitting around $\mathcal{C}_{N:M}$ (nonconvex)	many	ADMM
ProxSparse	end-to-end $\nabla\mathcal{L}$	exact penalty e_{N+1}	SGD	none (snap)
MaskLLM	end-to-end $\nabla\mathcal{L}$	vertex distribution	SGD	optional
KSP, no iters	$\mathrm{diag}(\boldsymbol{H})$	closed-form gates, hardened	0	masked CG
KSP, reduced	$\boldsymbol{H}$	splitting around $\Omega^{\boldsymbol{\beta}}_{N:M}$ (convex)	prox-grad., trunc.	masked CG
KSP, reported	$\boldsymbol{H}$	splitting around $\Omega^{\boldsymbol{\beta}}_{N:M}$ (convex)	precond.	masked CG

excluded. Hence τ, and $\boldsymbol{\theta}^\star$ on $\mathrm{supp}(\boldsymbol{v})$, are unique. If $r = N$ coordinates were capped, the budget would force $S_r = 0$, again excluded, so $r \le N-1$ — *in the norm problem*; the prox admits N capped coordinates in the integral regime (Rem. 2).

Sorted closed form. Order $|v|_{(1)} \ge \cdots \ge |v|_{(M)}$ and put $S_r := \sum_{i>r} |v|_{(i)}$. The capped set is $\{j : |v_j| \ge \tau\}$, so with r capped coordinates the budget reads $r + S_r/\tau = N$, i.e.

$$\tau \;=\; \frac{S_r}{N-r}, \qquad |v|_{(r)} \;\ge\; \frac{S_r}{N-r} \;\ge\; |v|_{(r+1)}, \qquad r \in \{0,\dots,N-1\}. \tag{10}$$

The water level τ is unique by the strict monotonicity of φ just established, and the pair (r,τ) with $r := \#\{j : |v_j| \ge \tau\}$ is then unique; if some $|v_j| = \tau$, other values of r also satisfy (10), all with the same τ and the same $\boldsymbol{\theta}^\star$. Substituting $\theta^\star_j = 1$ on the capped set and $\theta^\star_j = |v_j|/\tau$ elsewhere into $\sum_j v_j^2/\theta^\star_j$ gives

$$\|\boldsymbol{v}\|^2_{(N)} \;=\; \sum_{i\le r} |v|^2_{(i)} \;+\; \tau \sum_{i>r} |v|_{(i)} \;=\; \sum_{i\le r} |v|^2_{(i)} \;+\; \frac{S_r^2}{N-r}, \tag{11}$$

the closed form referred to in Thm. 1; one sort and one prefix-sum scan evaluate it in $O(M \log M)$. □

B.2 The weighted prox: roles, complexity and regimes

Tab. 3 places the existing methods, and our three solver budgets, as corners of one design space.

Two roles for the operator. The operator is used twice, and the weights differ, which matters for how the results below should be read. *As a FISTA primitive* (Alg. 1), the majorization is isotropic — a scalar Lipschitz constant L — so $\boldsymbol{a} = \mathbf{1}$ and the effective shrinkage $c_j = 2(\lambda_t/L)\beta_j$ varies across a block; a proximal step is a step, and its intermediate selection statistic carries no interpretation, so the mask is hardened from the *norm* gates of the final iterate (Alg. 1) rather than from the prox statistic, so that the ranking is read in the loss's own scale $s_j|\widehat{w}_j|$; the step metric's bias vanishes at the fixed point (App. B.3). *As a characterization of a solution* — the zero-iteration gate evaluation, and the block subproblem under diagonal curvature — the relevant weights are the loss's own curvature, $a_j = H_{jj} = s_j^2$, giving $c_j \equiv 2\lambda$ for $\beta_j = H_{jj}$ (and $2\lambda\zeta$ for $\beta_j = \zeta H_{jj}$); there c_j is constant, the ratio test of Cor. 1 is well posed, and every selection statement in the paper (Prop. 5, Prop. 6, Prop. 2) is made.

For dense $\boldsymbol{H}$ the two readings do not coincide. The FISTA fixed point $\widehat{\boldsymbol{W}}$ is a stationary point of $f + \lambda\Omega^{\boldsymbol{\beta}}_{N:M}$, and it is the $\mathrm{diag}(\boldsymbol{H})$-metric prox of $\lambda\Omega^{\boldsymbol{\beta}}_{N:M}$ at $\boldsymbol{z} = \widehat{\boldsymbol{W}} - (\widehat{\boldsymbol{W}} - \boldsymbol{W}_0)\boldsymbol{H}\,\mathrm{diag}(\boldsymbol{H})^{-1}$, which reduces to $\boldsymbol{W}_0$ identically in $\widehat{\boldsymbol{W}}$ iff $\boldsymbol{H}$ is diagonal. Under the second reading $c_j \equiv 2\lambda$ is constant and (7) applies at that $\boldsymbol{z}$; under the FISTA metric it does not, so for dense $\boldsymbol{H}$ we evaluate

Algorithm 1 KSP: per-layer pruning, one post-training pass (all steps batched over rows)

Require: $\boldsymbol{W}_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$; $\boldsymbol{H}$ (1); pattern (N, M); schedule $\lambda_0 < \cdots < \lambda_S$ over $S+1$ stages; FISTA steps K; CG steps K_{cg}; penalty weights $\boldsymbol{\beta} \propto \operatorname{diag}\boldsymbol{H}$ (Prop. 5)
$L \leftarrow (1+\delta)\,\widehat{\lambda}_{\max}(\boldsymbol{H})$, $\widehat{\lambda}_{\max}$ by power iteration, $\delta > 0$ so that $L \geq \lambda_{\max}$ (checked by backtracking on L); $\boldsymbol{W} \leftarrow \boldsymbol{W}_0$; $\boldsymbol{Y} \leftarrow \boldsymbol{W}$
for $t = 0, \ldots, S$ **do** ▷ continuation
  for $k = 1, \ldots, K$ **do** ▷ monotone FISTA, adaptive restart
    $\boldsymbol{G} \leftarrow (\boldsymbol{Y} - \boldsymbol{W}_0)\,\boldsymbol{H}$ ▷ one GEMM; $f(\boldsymbol{W}_{\text{new}})$ for the monotone test reuses $(\boldsymbol{W}_{\text{new}} - \boldsymbol{W}_0)\boldsymbol{H}$, the next iterate's gradient
    $\boldsymbol{W}_{\text{new}} \leftarrow \textsc{BlockProx}\big(\boldsymbol{Y} - \boldsymbol{G}/L;\ \lambda_t/L,\ \boldsymbol{\beta},\ \boldsymbol{a} = \mathbf{1}\big)$ ▷ Thm. 1
    update extrapolation $\boldsymbol{Y}$, monotone in F_{λ_t}
  end for
  break if per-block supports unchanged across stages
end for
$\boldsymbol{\Theta}^\star \leftarrow$ per-block top-N of the variational gates $\min(s_j|\widehat{w}_j|/\tau', 1)$ of the final iterate, τ' its water level ▷ Thm. 1; exact ranking under matched scaling
Refit: K_{cg} masked CG steps on $\min_{\boldsymbol{W}:\,\operatorname{supp}(\boldsymbol{W}) \subseteq \operatorname{supp}(\boldsymbol{\Theta}^\star)} f(\boldsymbol{W})$ ▷ one GEMM each
return N:M-sparse $\boldsymbol{W}$

the *general* integrality test on the returned gates, valid for arbitrary c_j and derived in the proof of Cor. 1,

$$\min_{j \in \mathcal{K}} \frac{c_j}{1+c_j}\,\rho_j \;\geq\; \max_{j \notin \mathcal{K}} \rho_j, \qquad \mathcal{K} = \text{the top-}N \text{ of } \rho, \tag{12}$$

of which (7) is the constant-c specialization; since $c_j/(1+c_j) < 1$, any integral keep-set is the top-N by ρ, so K is read off a single ranking for every c_j rather than searched. The splitting step of App. C uses the same isotropic reading with L replaced by its ridge parameter ϱ, so (12) is again the form evaluated there.

Proposition 3 (Operation counts and convergence)**.** *Per layer, forming $\boldsymbol{H}$ costs $O(n_{\text{cal}} d_{\text{in}}^2)$ once; each FISTA iteration costs one $(d_{\text{out}} \times d_{\text{in}})(d_{\text{in}} \times d_{\text{in}})$ GEMM plus an $O(d_{\text{out}} d_{\text{in}} \log M)$ prox pass; each masked-CG step costs one GEMM. The total is*

$$O\big(n_{\text{cal}} d_{\text{in}}^2\big) + \big((S+1)K + K_{cg}\big) \cdot O\big(d_{\text{out}} d_{\text{in}}^2\big) + O\big(P d_{\text{in}}^2\big) \text{ flops}, \qquad O\big(d_{\text{in}}^2 + d_{\text{out}} d_{\text{in}}\big) \text{ memory},$$

where P is the number of power iterations used to estimate L, and with no backpropagation anywhere. Each stage is ε-strongly convex, because the damping of (1) makes f so (§3.1). Plain FISTA guarantees $O(1/k^2)$; the strongly convex variant, or a restart interval tuned to ε, attains a linear rate in $\sqrt{\text{cond}}$ with $\text{cond} = L/\varepsilon$ (Nesterov, 2013; O'Donoghue & Candès, 2015). Monotone FISTA (Beck & Teboulle, 2009) retains the $O(1/k^2)$ rate; Alg. 1's adaptive restart is adopted for its empirical behaviour, and the rates above are stated for the variants without adaptive restart.

Implication. SparseGPT, ALPS and KSP sit in the same $d_{\text{out}} d_{\text{in}}^2$-per-iteration compute class, and KSP's only constant is the schedule length $(S+1)K + K_{cg}$, which is why the schedule reads as a dial. Measured runtime is in Fig. 9.

Remark 1 (Aggregation)**.** *The additive, squared aggregation in (3) is a modeling choice, standard and matched to the quadratic loss; the* per-block *tightness statement is what is guaranteed. Any convex nondecreasing aggregation of the per-block gauges yields a valid convex relaxation.*

Remark 2 (Cap count: norm vs. prox)**.** *In the* binding *case of the norm water-filling (5) at most $N-1$ gates are capped (in the slack case $\boldsymbol{\theta}^\star$ is already integral with up to N ones); in the prox (6) up to N gates can be capped* with the remainder zeroed *— the integral regime. The distinction matters: integrality is a property of the prox (and the fixed point), not of the bare norm, and is exactly what Cor. 1 certifies.*

Proposition 4 (Support regimes)**.** *In the binding case, the gate support of the KSP prox satisfies $N \leq \|\boldsymbol{\theta}^\star\|_0 \leq \min(M, \|\boldsymbol{z}\|_0)$. For $(N, M) = (2, 4)$ the regimes $\|\boldsymbol{\theta}^\star\|_0 \in \{2, 3, 4\}$ — characterized exactly as {2 capped + 2 zeroed (integral)}, {≤1 capped, 1 zeroed}, {≤1 capped, 0 zeroed} —*

match, by support cardinality, the three cardinalities $\{2, 3, 4\}$ that the e_{N+1} prox must consider (Kübler et al., 2025; Liu et al., 2025). The correspondence is one of structure*, not of algorithm: our prox returns a single relaxed point resolved in closed form per regime, whereas the e_{N+1} prox resolves its 3- and 4-sparse candidates iteratively. All three routes act on the same mask polytope by different means (Fig. 6); the unification is at the polytope level, not the penalty level. (Full case analysis in App. B.2.2.)*

B.2.1 Full proofs, and the rounding decomposition

Remark 3 (Rounding decomposition)**.** *Let $\widehat{\boldsymbol{W}}$ minimize F_λ, $\boldsymbol{\Theta}^\star$ its hardened mask, Π the projection onto $\boldsymbol{\Theta}^\star$, and $\bar{\boldsymbol{W}}$ the refit output. Then $f(\bar{\boldsymbol{W}}) \le f(\Pi\widehat{\boldsymbol{W}})$, and the projection error $\left\|\Pi\widehat{\boldsymbol{W}} - \widehat{\boldsymbol{W}}\right\|_F^2$ is bounded by a sum of dropped-coordinate terms, each controlled by $\min(\theta^\star_{ij}/c_j,\, 1)^2\, \widehat{z}_{ij}^2$, with c_j the effective shrinkage and $\widehat{\boldsymbol{Z}} := \widehat{\boldsymbol{W}} - (\widehat{\boldsymbol{W}} - \boldsymbol{W}_0)\boldsymbol{H}/L$ the prox input at the fixed point; the error therefore vanishes on certified blocks. (Statement with constants, and proof, below; the first inequality is immediate from the definition of $\bar{\boldsymbol{W}}$.)*

This is what the algorithm uses: the refit never increases the layerwise loss relative to naive projection and rounding is free on certified blocks; Prop. 2 shows that hardening and refitting $\widehat{\boldsymbol{W}}$ attains the combinatorial optimum $\min_{\mathcal{C}_{N:M}} f$ in the diagonal regime. Alg. 2 states the vectorized block prox that Thm. 1 licenses; the proofs follow.

Algorithm 2 BlockProx: weighted prox of $\lambda \left\|\sqrt{\boldsymbol{\beta}} \circ \cdot\right\|_{(N)}^2$ (Thm. 1), vectorized over all blocks of a layer

Require: $\boldsymbol{z}$ per block; pattern (N, M); λ; weights $\boldsymbol{a}, \boldsymbol{\beta}$; $\quad c_j \leftarrow 2\lambda\beta_j/a_j$
if $\|\boldsymbol{z}\|_0 \le N$ **then return** $v_j = z_j/(1 + c_j)$ on $\mathrm{supp}(\boldsymbol{z})$, else 0
end if
$t_j^{\mathrm{cap}} \leftarrow \sqrt{a_j c_j}|z_j|/(1 + c_j)$; $\quad t_j^0 \leftarrow \sqrt{a_j c_j}|z_j|/c_j$ ▷ $2M$ breakpoints
sort breakpoints descending; scan segments: on each, budget $= \#\mathrm{capped} + A/\tau - C$; solve for τ, accept if τ lies in the segment ▷ $O(M \log M)$; $\boldsymbol{\theta}^\star$ unique, τ need not be
$\theta_j \leftarrow \mathrm{clip}_{[0,1]}(\sqrt{a_j c_j}|z_j|/\tau - c_j)$; $\quad$ **return** $v_j \leftarrow z_j\, \theta_j/(\theta_j + c_j)$

Proof of Theorem 1(ii). Uniqueness. The objective is a strictly convex quadratic ($a_j > 0$) plus a convex function (Thm. 1 composed with a linear map), hence strictly convex in $\boldsymbol{v}$. (The gate may be non-unique only off $\mathrm{supp}(\boldsymbol{z})$, where we set it to 0.)

Two-stage exact reduction. Write the joint problem $\min_{\boldsymbol{v},\boldsymbol{\theta}\in\Delta_{N,M}} \sum_j \left[\frac{a_j}{2}(v_j - z_j)^2 + \lambda\beta_j v_j^2/\theta_j\right]$, jointly convex by Lemma 1. Minimizing exactly in v_j for fixed $\boldsymbol{\theta}$: $a_j(v_j - z_j) + 2\lambda\beta_j v_j/\theta_j = 0$ gives $v_j = z_j\theta_j/(\theta_j + c_j)$, $c_j = 2\lambda\beta_j/a_j$, with partial value (writing a, c, z, θ for the j-th quantities and using $\lambda\beta = ac/2$):

$$\frac{a}{2}\left(\frac{-cz}{\theta + c}\right)^2 + \frac{ac}{2} \cdot \frac{z^2\theta}{(\theta+c)^2} = \frac{ac\, z^2}{2} \cdot \frac{c+\theta}{(\theta+c)^2} = \frac{a_j c_j z_j^2}{2(\theta_j + c_j)}.$$

Minimizing the reduced convex, coordinate-wise nonincreasing $h(\boldsymbol{\theta}) = \sum_j \frac{a_j c_j z_j^2}{2(\theta_j + c_j)}$ over $\Delta_{N,M}$ exactly (not alternating) yields the joint — hence global — minimizer.

Slack case. If $\nu = 0$, stationarity $-\frac{a_j c_j z_j^2}{2(\theta_j + c_j)^2} + \mu_j - \xi_j = 0$ forces, for $z_j \neq 0$, $\mu_j > 0$ and $\theta_j = 1$; feasibility then requires $\|\boldsymbol{z}\|_0 \le N$, and conversely this point minimizes each term separately over $[0, 1]$.

Binding case. For $\|\boldsymbol{z}\|_0 > N$ the budget binds, $\nu > 0$. Interior: $\frac{a_j c_j z_j^2}{2(\theta_j + c_j)^2} = \nu$, i.e., $\theta_j = \sqrt{a_j c_j}|z_j|/\tau - c_j$ with $\tau := \sqrt{2\nu}$. Cap active iff this expression ≥ 1 (i.e., $\tau \le \sqrt{a_j c_j}|z_j|/(1 + c_j)$, with multiplier $\mu_j = \frac{a_j c_j z_j^2}{2(1+c_j)^2} - \nu \ge 0$); floor active iff $\sqrt{a_j c_j}|z_j| \le c_j\tau$, i.e., iff $\rho_j \le \tau$ (multiplier

$\xi_j = \nu - \frac{a_j z_j^2}{2c_j} \geq 0$). This is (6) and the selection law. Between consecutive sorted breakpoints the budget equation has the form $\#\{\text{capped}\} + A/\tau - C$ with $A = \sum_{\text{interior}} \sqrt{a_j c_j}|z_j|$, $C = \sum_{\text{interior}} c_j$: continuous, nonincreasing, strictly decreasing while any coordinate is interior; the sorted scan finds an admissible τ in $O(M \log M)$ (unique unless no coordinate is interior, in which case any point of the admissible interval yields the same $\boldsymbol{\theta}^\star$). Signs are preserved. For the ranking write $|v_j^\star| = \rho_j \sqrt{c_j/a_j}\, \phi_j$ with $\phi_j = \big(\min(1 - \tau/\rho_j,\, 1/(1+c_j))\big)_+$: at fixed τ and constant c_j, ϕ_j is nondecreasing in ρ_j, so $\theta_j^\star$ and $\sqrt{\beta_j}|v_j^\star|$ are ρ-monotone, while the raw $|v_j^\star|$ carries the extra factor $\sqrt{c_j/a_j}$ and is ρ-monotone only when that factor is constant. □

Proof of Corollary 1. Integrality means a τ exists with $\theta_j = 1$ on the keep-set $\mathcal{K}$ ($\tau \leq \sqrt{a_j c_j}|z_j|/(1+c_j) = \rho_j c_j/(1+c_j)$) and $\theta_j = 0$ off it ($\tau \geq \sqrt{a_j c_j}|z_j|/c_j = \rho_j$); such τ exists iff $\min_{\mathcal{K}} \rho_j c_j/(1+c_j) \geq \max_{\mathcal{K}^c} \rho_j$, and $|\mathcal{K}| = N$ from the budget. Both steps require the budget to bind, i.e. $\|\boldsymbol{z}\|_0 > N$; if $\|\boldsymbol{z}\|_0 \leq N$ the gates are integral for a different reason (slack branch) and (7) is vacuous.

Constant c. Whenever $c_j \equiv c$ across the block — the plain case $a = \beta = 1$, $c = 2\lambda$, and equally the matched scaling $\beta_j = a_j = s_j^2$ of Prop. 5, where $c_j \equiv 2\lambda$ — both thresholds are increasing in the selection statistic, so any integral solution's keep-set is automatically its top-N, and the condition reads $\rho_{(N)}/(1+c) \geq \rho_{(N+1)}/c$, i.e. $\rho_{(N)}/\rho_{(N+1)} \geq 1 + 1/c$. In both named cases this is (7), with $\rho \propto |z|$ and $\rho \propto s|z|$ respectively.

Non-constant c. With unequal c_j the displayed inequality remains valid and is the form (12) we evaluate for dense $\boldsymbol{H}$, with (7) its constant-c specialization (App. B.2). Because $\rho_j c_j/(1+c_j) < \rho_j$, it forces every $j \in K$ to exceed every $i \notin K$ in ρ, so the keep-set is again the top-N by ρ.

Monotonicity. The right side of (7) is decreasing in λ, so at a *fixed* $\boldsymbol{z}$ the certified set grows with λ.

Losslessness. Integral gates already place exact zeros off K, so hardening (projection onto the top-N support) changes nothing on the block. □

Remark 3 with constants. With $K_{i,b}$ the kept set of block b in row i, and at the FISTA fixed point with effective parameter λ/L (L as in Alg. 1) and per-coordinate shrinkage $c_j = 2\lambda\beta_j/L$ (for the splitting step of App. C, $L \to \varrho$ throughout),

$$\left\|\Pi\widehat{\boldsymbol{W}} - \widehat{\boldsymbol{W}}\right\|_F^2 = \sum_{i,b} \sum_{j \notin K_{i,b}} \widehat{w}_{ij}^2 \leq \sum_{i,b} \sum_{j \notin K_{i,b}} \min\Big(\tfrac{\theta_{ij}^\star}{c_j},\, 1\Big)^2 \widehat{z}_{ij}^2,$$

the bound following from (6).

Proof of Remark 3. $f(\bar{\boldsymbol{W}}) \leq f(\Pi\widehat{\boldsymbol{W}})$ because $\bar{\boldsymbol{W}}$ minimizes f over all matrices supported on $\boldsymbol{\Theta}^\star$, a set containing $\Pi\widehat{\boldsymbol{W}}$ (up to CG tolerance). The displayed identity is the definition of Π; the bound uses, at the FISTA fixed point $\widehat{\boldsymbol{W}} = \operatorname{prox}_{(\lambda/L)\Omega}(\widehat{\boldsymbol{Z}})$, the formula $\widehat{w}_{ij} = \widehat{z}_{ij}\theta_{ij}/(\theta_{ij} + c_j)$ with $c_j = 2\lambda\beta_j/L$, so $|\widehat{w}_{ij}| \leq |\widehat{z}_{ij}| \min(\theta_{ij}/c_j,\, 1)$, and on certified blocks $\theta_{ij} = 0$ exactly, so the projection error vanishes there. □

B.2.2 Support regimes: full case analysis

Fig. 6 shows $\Delta_{2,4}$ and the three routes to it; the case analysis below distinguishes the three support regimes of the KSP route.

Proof of Proposition 4. Bounds. In the binding case $\sum_j \theta_j^\star = N$ with $\theta_j^\star \leq 1$ forces at least N nonzero gates; $\|\boldsymbol{\theta}^\star\|_0 \leq M$ is trivial.

Case analysis for $(N, M) = (2, 4)$, binding case ($\|\boldsymbol{z}\|_0 \geq 3$). Write n_{cap} for the number of capped gates, n_{zero} for the number of zeroed gates, $n_{\text{int}} = 4 - n_{\text{cap}} - n_{\text{zero}}$ for strictly interior ones; the budget reads $n_{\text{cap}} + \sum_{\text{interior}} \theta_j = 2$ with each interior $\theta_j \in (0, 1)$.

- $n_{\text{cap}} = 2$: the budget forces $\sum_{\text{interior}} \theta_j = 0$, so $n_{\text{int}} = 0$ and $n_{\text{zero}} = 2$: the **integral** regime, support $= 2$, characterized by Cor. 1.

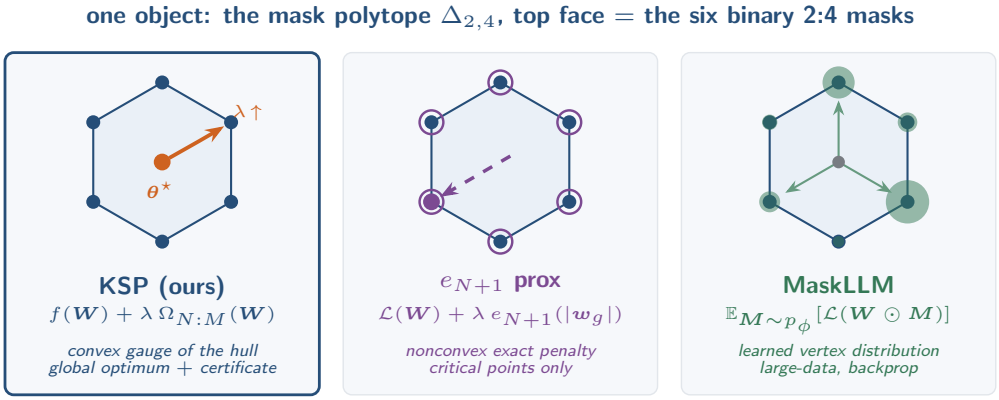


Figure 6: **One polytope, three relaxation routes.** Each panel shows the *same* object — the mask polytope $\Delta_{2,4}$, whose maximal-budget vertices are the six binary 2:4 masks (drawn here; the polytope also has the five vertices of lower budget) — reached three ways. **KSP** penalizes the convex gauge of the hull (nested level sets): gates are closed-form interior points driven to a vertex by continuation, with a per-stage global optimum and the certificate of Cor. 1. The e_{N+1} **prox** (Kübler et al., 2025; Liu et al., 2025) uses a nonconvex exact penalty vanishing on the vertex set, with critical-point guarantees only. **MaskLLM** places a learned categorical distribution on the vertices and samples it. The unification is at the polytope level, *not* the penalty level.

- $n_{\text{cap}} = 1$: interior gates sum to 1, so $n_{\text{int}} \in \{1, 2, 3\}$ and $n_{\text{zero}} = 3 - n_{\text{int}} \in \{0, 1, 2\}$. $n_{\text{zero}} = 2$ would give a single interior gate equal to 1 — a cap, contradiction. Hence $n_{\text{zero}} \in \{0, 1\}$: support $\in \{3, 4\}$.
- $n_{\text{cap}} = 0$: interior gates sum to 2 over $n_{\text{int}} = 4 - n_{\text{zero}}$ coordinates each < 1, forcing $n_{\text{int}} \geq 3$, i.e., $n_{\text{zero}} \in \{0, 1\}$: support $\in \{3, 4\}$.
- $n_{\text{cap}} \geq 3$ is impossible (budget $= 2$).

No other support size is possible, and each of $\{2, 3, 4\}$ is realized (e.g. $\boldsymbol{z} = (4, 3, 1, 1)$ at small λ gives the 4-regime, $\boldsymbol{z} = (4, 3, 2, 0.01)$ an intermediate λ the 3-regime, and any $\boldsymbol{z}$ satisfying (7) the 2-regime), so the realized supports are exactly $\{2, 3, 4\}$, with support 2 occurring iff the general certificate (12) holds (equivalently (7) when c_j is constant), support 3 iff exactly one coordinate falls below the kill threshold $\rho_j \leq \tau$, and support 4 otherwise — determined by the position of τ among the sorted breakpoints (in the integral regime τ ranges over an interval, but every point of it gives the same $\boldsymbol{\theta}^\star$, so the regime is well defined). These three regimes match, by support cardinality, the candidate cardinalities $\{2, 3, 4\}$ that the e_{N+1} prox must consider (Kübler et al., 2025; Liu et al., 2025), which resolves its 3- and 4-sparse candidates iteratively — by an inner solver in Kübler et al. (2025) and an alternating soft-thresholding loop in ProxSparse — whereas each KSP regime is a closed form (Alg. 2). □

B.3 Matched scaling and scale-equivariance

Proposition 5 (Matched scaling)**.** *Under the diagonal-curvature block model (loss weights $a_j = s_j^2$, $s_j = H_{jj}^{1/2}$), the ℓ_0-optimal block solution keeps the top-N of $s_j|z_j|$ (zeroing coordinate j costs $\frac{1}{2}s_j^2 z_j^2$). The selection statistic ρ_j of Thm. 1 induces that ranking* for every input $\boldsymbol{z}$ *iff $\beta_j \propto s_j^2$ within each block (**matched scaling**); with $\beta_j = s_j^2$ exactly, the change of variables $\boldsymbol{u} = \boldsymbol{s} \circ \boldsymbol{v}$ reduces the entire problem to the plain prox in rescaled coordinates — no new solver. Two mismatches, both predictable from ρ_j: curved loss with plain penalty ranks by $s_j^2|z_j|$ (under-prunes high-activation coordinates); plain loss with s_j-scaled penalty — the superficially natural "activation-aware penalty" — ranks by $|z_j|/s_j$ and preferentially prunes* high-*activation weights. The latter is never read* as a selection rule*: Alg. 1 hardens from the norm gates of its final iterate, and its isotropic majorization metric carries no selection interpretation (App. B.2). (Proof below.)*

Proof of Proposition 5. Equivariance. With $a_j = \beta_j = s_j^2$, substituting $\boldsymbol{u} = \boldsymbol{s} \circ \boldsymbol{v}$ gives $\frac{s_j^2}{2}(v_j - z_j)^2 = \frac{1}{2}(u_j - s_j z_j)^2$ and $\lambda s_j^2 v_j^2/\theta_j = \lambda u_j^2/\theta_j$, with the constraint set unchanged: the plain problem at $\boldsymbol{s} \circ \boldsymbol{z}$, hence $\boldsymbol{v}^\star = \boldsymbol{s}^{-1} \circ \mathrm{prox}_{\lambda\|\cdot\|^2_{(N)}}(\boldsymbol{s} \circ \boldsymbol{z})$ and $\rho_j = s_j|z_j|/\sqrt{2\lambda}$.

Consistency. The diagonal-model cost of zeroing coordinate j is $\frac{1}{2}s_j^2 z_j^2$, whose ranking coincides with that of $s_j|z_j|$ (monotone square root). By the definition of ρ_j with $a_j = s_j^2$, $\rho_j = s_j^2|z_j|/\sqrt{2\lambda\beta_j}$, which induces the ranking of $s_j|z_j|$ for all inputs iff $\sqrt{\beta_j} \propto s_j$. For "only if", note that $s_i|z_i| > s_j|z_j|$ iff $|z_i|/|z_j| > s_j/s_i$, whereas $\rho_i > \rho_j$ iff $|z_i|/|z_j| > s_j^2\sqrt{\beta_i}/(s_i^2\sqrt{\beta_j})$; the two thresholds agree for every pair exactly when $s_i/\sqrt{\beta_i} = s_j/\sqrt{\beta_j}$, and otherwise any $\boldsymbol{z}$ whose ratio lies strictly between them inverts one ordering relative to the other. *Mismatches.* $\beta_j = 1$: $\rho_j \propto s_j^2|z_j| = s_j \cdot (s_j|z_j|)$, reordering pairs whose s-ratio exceeds the inverse of their $s|z|$-ratio (under-pruning high-activation coordinates). $a_j = 1, \beta_j = s_j^2$: $\rho_j = |z_j|/(s_j\sqrt{2\lambda})$, reversing the role of the activation scale entirely. □

Proof of Proposition 2. Decoupling and one-step solution. With diagonal $\boldsymbol{H}$, the loss (1) is coordinate-separable: $f(\boldsymbol{W}) = \sum_{i,j} \frac{s_j^2}{2}(w_{ij} - w_{0,ij})^2$. Both f and the penalty are then sums of independent per-block terms, so (3) splits into $d_{\text{out}}B$ problems of exactly the form of Thm. 1 with $a_j = \beta_j = s_j^2$ and data $\boldsymbol{z}$ the corresponding block of $\boldsymbol{W}_0$; the relaxed optimum is the closed-form prox output — no iteration is needed, and FISTA converges to it. (For an iterate $\widetilde{\boldsymbol{W}}$ after a finite number of iterations, (ii)–(iii) hold once $\max_j s_j|\widetilde{w}_j - \widehat{w}_j| < \frac{1}{2}\left[(s|\widehat{w}|)_{(N)} - (s|\widehat{w}|)_{(N+1)}\right]$ in every block; the gap is positive under separation by (ii), so this holds after finitely many iterations, though not for every iteration budget.)

(i) Combinatorial optimum. Separability also solves the combinatorial problem exactly: on a fixed block support S with $|S| \le N$, $\min f$ is attained at $w_j = z_j$ on S, cost $\sum_{j \notin S} \frac{1}{2}s_j^2 z_j^2$; the optimal S keeps the top-N coordinates of $s_j^2 z_j^2$, equivalently of $s_j|z_j|$. Under the separation condition the top-N set is unique, and since blocks do not interact, per-block optimality is global optimality on $\mathcal{C}_{N:M}$.

(ii) Hardened support. With $a_j = \beta_j = s_j^2$ the shrinkage constants coincide: $c_j = 2\lambda\beta_j/a_j = 2\lambda$ for all j. By (6), $\theta_j^\star = \mathrm{clip}_{[0,1]}\left(\sqrt{2\lambda}\, s_j|z_j|/\tau - 2\lambda\right)$ (binding case; the slack case keeps all of $\mathrm{supp}(\boldsymbol{z})$ and is consistent with top-N trivially). At the block's common τ, $\theta_j^\star$ is nondecreasing in $s_j|z_j|$ and strictly increasing wherever interior, so the per-block top-N of the gates — KSP's hardened mask, ties broken by the statistic — is the top-N of $s_j|z_j|$. Alg. 1 hardens the *norm* gates $\min(s_j|\widehat{w}_j|/\tau', 1)$ of this output, τ' its own water level, rather than the prox gates, but they select the same set: on the prox support $s_j|\widehat{w}_j| = s_j|z_j|\,\theta_j^\star/(\theta_j^\star + 2\lambda)$ is strictly increasing in $s_j|z_j|$, and the norm gates cap at most $N-1$ coordinates in the binding case (Rem. 2) and equal $\mathbf{1}\{\widehat{w}_j \neq 0\}$ in the slack case. Under separation this set is exactly the unique combinatorial support of (i), for every $\lambda > 0$; only the *amount* of shrinkage, not the ranking, depends on λ.

(iii) Exactness after the refit. The refit solves $\min_{\boldsymbol{W}:\,\mathrm{supp}(\boldsymbol{W}) \subseteq \mathrm{supp}(\boldsymbol{\Theta}^\star)} f$, which under diagonal $\boldsymbol{H}$ sets $w_j = z_j$ on the mask — precisely the minimizer from (i). Hence hardening and refitting $\widehat{\boldsymbol{W}}$ attains $\min_{\mathcal{C}_{N:M}} f$. □

Proposition 6 (Recovering Wanda and magnitude)**.** *Evaluating the gates of Thm. 1 at $\boldsymbol{W}_0$ under matched scaling (statistic $s_j|(\boldsymbol{W}_0)_{ij}|$) and hardening per-block top-N yields Wanda's N:M mask (Sun et al., 2024) up to the Gram damping; with all $s_j = 1$, blockwise magnitude pruning.*

Proof. Evaluate the gates at $\boldsymbol{W}_0$ under matched scaling, so $\boldsymbol{z}$ is the corresponding block of $\boldsymbol{W}_0$. By Thm. 1 and (5) the gate is $\theta_j^\star = \min(s_j|w_{0,j}|/\tau,\, 1)$ at the block's common τ, which is nondecreasing in $s_j|w_{0,j}|$ and strictly increasing wherever interior; in the binding case at most $N-1$ coordinates are capped (Rem. 2, and the slack case is immediate), so hardening per-block top-N of the gates returns the top-N of $s_j|w_{0,j}|$, which is Wanda's score up to the Gram damping and up to ties broken identically. Setting $s_j \equiv 1$ (plain geometry) leaves $|w_{0,j}|$, the magnitude mask. □

C THE SOLVER: SUPPORT SEARCH AND THE COMPUTE DIAL

C.1 HOW THE SUPPORT IS SEARCHED

A *preconditioned* step — one that solves a ridge system in the loss's own metric — reaches supports an isotropic $1/L$ move does not. The configuration we report drives (3) with such a step, an operator-splitting loop of the kind ALPS uses (Meng et al., 2024; Boyd et al., 2011). The difference is what the loop is split around. ALPS splits around the nonconvex set $\mathcal{C}_{N:M}$, so its second block is a hard top-k projection and the joint problem it solves is nonconvex. We split around the convex penalty: per layer the loop alternates an exact ridge solve $(\boldsymbol{H} + \varrho \boldsymbol{I})^{-1}$ with the block-exact prox of Thm. 1, so the problem being split is the same convex F_{λ_t} that Alg. 1 minimizes, and both blocks are solved exactly. Two-block ADMM convergence at a fixed penalty ϱ therefore applies, and since each stage is strongly convex with a unique minimizer, both of our solvers converge to the same per-stage global optimum — where ALPS's loop, split around the nonconvex N:M set, converges but carries no global-optimality guarantee. With adaptive restart the FISTA path still converges: it is monotone and every restart begins with a proximal-gradient step, so either restarts stop and the monotone-FISTA rate covers the tail, or infinitely many sufficient-decrease steps occur, and ε-strong convexity forces convergence in both cases. At the finite iteration budgets we run, the two return different iterates, which is the only way their hardened supports can differ. The results in §3–§4 divide accordingly. Prop. 1, Thm. 1, Prop. 5 and the allocation results of §4 are properties of the geometry and hold for any solver; Cor. 1 is evaluated on the returned gates in whichever metric produced them, which for both of our steps is the general form (12); and the per-iteration counts and rates of Prop. 3 describe Alg. 1 (the rates for its variant without adaptive restart), with the splitting solver additionally paying one cached $O(d_{\text{in}}^3)$ factorization per layer.

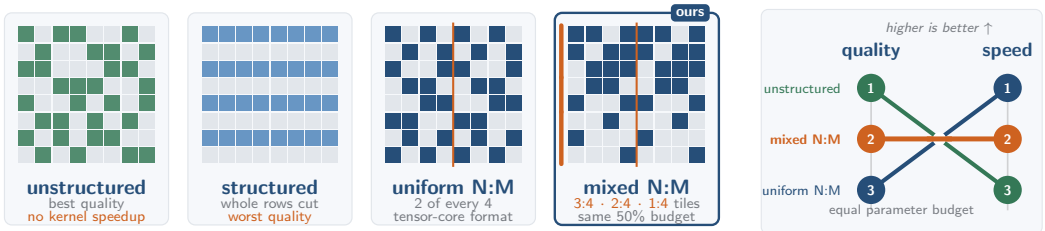
Two practical details complete the loop. It closes with a warm-started masked-CG refit, and it returns the iterate with the lowest objective on a held-out calibration split rather than the last one — a stopping rule, which selects among iterates of the same program and does not change the program being solved.

The prox control. The same preconditioned search with our block-exact prox replaced by ALPS's hard top-k projection reproduces ALPS, at 23.35 against 23.38 on the held-out split at 3B, a difference of 0.1%. What moves the KSP row of Tab. 1 is the geometry: the preconditioned search is what lets it be reached, and the prox is the source of the gain.

Which configuration is chosen. The search is run in four candidate settings at each model's seed count (Tab. 5), and the candidate with the lowest mean perplexity on a selection slice disjoint from every reported column is the one reported. The rule was declared before the deciding runs, is applied by a script, and never sees a reported metric; ALPS receives the same courtesy over its penalty parameter. With ϱ fixed per layer, the ridge solve is one cached $O(d_{\text{in}}^3)$ Cholesky factorization per layer, amortized over that layer's iterations, which is what makes the search feasible at 7B and at 70B.

C.2 THE COMPUTE DIAL, AND THE REDUCED BUDGET

The schedule is a compute budget rather than a recipe, so the same formulation is available at other budgets without changing a line of the algorithm; several controls were run at a reduced one, as their captions state. Those controls vary a selector, a mask or an allocation with everything else held fixed, so a cheaper solver changes the level of every arm and not the comparison being made. Tab. 4 lists the settings side by side.



(a) Four patterns, one budget.

(b) Approximation quality and hardware efficiency disagree.

Figure 7: **Sparsity patterns at one 50% budget.** a Grey cells are pruned. Unstructured pruning attains the lowest error but earns no tensor-core speedup; structured pruning removes whole rows; uniform N:M keeps N of every M consecutive weights, the pattern with first-class hardware support; mixed N:M (§4) spends the same budget unevenly across tiles, priced by their water levels. b Approximation quality is ordered by Prop. 7; hardware efficiency, measured for uniform and mixed N:M in §5.3, runs the other way, so mixing is a Pareto move on the speed–accuracy frontier.

Table 4: **The cost–quality dial** on Llama-3.2-3B at 2:4, all rows at 512×2048 calibration on one GPU (WikiText-2 PPL ↓). The three KSP rows are one objective at three budgets, the last adding the stronger support search of App. C; SparseGPT and ALPS are shown at the same calibration for reference. Wanda is a mean over 3 calibration seeds; the other rows are single runs (Tab. 5). *The gates are evaluated on the *damped* Gram, so this row is Wanda's ranking up to that damping (Prop. 6) and sits a little below the Wanda-masked 20.44 of Tab. 11.

Configuration	Time	PPL ↓	Note
Wanda	89 s	35.35	the zero-iteration ranking, no weight update
SparseGPT	226 s	21.11	greedy solver
ALPS	1570 s	18.20	top-k projection search
KSP, no iterations	289 s	20.38	that ranking, hardened, then refit*
KSP, reduced budget	397 s	19.79	the budget used by the controls
KSP, reported budget	2200 s	**17.93**	stronger search, same objective

Along the KSP rows of Tab. 4 the objective is fixed and only the optimization budget changes: from Wanda, the 200 s to reach the zero-iteration setting buys 15.0 perplexity and the next 108 s buys 0.59, so a single gate evaluation followed by the refit already closes most of the gap from Wanda — Prop. 6 made quantitative. The preconditioned search is not a further point on that axis but a different move through the support space, at 1803 s more than the reduced budget for 1.86 perplexity and 2.0 accuracy points at 3B.

Every calibration-based method pays for the forward pass that builds $\boldsymbol{H}$; what KSP adds on top of it is solver time, and Tab. 4 prices that directly.

D Mixed Sparsity: Allocator and Full Proofs

Fig. 7a places mixed N:M among the patterns available at one budget, and b shows why it is the interesting one: the orderings by approximation quality and by hardware efficiency disagree.

D.1 Unconstrained mixing collapses to unstructured sparsity

Proposition 7 (Collapse)**.** *Let blocks $\{g_b\}_{b=1}^{B}$ partition the coordinates and let the per-block budgets be free under a global budget: $\mathcal{C}_{\text{free}}(\kappa) := \bigcup_{\boldsymbol{n}:\,\sum_b n_b \le \kappa} \prod_b \{\|\boldsymbol{v}_{g_b}\|_0 \le n_b\}$. Then $\mathcal{C}_{\text{free}}(\kappa) = \{\boldsymbol{v} : \|\boldsymbol{v}\|_0 \le \kappa\}$, and identically on the relaxation side: minimizing (14)'s objective jointly over $(\boldsymbol{\theta}, \boldsymbol{n})$ with free $\boldsymbol{n}$ yields the* global *norm $\|\boldsymbol{v}\|_{(\kappa)}^2$ over the whole vector. Let $\mathcal{P}$ be a family of admissible patterns, $n(p)$ the per-block budget of pattern p, and $T(b)$ the tile containing*

block b. Define the tile-uniform mixed family under the parameter-budget *cost model ($\gamma_T = B_T$):* $\mathcal{C}_{\mathrm{mix}}(\mathcal{P}) := \bigcup_{p \in \mathcal{P}^{|\mathcal{T}|}:\, \sum_T B_T\, n(p_T) \le \kappa} \{\boldsymbol{v} : \|\boldsymbol{v}_{g_b}\|_0 \le n(p_{T(b)})\ \forall b\}$. *Then, at equal parameter budget κ,*

$$\min_{\mathcal{C}_{N:M}} f \;\ge\; \min_{\mathcal{C}_{\mathrm{mix}}(\mathcal{P})} f \;\ge\; \min_{\|\cdot\|_0 \le \kappa} f, \tag{13}$$

and the same ordering holds for the three relaxed values $F^\star_\lambda$, where the mixed relaxed penalty Ω_{mix} is (14)*'s objective minimized jointly over $(\boldsymbol{\theta}, \boldsymbol{n})$ subject to the tile-coupled budget:* ***mixed N:M error is sandwiched between uniform N:M and unstructured*** *for any family $\mathcal{P}$ of admissible patterns containing the uniform one. (With general encoding costs $\gamma_T \neq B_T$, the budget measures cost units rather than parameters and the parameter-count nesting can fail; the sandwich is a statement about the parameter-budget model, while the allocator* (9) *handles general costs.) The value of mixing is therefore speed-per-accuracy on real kernels. The first inequality collapses to an equality when $\mathcal{P} = \{N{:}M\}$.*

The sandwich (13) places mixed N:M inside a window whose far end is unreachable by hardware, so the question is how much of it is recovered per unit of *realized latency* — which is why the allocation experiments of §5.3 are latency-matched. The equal-level characterization of Thm. 2 is the parameter-budget optimum of the continuous relaxation; the latency-priced allocation runs the same knapsack on measured costs.

D.2 From the continuous allocation to executable patterns: a worked example

The V:N:M format. Real kernels admit granularities beyond 2:4. A V:N:M *kernel tile* (Castro et al., 2023) — not to be confused with the allocator's tiles of §4 — is $V \times M$ and prunes in two stages: the same four of the M columns are retained for all V rows, and inside them each row keeps a 2:4 pattern, so every row ends with 2 of M nonzeros. The shared column set amortizes the metadata over V rows, and the per-row pattern is what the 2:4 tensor cores execute. What matters here is that such formats, together with dense kernels, supply an admissible *family* of patterns on real hardware rather than the single 2:4 point, which is what makes a mixed allocation executable at all.

Consider one tile of three $M{=}4$ blocks with (absolute) weights $(1.6, 1.2, 0.9, 0.4)$, $(1.5, 1.1, 0.7, 0.5)$ and $(1.4, 1.0, 0.8, 0.3)$, and an illustrative admissible set $\mathcal{P} = \{1{:}4, 2{:}4, 3{:}4, 4{:}4\}$ with parameter cost $\gamma_T = B_T = 3$ per budget unit. Summing the value functions of Prop. 8 gives

$$G_T(1) = 43.50, \quad G_T(2) = 21.75, \quad G_T(3) = 14.745, \quad G_T(4) = 12.86,$$

and G_T is convex and decreasing between these points. Suppose the dual price returned by Alg. 3 places this tile's continuous optimum at $n^\star_T = 2.37$, where $G_T(2.37) = 18.354$ and the marginal value $-G'_T(2.37) = \sum_b \tau_b^2(2.37)$ equals the common price $\mu\gamma_T$. No kernel executes 2.37 survivors per block, so the neighbouring executable choices are 2:4 (cost 6, value 21.75) and 3:4 (cost 9, value 14.745): rounding down costs 3.40 in penalty value and frees 1.11 cost units, rounding up gains 3.61 for 1.89. The multiple-choice knapsack then solves the discrete choice over all tiles exactly, with the closed-form values $G_T(n(p))$ and the costs as its coefficients; it need not pick a neighbour of $n^\star_T$ for every tile, and when its costs are $C_T(p) = \gamma_T n(p)$, as here, the continuous optimum lower-bounds its value, so the gap between the two reports what the discretization costs — a computable by-product, since both values are closed-form.

D.3 The allocator and the proofs of §4

The allocator solves (9) by bi-level water-filling; we state it, then prove the two results §4 relies on.

For real $n \in (0, M]$ the fractional k-support norm of §4 is

$$\|\boldsymbol{v}\|_{(n)} := \Big(\min_{\boldsymbol{\theta} \in \Delta_{n,M}} \textstyle\sum_j v_j^2/\theta_j\Big)^{1/2}. \tag{14}$$

Proposition 8 (Fractional k-support norm)**.** *$\|\cdot\|_{(n)}$ is a norm for every real $n \in (0, M]$, computed by the water-filling* (5) *with budget n; it interpolates the integral family, with $\|\cdot\|_{(M)} = \|\cdot\|_2$ and*

Algorithm 3 MIXEDALLOC: exact allocation by bi-level water-filling (dual decomposition of (9))

1: **Require:** per-block inputs $\{\boldsymbol{v}_b\}$; tiles $\mathcal{T}$; costs γ_T; budget κ; tolerance ϵ; box floor $n_{\min} = \min_{p\in\mathcal{P}} n(p) > 0$ with $\sum_T \gamma_T n_{\min} < \kappa$ (the fractional norm is defined only on $(0, M]$); $\mu_{\max}$ any price at which the total budget falls below κ

2: **outer bisection on** $\mu \in [0, \mu_{\max}]$: ▷ dual function is concave and 1-D

3: **for** each tile T (parallel) **do**

4: **inner 1-D minimization on** $n_T \in [n_{\min}, M]$ of the convex $G_T(n_T) + \mu\gamma_T n_T$, by bisection on the *subdifferential* inclusion $\mu\gamma_T \in -\partial G_T(n_T)$ ▷ each evaluation = B_T sorted water-fillings, Thm. 1

Note: $\tau_b^2(\cdot)$ jumps to 0 at $n = \|\boldsymbol{v}_b\|_0$, so the equation $\sum_b \tau_b^2 = \mu\gamma_T$ may have no root at an *interior* minimizer; bisecting on the inclusion rather than the equation returns the correct kink.

5: **end for**

6: adjust μ until $\sum_T \gamma_T n_T(\mu) = \kappa$ (or boxes absorb the budget); for $\mu > 0$ each $n_T(\mu)$ is unique and continuous in μ (a kink holds n_T fixed over an interval of μ, never the reverse), so $\sum_T \gamma_T n_T(\mu)$ is continuous and nonincreasing and the bisection meets κ to tolerance; only at $\mu = 0$ are minimizers intervals, and any point within budget may be returned; **return** $\boldsymbol{n}^\star$, levels $\{\tau_b\}$

Cost: $O\big(d \log M \cdot \log^2(1/\epsilon)\big)$, with d the number of weights the tiles cover; deterministic, no solver reruns. Output is the *continuous* allocation. Mapping it to executable patterns is the multiple-choice knapsack $\min_{p\in\mathcal{P}^{|\mathcal{T}|}} \sum_T G_T(n(p_T))$ s.t. $\sum_T C_T(p_T) \le \kappa$, one pattern per tile from the admissible set $\mathcal{P}$ with total cost $C_T(p_T)$ ($= \gamma_T n(p_T)$ under parameter cost, measured otherwise), solved by dynamic programming over tiles with the budget discretized to the kernel's granularity — exact when the costs lie on that grid, as parameter costs do; measured costs are rounded up to it.

$\|\boldsymbol{v}\|_{(n)} = \|\boldsymbol{v}\|_1 / \sqrt{n}$ *for* $n \le 1$. *The* value function $V_{\boldsymbol{v}}(n) := \|\boldsymbol{v}\|_{(n)}^2$ *is convex and nonincreasing in* n*, continuously differentiable on the binding range* $n < \|\boldsymbol{v}\|_0$*, with*

$$V'_{\boldsymbol{v}}(n) \;=\; -\tau(n)^2; \tag{15}$$

at the kink $n = \|\boldsymbol{v}\|_0$ *its subdifferential is* $[-\min_{j\in\mathrm{supp}(\boldsymbol{v})} v_j^2,\ 0]$*, and* $V_{\boldsymbol{v}} \equiv \|\boldsymbol{v}\|_2^2$ *for* $n \ge \|\boldsymbol{v}\|_0$.

Proof of Proposition 8. Norm and water-filling. The proofs of Thm. 1 use integrality of the budget nowhere: homogeneity, $\ge \|\cdot\|_2$, convexity via Lemma 1 and partial minimization, and the KKT/ water-filling analysis hold with real n, the index range of (10) becoming $r \in \{0, \dots, \lceil n\rceil - 1\}$ and the cap count at most $\lceil n\rceil - 1$, with the slack case $\|\boldsymbol{v}\|_0 \le n$ unchanged. *Monotonicity in* n: $\Delta_{n,M} \subseteq \Delta_{n',M}$ for $n \le n'$. *Corners*: $n = M$ gives $\boldsymbol{\theta} = \mathbf{1}$ feasible, value $\|\boldsymbol{v}\|_2^2$; for $n \le 1$ the cap is inactive at the optimum ($\theta_j = |v_j|/\tau \le 1$ iff $|v|_{(1)} \le \tau = \|\boldsymbol{v}\|_1 / n$, implied by $n \le 1$), giving value $\tau\|\boldsymbol{v}\|_1 = \|\boldsymbol{v}\|_1^2 / n$. *Dual*: as in App. B.1, $\Omega^*(\boldsymbol{u})^2 = \max_{\boldsymbol{\theta}\in\mathrm{ext}\,\Delta_{n,M}} \sum_j \theta_j u_j^2$; the extreme-point analysis of Lemma 2 changes in exactly one place — with a tight non-integer budget, one fractional coordinate equal to $n - \lfloor n\rfloor$ is forced and permitted — so the maximum places ones on the top $\lfloor n\rfloor$ coordinates of u_j^2 and the fraction on the next, giving $\Omega^*(\boldsymbol{u})^2 = \sum_{i\le\lfloor n\rfloor} |u|_{(i)}^2 + (n - \lfloor n\rfloor)\,|u|_{(\lfloor n\rfloor+1)}^2$. *Value function*: $V_{\boldsymbol{v}}(n)$ is the infimal projection over $\boldsymbol{\theta}$ of a jointly convex function on the jointly convex set $\{(\boldsymbol{\theta}, n) : \boldsymbol{\theta} \in [0,1]^M, \sum_j \theta_j \le n\}$, hence convex in n; nonincreasing by monotonicity. *Envelope*: let $\nu^\star$ be an optimal multiplier of the budget at n in the binding range. For any n',

$$\begin{aligned} V(n') \;&=\; \min_{\boldsymbol{\theta}\in[0,1]^M,\ \sum\theta\le n'} \sum_j v_j^2/\theta_j \\ &\ge\; \min_{\boldsymbol{\theta}\in[0,1]^M} \Big[\sum_j v_j^2/\theta_j + \nu^\star\big(\textstyle\sum_j \theta_j - n'\big)\Big] \;=\; V(n) + \nu^\star\,(n - n'), \end{aligned}$$

the last equality by strong duality at n (the problem is convex with Slater point $\boldsymbol{\theta} = (n/2M)\mathbf{1}$) and complementary slackness; hence $-\nu^\star \in \partial V(n)$, and stationarity at an interior support coordinate (one always exists, since capping all of $\mathrm{supp}(\boldsymbol{v})$ would give $\sum_j \theta_j \ge \|\boldsymbol{v}\|_0 > n$) pins $\nu^\star =$

$v_j^2/\theta_j^2 = \tau^2$. Differentiability follows from the closed form: on each interval of n with a constant number r of capped coordinates, $V(n) = \sum_{i\le r} |v|_{(i)}^2 + S_r^2/(n-r)$ with $S_r := \sum_{i>r} |v|_{(i)}$ and $\tau = S_r/(n-r)$, so $V'(n) = -S_r^2/(n-r)^2 = -\tau(n)^2$; continuity of $\tau(\cdot)$ across the breakpoints (the budget equation is continuous and strictly decreasing in τ on this range) gives C^1. For $n \ge \|\boldsymbol{v}\|_0$ the gates $\mathbf{1}\{v_j \ne 0\}$ are feasible and optimal, so $V \equiv \|\boldsymbol{v}\|_2^2$ there and the right derivative at the kink is 0; the left derivative is $\lim_{n\uparrow\|\boldsymbol{v}\|_0} -\tau(n)^2 = -\min_{j\in\mathrm{supp}(\boldsymbol{v})} v_j^2$, because $\tau(\cdot)$ is continuous and the budget equation at $\tau = \min_{j\in\mathrm{supp}(\boldsymbol{v})} |v_j|$ gives exactly $\|\boldsymbol{v}\|_0$. By convexity the subdifferential at the kink is the interval between the two. □

Proof of Proposition 7. Combinatorial. "$\subseteq$": for $\boldsymbol{v}$ in the union, $\|\boldsymbol{v}\|_0 = \sum_b \|\boldsymbol{v}_{g_b}\|_0 \le \sum_b n_b \le \kappa$ since the blocks partition the coordinates. "$\supseteq$": given $\|\boldsymbol{v}\|_0 \le \kappa$, take $n_b := \|\boldsymbol{v}_{g_b}\|_0$. *Relaxation.* Jointly over $(\boldsymbol{\theta}, \boldsymbol{n})$ with constraints $\boldsymbol{\theta} \in [0,1]^d$, $\sum_{j\in g_b} \theta_j \le n_b$, $\sum_b n_b \le \kappa$: eliminating $\boldsymbol{n}$ by setting $n_b = \sum_{j\in g_b} \theta_j$ (optimal since the objective does not depend on $\boldsymbol{n}$ and this choice is least restrictive) leaves $\boldsymbol{\theta} \in [0,1]^d$, $\sum_j \theta_j \le \kappa$ — the global capped simplex — whose variational value is $\|\boldsymbol{v}\|_{(\kappa)}^2$ by (14) applied with block size d. *Sandwich.* $\mathcal{C}_{N:M} \subseteq \mathcal{C}_{\mathrm{mix}}(\mathcal{P}) \subseteq \mathcal{C}_{\mathrm{free}}(\kappa) = \{\|\cdot\|_0 \le \kappa\}$ whenever the uniform pattern belongs to $\mathcal{P}$ and $\gamma_T = B_T$ (the tile budgets then sum to a parameter count: $\sum_b n_{T(b)} = \sum_T B_T n_T \le \kappa$); minima over nested sets are ordered. For the relaxed values, the gate polytopes are nested (products of capped simplices $\subseteq$ tile-coupled polytope $\subseteq$ global capped simplex, the latter inclusion because any mixed-feasible $\boldsymbol{\theta}$ has $\sum_j \theta_j = \sum_b \sum_{g_b} \theta_j \le \sum_T B_T n_T \le \kappa$), so the three penalties are ordered *pointwise*, $\Omega_{\mathrm{unif}} \ge \Omega_{\mathrm{mix}} \ge \|\cdot\|_{(\kappa)}^2$, and hence so are the penalized minima $F_\lambda^\star$. □

Proof of Theorem 2. Each G_T is a finite sum of convex nonincreasing functions (Prop. 8), so (9) is a convex separable program whose constraints are all affine; with each G_T finite and convex on an open set containing the box (write $V_b := V_{\boldsymbol{v}_b}$, and extend it by $\|\boldsymbol{v}_b\|_2^2$ for $n \ge M$), KKT is necessary and sufficient whenever the program is feasible, $\sum_T \gamma_T n_{\min} \le \kappa$. With multiplier $\mu \ge 0$ for the budget and box multipliers, stationarity reads $0 \in \partial G_T(n_T^\star) + \mu\gamma_T + \mathcal{N}_{[n_{\min},M]}(n_T^\star)$. By the Moreau–Rockafellar sum rule, $\partial G_T(n) = \sum_{b\in T} \partial V_b(n)$, where Prop. 8 gives $\partial V_b(n) = \{-\tau_b^2(n)\}$ for $n < \|\boldsymbol{v}_b\|_0$, $\{0\}$ for $n > \|\boldsymbol{v}_b\|_0$ (consistent with the convention $\tau_b := 0$ for $n \ge \|\boldsymbol{v}_b\|_0$, which takes the right derivative), and the kink interval $[-\min_{j\in\mathrm{supp}(\boldsymbol{v}_b)} (\boldsymbol{v}_b)_j^2, 0]$ at equality. Away from the finitely many kink points this yields the stated equation, with the inequalities at the box from the normal cone. At a kink strictly above $n_{\min}$ — including one at M, where $G_T'(M^+) = 0$ — it yields the stated inclusion; at a kink on the lower box the normal cone leaves only $\mu\gamma_T \ge -G_T'(n_{\min}^+) = \sum_b \tau_b(n_{\min})^2$, the stated '$\le$' condition. Uniqueness of the per-tile root: each $\tau_b(\cdot)$ is nonincreasing and strictly decreasing on its binding range (raising the budget lowers the water level: the budget equation $\sum_j \min(|v_j|/\tau, 1) = n$ pairs a larger n with a smaller τ), so $\sum_b \tau_b^2(\cdot)$ is strictly decreasing wherever any block is binding, and the inner bisection of Alg. 3 is well posed. The outer dual function $\mu \mapsto \min_{\boldsymbol{n}\,\in\,\mathrm{box}} \sum_T [G_T(n_T) + \mu\gamma_T n_T] - \mu\kappa$ is concave and one-dimensional; strong duality (affine constraints, feasibility) makes the bisection exact to tolerance.

Joint convexity of the full mixed program. The program $\min_{\boldsymbol{W},\boldsymbol{\theta},\boldsymbol{n}} f(\boldsymbol{W}) + \lambda \sum_j \beta_j w_j^2/\theta_j$ subject to $\boldsymbol{\theta} \in [0,1]^d$, $\sum_{j\in g_b} \theta_j \le n_{T(b)}$, $\sum_T \gamma_T n_T \le \kappa$, $\boldsymbol{n} \in [n_{\min}, M]^{|\mathcal{T}|}$ has a jointly convex objective (quadratic plus perspectives, Lemma 1) and linear constraints, hence is convex; the relaxation phase therefore contains no integer variables. □

E Experimental Details and Additional Results

This appendix gives the experimental configuration fixed across every run (below and App. E.1) and the results deferred from §5 (App. E.2).

Environment. All runs, including the 70B models, use a single RTX PRO 6000 Blackwell (sm_120, 96 GiB); every latency figure is captured under CUDA graphs on an otherwise idle device.

Models and calibration. Llama-3.2-1B and Llama-3.2-3B (Grattafiori et al., 2024), Llama-2-7B, Llama-2-13B and Llama-2-70B (Touvron et al., 2023b), Qwen2.5-7B (Yang et al., 2024) and Mistral-7B-v0.3 (Jiang et al., 2023), plus Llama-3.1-70B (Grattafiori et al., 2024) for the deployment check of §5.3. Calibration is C4 (Raffel et al., 2020) at sequence length 2048, 512 sequences for Llama-3.2-3B and 128 elsewhere (256 for the 70B allocation runs and Llama-3.1-70B), drawn identically for every one-shot method within a comparison; ProxSparse follows its authors' recipe (App. E.2.4), and the calibration-corpus control of Tab. 13 varies the corpus by design. Mistral-7B denotes Mistral-7B-v0.3 throughout. Seeds $\{0, 1, 2\}$ index the calibration draw; seed counts are given per model in Tab. 5 and per table in its caption.

KSP configuration. Matched scaling $\boldsymbol{\beta} \propto \operatorname{diag}\boldsymbol{H}$ (Prop. 5, strictly positive by the damping) throughout; damped Gram with $\varepsilon = 10^{-2} \cdot \frac{1}{d_{\text{in}}}\operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{X})$; shared Gram matrices across the $q/k/v$ and gate/up projections; masked-CG refit with a divergence guard; monotone FISTA with a logarithmically spaced λ schedule in mean-diagonal-curvature units. The support search, the reduced budget used by several controls and the pre-registered per-model configuration rule are in App. C.

E.1 Reproducibility record

The zero-shot mean is over ARC-Easy, ARC-Challenge, BoolQ, HellaSwag, OpenBookQA, RTE and WinoGrande, scored with lm-eval-harness (Gao et al., 2023). Continuation hyperparameters, the support-freeze rule, per-baseline commits and command lines, and the recorded per-task scores are in the released code.

E.2 Additional results

In order: the calibration protocol, the Llama-2-13B column in full, per-task accuracy, the baselines and solver budgets, the selector controls, the allocation results and deployment tables, and the cost diagnostics.

E.2.1 Protocol

Tab. 5 gives the calibration size and seed count behind every reported number.

Table 5: **Calibration protocol.** Calibration is C4 at sequence length 2048 throughout (except ProxSparse, which follows its authors' recipe, and the calibration-corpus control of Tab. 13), the same for every one-shot method within each row. Where two seed counts are given, the first is the uniform-2:4 comparison of Tab. 1 and the second the matched-allocation control of Fig. 8. Seed counts refer to the daggered rows of Tab. 1; unmarked baselines are single runs unless the Notes column says otherwise.

Model	Calibration sequences	Seeds	Notes
Llama-3.2-1B	128	3 / 3	
Llama-3.2-3B	512	1 / 3	Wanda and RIA 3; daggered rows single run
Llama-2-7B	128	3 / 3	seed-matched across daggered rows
Qwen2.5-7B	128	3 / 3	
Mistral-7B-v0.3	128	3 / 3	
Llama-2-13B	128	3	main comparison only
Llama-3.1-70B	256	1	deployment check only
Llama-2-70B	128 (256 for the frontier)	1	

MaskLLM (Fang et al., 2024) learns masks by gradient descent through the model over GPU-days and occupies a different computational regime; we compare with it at the level of the mask polytope (App. A), and numerically with *ProxSparse* (Liu et al., 2025), which also learns its mask through the model (App. E.2.4). ALPS, Pruner-Zero, MaskPro and ProxSparse are absent at 70B for resource reasons on our single device: Pruner-Zero needs a full-model backward pass (∼280 GB), MaskPro's training pipeline already peaks near the device's 96 GiB at 13B, ProxSparse likewise trains through the full model, and ALPS's reference implementation exceeded our resource budget there.

E.2.2 LLAMA-2-13B IN FULL

Tab. 6 gives the 13B column of Tab. 1 on all three corpora. KSP leads ALPS on WikiText-2 by 2.18 pooled standard deviations and on C4 by 2.1%, so the perplexity lead is not specific to WikiText-2, and the two zero-shot means sit within 0.22 pooled standard deviations of each other, with the per-task scores placing KSP ahead of ALPS on the reasoning tasks — ARC-Challenge 38.71 against 37.86, ARC-Easy 72.05 against 71.56, and HellaSwag and OpenBookQA. The reported KSP configuration ran in 3213 s against ALPS's 4658 s, 1.45× faster (Fig. 9), and 1.74× faster than ALPS's other in-band setting (5591 s); MaskPro's learned mask, at 19.1 GPU-hours, costs 21× KSP for a lower score on both metrics. Two features of this scale are worth recording: the saliency rows compress into 8.0–9.0 WikiText-2 perplexity, so the spread between saliency rules narrows here while KSP's lead over them holds; and Wanda loses WikiText-2 and C4 to magnitude (the latter by 6%) while leading it on PTB by a factor of 2.2 and on accuracy by five points — a ranking a single-corpus table would reverse.

Table 6: **Llama-2-13B at uniform** 2:4, full corpora. Perplexities ↓ on WikiText-2, PTB and C4; 0-shot is the seven-task mean ↑. Best pruned entry per column in **bold**. n is the calibration-seed count; three-seed rows are means; a dash, not measured. KSP is at the reported configuration of App. C, tuned by the same pre-registered rule as ALPS on a slice disjoint from this column.

Method	n	WT2	PTB	C4	0-shot
Dense	1	4.884	32.19	6.727	63.19
Magnitude	1	8.860	200.81	11.709	49.06
Wanda	1	8.993	91.32	12.452	54.05
RIA	1	8.406	80.86	11.466	54.71
DSnoT	1	8.839	89.76	12.223	54.08
Pruner-Zero	1	8.027	82.83	11.893	53.74
MaskPro	1	8.614	99.12	11.604	54.37
ProxSparse	1	**7.102**	—	—	56.04
SparseGPT	3	8.793	68.03	11.251	56.19
ALPS	3	8.210	**54.94**	10.386	**56.70**
KSP (ours)	3	7.985	64.42	**10.163**	56.55

E.2.3 PER-TASK ZERO-SHOT ACCURACY

Tabs. 7 and 8 open the seven-task mean of Tab. 1 into its components. On the four commonsense and science question-answering sets KSP takes ARC-Easy and OpenBookQA at four of the six models and HellaSwag and ARC-Challenge at three, with the remaining columns split between ALPS and ProxSparse. RTE and BoolQ, the two binary yes/no tasks, are the noisiest columns — on RTE's 277 examples a single item moves the score by 0.36 points — and the ones where per-model rankings are least stable, Pruner-Zero topping BoolQ at both Qwen2.5-7B and Mistral-7B. The Llama-2-13B block is complete; rows whose per-task record was not retained are reported as means in Tab. 1.

E.2.4 BASELINES, AND THE SOLVER BUDGETS BEHIND FIG. 3

MaskPro (Sun et al., 2026) learns a 2:4 mask by policy gradient from a fixed initial mask; we initialize from Wanda, so the learned mask is measured against a selector we also report, and run the published 10,000 steps. On no model does the learned mask reach KSP.

ProxSparse (Liu et al., 2025) is evaluated from the authors' released checkpoints at Llama-2-7B, Mistral-7B and Llama-2-13B (at 13B the regularized checkpoint, $\lambda_1 = 0.5$, $\lambda_2 = 0.25$), which we verified to be strict 2:4. At Llama-3.2-1B, Llama-3.2-3B and Qwen2.5-7B, where no checkpoint is released (the authors release Qwen2.5-14B but not 7B), we ran the authors' code, ported to current library versions, extracting the mask from the regularized checkpoint and applying it to the original weights as their own evaluator does; the regularization weight was selected from the authors' cross-model range on the same C4 selection slice that configures KSP and ALPS (from $\{0.2, 0.85, 5.0\}$ at Qwen2.5-7B); the selection independently picked 0.85 at all three, their published value for Llama-3.1-8B. A from-scratch run of their Llama-2-7B recipe through the same port reproduces the released checkpoint to about 1% on WikiText-2 (9.27 against 9.17) and C4 (12.31 against 12.20), and within

Table 7: **Per-task zero-shot accuracy at uniform** 2:4, Llama-3.2-1B, Llama-3.2-3B and Llama-2-7B. Best pruned entry per task and per model in **bold**, ties bolded jointly; n is the calibration-seed count and multi-seed rows are means. Mean is the seven-task average reported in Tab. 1. Rows whose per-task record was not retained are reported as means in Tab. 1; see App. E.2.3.

Method	n	BoolQ	RTE	HellaSw.	WinoG.	ARC-e	ARC-c	OBQA	Mean
Llama-3.2-1B									
DSnoT	1	58.41	52.71	28.33	50.83	37.25	18.69	14.20	37.20
MaskPro	1	60.43	52.71	28.78	50.83	39.27	18.52	14.40	37.85
ProxSparse	1	55.90	**53.43**	30.21	51.38	39.18	19.20	15.00	37.76
SparseGPT	3	62.11	53.07	32.70	55.44	47.22	21.42	16.07	41.15
ALPS	3	62.14	52.95	34.50	55.98	49.84	**23.32**	18.53	42.47
KSP (ours)	3	**62.17**	52.71	**34.65**	**56.20**	**50.45**	22.41	**18.87**	**42.49**
Llama-3.2-3B									
Wanda	3	64.24	52.83	34.56	54.80	52.30	24.80	18.53	43.15
RIA	3	62.72	52.35	34.72	55.38	53.01	25.43	18.33	43.13
DSnoT	1	64.34	52.71	34.85	55.41	52.31	25.00	19.60	43.46
Pruner-Zero	1	64.34	52.71	34.02	56.67	53.41	25.09	19.80	43.72
MaskPro	1	64.16	52.71	34.54	54.62	53.07	25.43	18.40	43.28
ProxSparse	1	57.09	53.07	39.03	58.72	50.88	25.60	18.00	43.20
ALPS	1	68.59	53.43	41.76	63.38	**62.71**	29.69	23.60	49.02
KSP (ours)	1	**70.73**	**57.40**	**42.62**	**64.80**	62.12	**29.95**	**24.20**	**50.26**
Llama-2-7B									
DSnoT	1	68.07	53.79	41.03	62.90	62.67	30.03	23.40	48.84
MaskPro	1	69.33	53.43	43.06	63.54	63.85	31.23	23.00	49.63
ProxSparse	1	68.35	57.04	**48.00**	63.85	64.18	33.45	25.20	51.44
SparseGPT	3	68.47	**59.93**	43.50	65.12	63.45	30.23	24.20	50.70
ALPS	3	**70.68**	57.40	45.99	65.35	65.22	33.02	**26.20**	51.98
KSP (ours)	3	69.81	57.04	46.30	**65.88**	**65.77**	**33.53**	25.87	**52.03**

half a point of accuracy (51.91 against 51.44). ProxSparse uses 164,833 calibration tokens and 400 gradient steps through the model, where at the 128-sequence protocol the one-shot methods use 262,144 tokens and none. At both small models it is below every saliency rule on perplexity yet within the saliency tier on accuracy. KSP leads it on accuracy at all six models where both run, and on perplexity at four — by 2.57× at 1B, 1.44× at 3B, 1.52× at Qwen2.5-7B, and at Mistral-7B.

The reduced solver budget across patterns. The solid series of Fig. 3b and c compares each reported cell with the same cell rerun at the reduced budget, everything else held fixed; Tab. 9 is that arm. The two series of those panels are five-model means: the reported cells of Tabs. 1 and 10 against this table (solid) and against the strongest baseline in each cell (dashed); the solid values are printed in the figure. The dashed baseline is ALPS in every perplexity cell at 75% and in every accuracy cell but Llama-3.2-1B at 1:4 (SparseGPT).

Zero iterations at 75%. Run with no inner iterations, KSP takes the same ranking as activation-scaled saliency (Prop. 6) and adds only the refit. Across the ten cells of Tab. 10 it sits one to two orders of magnitude below the three ranking-only rules on perplexity in nine of them, and beats SparseGPT in nine; per-cell values are in the released logs.

E.2.5 Isolating the selector

The bars of Fig. 3a. Each is a difference between two reported arms on Llama-3.2-3B at 2:4: the selector, $18.20 - 17.93 = 0.27$ (ALPS against KSP at the reported budget, Tab. 1); the search, $19.79 - 17.93 = 1.86$ (Tab. 9 against Tab. 1); the allocation, $14.50 - 12.68 = 1.82$ at matched latency and $14.55 - 11.91 = 2.64$ at matched parameter budget, both against the random dense-selection control of Tab. 13; and the refit, $35.19 - 19.48 = 15.71$ on RIA's mask (Tab. 11 against Tab. 1's RIA row).

KSP's mask is best on every metric at both scales.

Table 8: **Per-task zero-shot accuracy at uniform** 2:4, Qwen2.5-7B, Mistral-7B-v0.3 and Llama-2-13B. Conventions as in Tab. 7; the Llama-2-13B block is complete for every method in Tab. 1.

Method	n	BoolQ	RTE	HellaSw.	WinoG.	ARC-e	ARC-c	OBQA	Mean
Qwen2.5-7B									
Magnitude	1	64.59	54.87	30.14	49.96	37.37	24.23	21.80	40.42
DSnoT	1	77.49	73.65	45.29	64.33	71.34	36.43	27.00	56.50
Pruner-Zero	1	**82.20**	75.45	44.63	65.90	71.72	38.05	26.20	57.74
MaskPro	1	75.41	**76.17**	45.38	64.96	71.51	36.69	25.80	56.56
ProxSparse	1	70.24	66.79	47.97	62.51	69.70	38.31	26.60	54.59
SparseGPT	3	80.38	75.33	47.81	68.69	73.54	38.37	27.33	58.78
ALPS	3	79.15	67.99	49.00	**69.72**	73.58	38.51	**27.53**	57.93
KSP (ours)	3	82.00	75.33	**49.87**	69.59	**73.64**	**39.62**	**27.53**	**59.65**
Mistral-7B-v0.3									
Magnitude	1	61.31	51.99	49.25	63.54	67.55	32.85	25.80	50.33
DSnoT	1	69.94	54.87	44.44	64.88	65.66	32.25	23.00	50.72
Pruner-Zero	1	**75.41**	54.87	45.77	64.72	67.42	31.91	23.20	51.90
MaskPro	1	68.47	57.40	44.75	65.90	66.33	32.51	22.80	51.17
ProxSparse	1	71.99	53.79	**53.21**	64.40	**70.54**	**36.69**	**27.40**	54.00
SparseGPT	3	70.83	**64.02**	47.05	68.30	68.42	34.30	26.40	54.19
ALPS	3	72.05	63.30	48.33	68.72	68.97	35.29	27.00	54.81
KSP (ours)	3	74.92	59.57	49.02	**68.75**	70.09	35.75	27.20	**55.04**
Llama-2-13B									
Magnitude	1	64.46	53.43	48.34	60.38	62.67	31.74	22.40	49.06
Wanda	1	76.15	58.12	46.55	66.69	69.11	34.73	27.00	54.05
RIA	1	77.83	57.76	48.03	67.64	69.53	35.15	27.00	54.71
DSnoT	1	76.82	58.12	46.76	66.54	68.94	35.15	26.20	54.08
Pruner-Zero	1	77.61	54.51	46.94	65.67	70.24	33.19	28.00	53.74
MaskPro	1	76.24	57.40	47.56	68.11	68.90	36.01	26.40	54.37
ProxSparse	1	73.67	59.93	**52.91**	66.30	71.34	**39.16**	29.00	56.04
SparseGPT	3	**79.28**	**60.65**	47.94	70.11	69.95	36.32	29.07	56.19
ALPS	3	78.51	60.41	49.11	**70.38**	71.56	37.86	29.07	**56.70**
KSP (ours)	3	77.15	58.49	49.59	70.19	**72.05**	38.71	**29.67**	56.55

Table 9: **KSP at the reduced solver budget**, single seed, calibration as in Tab. 1. Same objective, same penalty weights, same exact prox and refit as the reported configuration; only the inner descent step and the stopping rule differ (App. C).

Pattern	Llama-3.2-1B	Llama-3.2-3B	Llama-2-7B	Qwen2.5-7B	Mistral-7B
	WikiText-2 PPL ↓ / mean 0-shot acc. ↑				
2:4	29.36 / 41.57	19.79 / 48.29	10.28 / 51.49	10.63 / 58.09	9.64 / 54.53
2:8	249.7 / 32.84	194.1 / 33.05	84.1 / 32.69	62.0 / 37.46	74.8 / 34.33
1:4	514.5 / 33.73	267.9 / 32.10	129.1 / 32.05	125.5 / 34.44	100.4 / 33.09

At the reduced solver budget KSP leads perplexity on every model against the selectors of Fig. 8 under matched allocation, by 4.9 pooled standard deviations at 1B. Restoring the reported solver budget inside the same allocation composes the two (stars in Fig. 8), and perplexity is first on every model.

Table 10: **The** 75% **regime**, single seed, calibration as in Tab. 1; best per column and metric in **bold**, second best underlined; PPL is WikiText-2 (↓) and Acc the mean over seven zero-shot tasks (↑); KSP is at the reported configuration of App. C. The three saliency rows sit at the random-guessing accuracy floor throughout.

		Llama-3.2-1B		Llama-3.2-3B		Llama-2-7B		Qwen2.5-7B		Mistral-7B	
	Method	PPL	Acc	PPL	Acc	PPL	Acc	PPL	Acc	PPL	Acc
2:8	Wanda	14211	31.73	2954	32.13	2792	32.48	1912	32.56	631	32.56
	RIA	20395	32.11	4923	31.97	1332	32.27	1510	32.16	427	32.79
	DSnoT	36103	31.95	1922	32.07	1800	31.95	910	32.48	456	32.69
	SparseGPT	395.8	32.88	221.8	33.54	85.4	32.91	89.1	35.61	87.0	34.12
	ALPS	163.1	33.65	132.4	**36.60**	57.1	37.18	40.3	39.25	51.2	37.39
	KSP	**147.2**	**35.85**	**122.0**	36.40	**56.8**	**37.72**	**39.7**	**39.83**	**45.2**	**37.48**
1:4	Wanda	29765	32.32	5144	32.42	3676	32.58	3555	32.44	1493	32.90
	RIA	26721	32.65	18350	32.21	1947	32.04	5757	32.14	1161	32.69
	DSnoT	30930	32.22	9151	32.51	4275	32.73	1636	32.41	1454	32.42
	SparseGPT	1005	**33.67**	313.9	32.33	142.0	32.34	206.8	32.66	125.4	32.40
	ALPS	**250.1**	33.13	192.7	35.09	80.3	35.20	**62.9**	37.46	70.0	35.89
	KSP	251.7	33.46	**165.5**	**35.84**	**77.3**	**35.89**	65.2	**37.80**	**61.5**	**36.45**

Table 11: **Mask ablation with the compensation held fixed** (2:4). Every row receives KSP's masked-CG refit, so the mask is the only free variable; seed 0 at 512×2048 (3B) and 256×2048 (70B), the 70B column being the uniform-2:4 run of Tab. 15. Format: PPL ↓ / avg. 0-shot ↑; best in **bold**, second underlined. The 3B column is at the reported solver budget, the 70B column at the reduced one. Uncompensated, RIA's mask measures $35.19/43.13$ at 3B (Tab. 1, a three-seed mean), so the refit is worth about 15.7 perplexity and 4.4 accuracy points.

Mask	Compensation	Llama-3.2-3B	Llama-2-70B
KSP	KSP masked CG	**17.93 / 50.26**	**5.430 / 64.33**
RIA	KSP masked CG	19.48 / 47.49	5.440 / 63.09
Wanda	KSP masked CG	20.44 / 47.00	5.466 / 64.26

E.2.6 Allocation: controls, corpora and architectures

Table 12: **Three corpora on Mistral-7B-v0.3**, perplexities ↓ and zero-shot accuracy ↑: the allocated KSP row is a mean over 3 calibration seeds with the allocator *recomputed per seed*; other rows are single runs, at the reduced solver budget of App. C.

Method	WT2	PTB	C4	0-shot
Dense	5.32	—	—	64.87
Wanda 2:4	11.95	68.28	16.82	50.79
RIA 2:4	11.10	63.41	15.77	52.17
SparseGPT 2:4	10.36	56.64	14.11	54.01
KSP 2:4	9.64	48.71	13.03	54.53
KSP @62.5	**6.79**	**38.07**	**10.11**	**59.31**

On Llama-2-70B (Tab. 15) adopting an allocation at all is worth -23.6% WikiText-2, with gains on all three corpora, and moving further along the frontier buys a further -14.2% there and -7.1% on C4. Whole-model throughput on Llama-3.2-3B with real weights (4096-token prefill, graph-captured) is 15,456 tok/s dense, 18,289 under uniform 2:4 ($1.183\times$) and 17,123 for KSP at 62.5% density ($1.108\times$): the allocated model keeps 59% of uniform 2:4's speed gain at a 40% lower perplexity (11.91 against 19.79), so the layer-level and whole-model pictures agree.

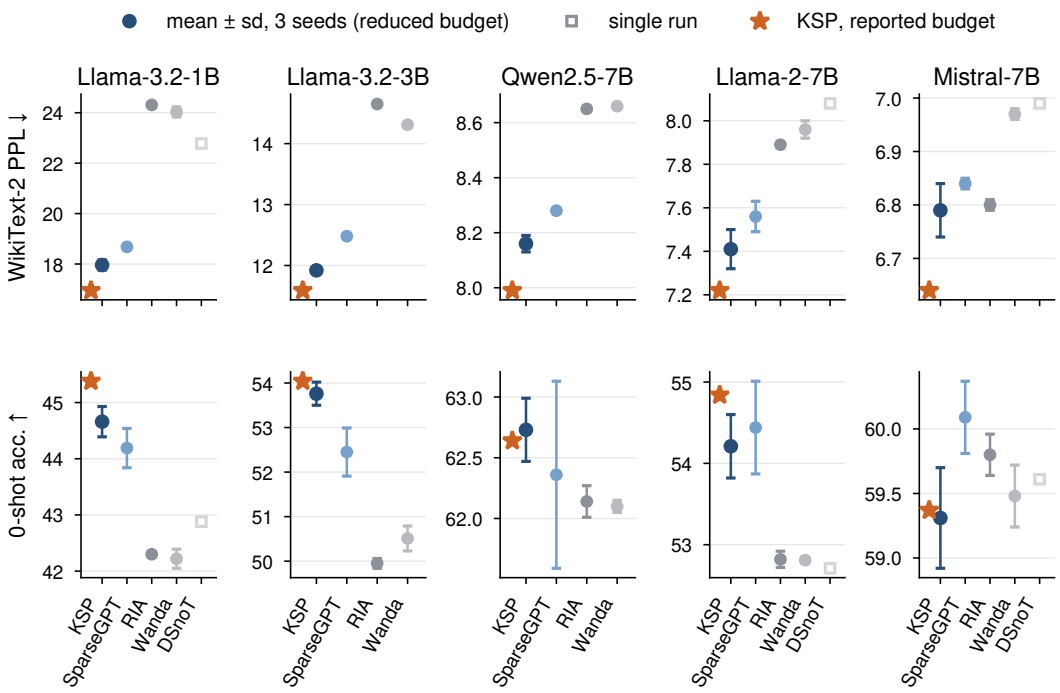


Figure 8: **Matched-allocation control across five models**, 62.5% density, WikiText-2 perplexity ↓ (top) and mean zero-shot accuracy ↑ (bottom). Filled markers are means ± sd over 3 calibration seeds with every arm at the reduced solver budget of App. C, so the selector is the only free variable; hollow squares are single runs (DSnoT). The star repeats KSP at the reported budget inside the same allocation (single seed).

Table 13: **Allocation controls**, three blocks. (i) Matched on *measured latency*, allocator recomputed per seed. (ii) Matched on *parameter budget* at 62.5% density, single runs, adding the outlier-ratio allocator OWL, whose per-layer ratios we round to the same admissible patterns ours uses. (iii) The calibration-corpus check at 62.5% density, with the allocator recomputed per corpus, which is why its C4 row sits above block (ii)'s fixed-allocation value. Means ± sd over 3 seeds where shown, Llama-3.2-3B unless stated; every KSP row uses the reduced solver budget of App. C, held constant across the arms being compared.

Control	Setting	PPL ↓	Avg. 0-shot ↑
KSP allocation	latency 108.60 ms	12.68 ± 0.68	53.42 ± 0.62
random dense-selection	latency 108.73 ms	14.50 ± 0.07	51.44 ± 0.32
KSP allocation	62.5% density	11.91	53.69
random dense-selection	62.5% density	14.55	51.65
OWL (Yin et al., 2024)	62.5% density	14.77	50.26
C4-calibrated	3B, WT2 / PTB / C4	12.31 / 17.30 / 17.18	53.90
WikiText-2-calibrated	3B, WT2 / PTB / C4	10.62 / 17.52 / 18.60	51.46

Tab. 13 gives the per-arm seed spreads behind the latency-matched control of §5.3; the allocation agreement across calibration corpora is 174/196 matrices, and the allocated 70B runs use a 7B-sourced allocation transferred unchanged, which removes the 70B allocation search entirely. On Mistral (Tab. 12) the ordering Wanda < RIA < SparseGPT < uniform KSP < allocated KSP is identical on all three corpora, and allocation is worth −30% WikiText-2, −22% PTB, −22% C4 and +4.8 accuracy over uniform 2:4.

Composition with quantization and recovery fine-tuning. In Tab. 16, INT8 per-channel weights are free on top of 2:4 for every configuration, so 2:4 plus INT8 is a no-cost compound format. At 4-bit the allocated model still beats both uniform baselines at full precision, and after identical LoRA recovery it keeps its lead on both metrics — indeed the allocated model *without* recovery beats both uniform models *with* it.

The floats behind §5.2 and §5.3. Tab. 14 is the selector-gap control: the gap between the two selectors all but vanishes once both run under our allocation, and the accuracy advantage reverses. The uniform-to-allocated step is larger still, but it moves both the size of the budget and where it is spent, which is why the latency-matched control of §5.3 is what we read as an allocation effect. Tab. 15 separates the two latency-matched comparisons behind Fig. 4a, and its lower block transfers the Llama-2-70B free-dense allocation unchanged to Llama-3.1-70B, where it returns more than twice the absolute WikiText-2 and C4 perplexity gain it returns on Llama-2-70B. Tab. 16 gives the capability scores plotted in Fig. 4b.

Table 14: **Allocation absorbs the selector gap.** The strongest competing selector of Tab. 1 — ALPS — evaluated under uniform 2:4 and then under KSP's allocation at 62.5% density. The allocated rows are means over 3 calibration seeds with the standard deviation shown; the Llama-3.2-3B uniform-2:4 rows are single runs for both arms; at Qwen2.5-7B the uniform KSP row is a single run and the ALPS row the three-seed mean of Tab. 1. Our column is at the reduced solver budget of App. C, held fixed across the comparison.

Setting	Metric	KSP	ALPS	Selector gap
Llama-3.2-3B	uniform 2:4, PPL	19.79	18.20	1.59
	allocated, PPL	11.92 ± 0.02	11.77 ± 0.02	**0.15**
	uniform 2:4, acc	48.29	49.02	-0.73
	allocated, acc	53.76 ± 0.26	53.52 ± 0.49	$+\mathbf{0.24}$
Qwen2.5-7B	uniform 2:4, PPL	10.63	10.58	0.05
	allocated, PPL	8.16 ± 0.03	8.12 ± 0.03	**0.04**

Table 15: **Latency-priced allocation on Llama-2-70B, and its transfer to Llama-3.1-70B**, 2:4 and dense kernels only. The table holds *two* latency-matched comparisons, not one common speed: free-dense against uniform 2:4 at 1.123× prefill, and latency-priced against parameter-budget allocation at 1.078×. Within the first pair, keeping the zero-gain gate/up projections dense — one line, no allocator — beats uniform 2:4 by 1.28 PPL and 1.37 accuracy points at identical speed; within the second, pricing the budget in measured latency rather than parameters is worth 0.89 PPL. Pushed to 1.053× the allocation recovers close to nine tenths of the perplexity uniform 2:4 gives up. Best pruned entry per column in **bold**; single seed, at the reduced solver budget of App. C, with 256 × 2048 calibration, so the uniform-2:4 row is not Tab. 1's 128 × 2048 70B cell. The lower block applies the Llama-2-70B free-dense rule unchanged to Llama-3.1-70B under the same protocol.

Configuration	Prefill	WT2 ↓	PTB ↓	C4 ↓	0-shot ↑
dense (measured)	1.000×	3.319	16.69	5.710	66.53
uniform 2:4	1.123×	5.430	20.78	7.729	64.33
free-dense (gate/up kept dense)	1.123×	4.148	17.52	6.340	65.70
parameter budget @62.5%	1.078×	4.610	—	—	64.99
latency-priced allocation	1.078×	3.716	17.37	5.995	**65.84**
latency-priced @1.053×	1.053×	**3.557**	**17.22**	**5.893**	65.79
Llama-3.1-70B, allocation transferred unchanged					
dense (measured)	1.000×	2.808	7.693	7.110	68.52
SparseGPT uniform 2:4	—	8.813	11.01	13.06	63.70
KSP uniform 2:4	—	8.196	10.84	12.15	63.60
KSP free-dense (transferred)	—	**5.380**	**8.669**	**9.055**	**67.29**

Table 16: **Capability and compound compression on Llama-3.2-3B.** Left: WikiText-2 perplexity, zero-shot mean, MMLU and GSM8K, with the retained share of the dense model's *above-chance* MMLU (25% chance floor) and of its GSM8K; the zero-shot mean separates the three pruned configurations by 6.6 points, MMLU by 14.5 and GSM8K by more than a factor of two. Right: WikiText-2 perplexity / zero-shot mean after adding INT8 per-channel or INT4 group-128 weights, or LoRA recovery (rank 16, 500 steps on C4). Single checkpoints evaluated end to end, KSP rows at the reduced solver budget of App. C; single seed, compare within this table.

Configuration	WT2 ↓	0-shot ↑	MMLU ↑	GSM8K ↑	retained	+ W8	+ W4	+ LoRA
Dense (reference)	7.81	57.42	56.68	24.80	—	—	—	—
SparseGPT, uniform 2:4	21.11	47.07	29.97	2.40	16% / 10%	21.20 / 47.26	24.15 / 46.06	13.22 / 50.12
KSP, uniform 2:4	19.79	48.29	32.21	2.20	23% / 9%	19.81 / 48.24	22.70 / 46.65	12.87 / 51.33
KSP @62.5%, allocated	**11.91**	**53.69**	**44.50**	**5.20**	**62% / 21%**	**11.93 / 53.78**	**13.51 / 52.22**	**10.64 / 54.84**

E.2.7 Cost and solver diagnostics

Fig. 9 gives pruning wall-clock and peak memory: at the reduced budget KSP stays in the one-shot class, minutes at 3B and 7B and under two hours at 70B.

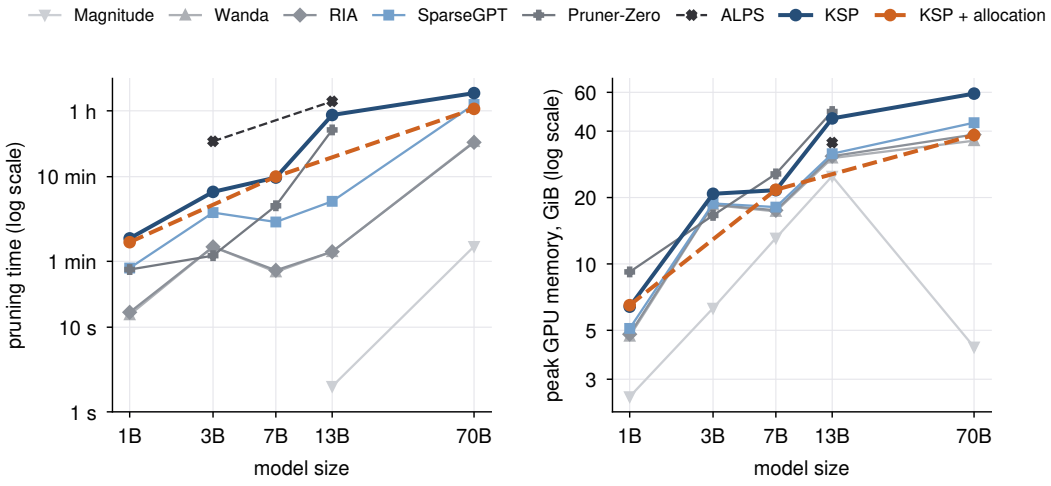


Figure 9: **Pruning cost** — wall-clock (left) and peak GPU memory (right) on one GPU, both on log scales, at the same calibration size as the quality numbers of Tab. 1, so cost scales with that size and is not comparable across model sizes; the 7B and 13B points are Llama-2-7B and Llama-2-13B, the 13B points were timed on an otherwise idle device, untimed cells are omitted, and Magnitude's sub-second times at 1B–7B fall below the axis. KSP stays in the one-shot compute class — minutes at 3B and 7B, under two hours at 70B — with no backpropagation through the model anywhere, and pruning is a one-off offline cost paid once per deployed model. The allocation is computed on a smaller model and transferred, adding no 70B search cost; the allocated 70B point is the faster of the two because a transferred allocation leaves several projections dense. KSP points are at the reduced budget of App. C, except at 13B, which is the reported configuration of Tab. 1.

Independent numerical checks of the closed-form norm, the prox, the certificate and the allocator's optimality conditions ship with the code.