CONSTRAINED REFINEMENT OF LAYERWISE SPARSITY CONTROL FOR LLM PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training pruning of large language models (LLMs) requires deciding which weights to remove within each layer and how to distribute a fixed sparsity budget across layers. At high sparsity, different layerwise allocations under the same global budget can yield substantially different perplexity (PPL) after pruning. Many existing methods derive layerwise sparsity ratios from importance or sensitivity scores or from predefined allocation rules, providing useful starting points. However, these initial allocations may not account for the layerwise reconstruction response of the selected pruning backend at the assigned sparsity levels. This motivates Plug-and-Prune, a constrained method that refines a uniform allocation or one produced by an existing allocator. At the input allocation, it measures layerwise reconstruction responses under the selected backend and uses them in a regularized update of the keep ratios. The update penalizes deviations from the input allocation and enforces the global budget and per-layer bounds, while leaving within-layer mask construction to the backend. We evaluate Plug-and-Prune with WANDA and SparseGPT on LLaMA, Qwen3, and Phi-4 models, primarily at 70% unstructured sparsity, supplemented by sparsity-sweep and structured-pruning experiments. The results show that our method generally lowers PPL relative to the input allocation, with gains tending to be larger at higher sparsity. Overall, response-guided budget redistribution is a practical, backend-agnostic complement to existing inner-layer criteria, especially at high sparsity.

1 INTRODUCTION

Post-training pruning reduces inference and storage costs by removing weights from large language models (LLMs) with limited performance loss (Frantar & Alistarh, 2023; Sun et al., 2024; Zhou et al., 2024). Pruning involves two coupled decisions: which weights to remove within each layer and how much sparsity to assign to each layer. We study the latter, *layerwise sparsity-ratio control*: distributing the pruning ratio across layers[1]. A global sparsity target couples these ratios through a global budget (Sun et al., 2024; Frantar & Alistarh, 2023). This work focuses on high-sparsity settings, primarily above 50% sparsity, where the global budget is tight and layerwise allocation has a pronounced effect: different allocations under the same budget can yield substantially different perplexity. This effect arises because transformer layers differ in their sensitivity to parameter removal, so a uniform allocation may over-prune sensitive layers while leaving layers that tolerate greater sparsity under-pruned; the resulting imbalance is amplified when the budget is tight. Advances in *inner-layer* pruning criteria (Shao et al., 2026; Liu et al., 2025) therefore leave a separate question of how to allocate sparsity across layers for the chosen pruning backend.

A common class of layerwise sparsity methods derives allocations from model statistics or structural priors, including magnitude, layer topology, outlier measures, relative importance, and spectral properties (Zhu & Gupta, 2018; Lee et al., 2021; Zafrir & Shen, 2021; Li et al., 2024b). Methods that induce or prescribe layerwise ratios include global thresholding (Frankle & Carbin, 2019), Erdős–Rényi-style density rules (Mocanu et al., 2018; Liu et al., 2022), and layer-adaptive magnitude rescaling (Lee et al., 2021). LLM-specific methods such as OWL (Yin et al., 2024), DLP (Chen et al., 2025), and AlphaPruning (Lu et al., 2024) provide further nonuniform allocations.

[1]We use sparsity-ratio control for the decision problem and sparsity allocation for a concrete layerwise ratio vector. For convenience, we use the equivalent keep ratio, the complement of the pruning ratio below.

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.

These methods provide useful nonuniform starting allocations through method-specific mappings from statistics to layerwise ratios. At high sparsity, however, allocations derived from these statistics may not account for the layerwise reconstruction response measured under the selected pruning backend at the assigned sparsity levels. Local importance signals (Sun et al., 2024) and reconstruction-based pruning (Frantar & Alistarh, 2023) are already established. Beyond fixed score-to-ratio mappings, prior work has explored blockwise reconstruction-based joint optimization of masks and sparsity in BESA and search-based discovery of layerwise allocation rules in DSA (Xu et al., 2024; Li et al., 2024a). This leaves a distinct allocation-level question: given a supplied allocation and the selected backend's response measured at that allocation, how should the layerwise ratios be refined under the same global budget and per-layer bounds without changing the backend's within-layer pruning rule?

We address this question with Plug-and-Prune. Given a uniform, user-specified, or existing allocation, it optimizes a response-guided objective regularized toward the input allocation under the global budget and per-layer bounds. The refinement is not tied to a particular pruning backend: any backend that accepts layerwise sparsity targets and permits measurement of its layerwise response can be used without modifying its within-layer pruning procedure. Our main contributions are:

- We formulate refinement of a supplied layerwise allocation as constrained optimization over a global budget and per-layer bounds. The formulation separates layerwise reallocation from the selected backend's within-layer pruning procedure and yields a feasible allocation.
- We develop a response-guided refinement for any pruning backend that accepts layerwise sparsity targets and measurable layerwise responses. It optimizes an objective regularized toward the input allocation without altering within-layer pruning. We further derive a one-dimensional multiplier search and characterize the resulting redistribution locally.
- We evaluate Plug-and-Prune with WANDA and SparseGPT on LLaMA, Qwen3, and Phi-4 models with different parameters (Touvron et al., 2023a;b; Yang et al., 2025; Abdin et al., 2024), primarily at 70% unstructured sparsity, supplemented by sparsity-sweep and mixed N:8 experiments. The results show that our method generally lowers PPL relative to the input allocation.

2 Related Work and Problem Formulation

Pruning as layerwise sparsity control. We formulate layerwise pruning as a sparsity-control problem with a given inner-layer pruning backend; the backend itself is not restricted because our focus is the layerwise control decision. For an L-layer model with layer parameters $\{W_l\}_{l=1}^{L}$, let $s_l \in [0,1]$ denote the pruning ratio assigned to layer l, and let $d_l = 1 - s_l$ be the corresponding keep ratio. Write $d = (d_1, \ldots, d_L)$ for the layerwise keep-ratio allocation. If the target global sparsity is ρ, then the average keep ratio is $\bar{d} = 1 - \rho$. Let $\underline{d}_l$ and $\overline{d}_l$ denote lower and upper keep-ratio bounds for layer l, with $0 \le \underline{d}_l < \overline{d}_l \le 1$. The feasible set is

$$\mathcal{C} = \left\{ d \in \mathbb{R}^L \;\middle|\; \frac{1}{L}\sum_{l=1}^{L} d_l = \bar{d},\ \underline{d}_l \le d_l \le \overline{d}_l,\ l = 1, \ldots, L \right\}. \tag{1}$$

For a backend B, let $\mathcal{P}_B(W_l; d_l)$ denote the inner-layer pruning backend that applies backend-specific mask construction and possible weight correction to layer l under keep ratio d_l. Let $F_l(\cdot; W_l)$ denote the l-th transformer block, and let h_0 be the input representation. The pruned model induced by an allocation d is defined recursively as

$$\widehat{h}_0 = h_0, \quad \widehat{W}_l^B(d_l) = \mathcal{P}_B(W_l; d_l), \quad \widehat{h}_l = F_l\Big(\widehat{h}_{l-1}; \widehat{W}_l^B(d_l)\Big), \quad l = 1, \ldots, L. \tag{2}$$

Let $\mathcal{L}_{\text{prune}}$ denote the pruning loss, such as downstream task loss or calibration reconstruction loss relative to the dense model. The layerwise sparsity-control problem is then

$$\min_{d \in \mathcal{C}} \ \mathcal{L}_{\text{prune}}\Big(\widehat{h}_L(d)\Big) \quad \text{s.t. Eq. (2)}, \tag{3}$$

[1]Given a layerwise sparsity allocation, inner-layer pruning backends can further account for weight errors caused by pruning like Optimal Brain Damage and SparseGPT (LeCun et al., 1989; Frantar & Alistarh, 2023). Appendix A details representatives; broader surveys are in (Wan et al., 2024; Zhou et al., 2024).

This formulation separates the layerwise allocation decision from the selected backend's within-layer pruning procedure. Direct optimization is costly because evaluating a candidate allocation requires pruning and assessing the resulting model, and a layer's local pruning response does not by itself determine the loss of the composed model. We therefore use responses measured at a supplied allocation to construct a tractable, constrained refinement objective.

Adaptive layerwise sparsity control in existing work. A common approach constructs layerwise allocations from scores or structural statistics, including magnitude, topology, outlier measures, relative importance, and spectral properties. These methods provide useful nonuniform allocations under a target global budget. In our formulation, meeting that budget alone does not ensure that an allocation also satisfies the per-layer bounds in Eq. (1). For the score-based methods considered here, the mapping from scores to ratios is typically specified without measuring the selected backend's response at the ratios produced by that mapping. This leaves room for refinement once the backend's response is observed. LSA uses minimal linear reconstruction error at a reference sparsity for allocation (Yang et al., 2026), BESA uses blockwise reconstruction loss for joint mask-and-sparsity optimization (Xu et al., 2024), and DSA searches for allocation rules (Li et al., 2024a). Plug-and-Prune addresses a distinct refinement setting: given a supplied allocation and a selected backend, it measures the backend's layerwise response at that allocation and uses it in a regularized update subject to the global budget and per-layer bounds, leaving within-layer pruning to the backend.

3 Plug-and-Prune for Cross-Layer Sparsity Control

As shown in Fig. 1, building on Eq. (3), we replace the computationally expensive full pruning objective with a tractable allocation-level surrogate. Let B be the fixed inner-layer pruning backend and let $d^0 \in \mathcal{D}_{\text{in}}$ be an input allocation, where $\mathcal{D}_{\text{in}}$ includes uniform, user-specified, and existing-rule allocations. Plug-and-Prune uses the following plug-and-prune step:

$$\begin{aligned} \textbf{Plug:} \quad & d^0 \in \mathcal{D}_{\text{in}}, \\ \textbf{Surrogate for Eq. (3):} \quad & R(d; d^0, B) = \overbrace{\mathcal{R}_B(d; d^0)}^{\text{response surrogate}} + \overbrace{\lambda \Omega(d, d^0)}^{\text{regularization}}, \\ \textbf{Sparsity Control:} \quad & d^\star \in \arg\min_{d \in \mathcal{C}} R(d; d^0, B), \\ \textbf{Apply backend:} \quad & \widehat{h}_L(d^\star) \text{ is generated by Eq. (2).} \end{aligned} \tag{4}$$

The term $\mathcal{R}_B$ denotes a backend-aware response surrogate, and Ω regularizes the refinement and penalizes deviations from the input allocation. The same objective form is used across input allocations; only scalar refinement strengths are tuned. Thus Eq. (4) is a constrained surrogate for Eq. (3): it preserves the same feasible set and backend recursion, but avoids repeated full-model pruning and evaluation. As a basic well-posedness check, we first show that this refinement problem has a minimizer under standard compactness and lower-semicontinuity conditions.

Proposition 1 (Existence of a refinement minimizer)**.** *For a fixed input allocation d^0 and backend B, assume that $\mathcal{C}$ is nonempty and compact, and that $R(\cdot; d^0, B)$ is lower semicontinuous on $\mathcal{C}$. Then the refinement problem in Eq.* (4) *has at least one minimizer $d^\star \in \mathcal{C}$.*

Proposition 1 is a well-posedness check: it verifies that the constrained refinement is defined once d^0, B, and R are fixed; see Appendix B. We next instantiate these components into Plug-and-Prune, which differs from existing rule-driven schemes by optimizing an input allocation under constraints.

3.1 Input Allocation and Backend Response

The input allocation d^0 initializes the refinement problem in Eq. (4). It can be uniform, user-specified, or generated by an existing allocator. Regardless of its source, d^0 is an allocation to be refined rather than the final pruning decision. A more informative d^0 may reduce the required correction, but the framework also applies to less informative inputs such as uniform allocation; Section 4 compares refinement from these different inputs.

We next construct the backend response used by the response surrogate. Let X_l denote the calibration input activations to layer l, and write $F_l(X; W)$ for the layer map with input X and weight W. The input allocation fixes the probing sparsity ratio $s_l^0 = 1 - d_l^0$. Let ψ_B be the backend-specific

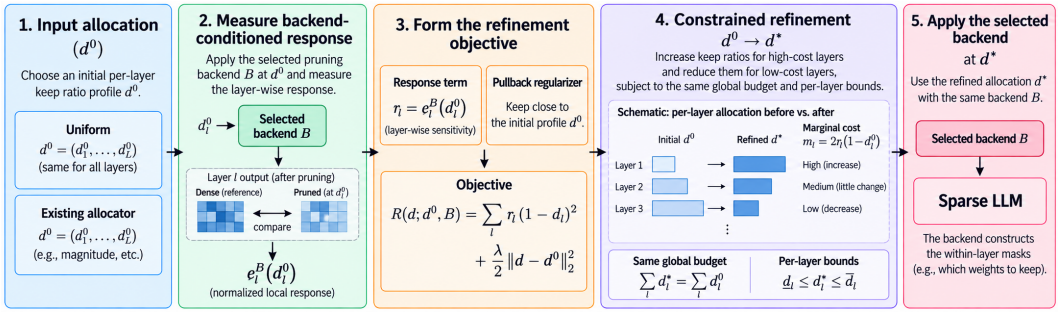

Figure 1: Plug-and-Prune workflow. An input allocation is refined using the selected backend's measured layerwise response, a pullback regularizer, and the global budget and per-layer bounds. The diagram shows the local-only response coefficient $r_l = e_l^B(d_l^0)$; an optional downstream modifier is described in the text.

scoring rule that computes S_l^B, $\mathcal{M}_B$ the mask constructor, and $\mathcal{T}_B$ the pruning operator, including possible weight correction. At this fixed ratio, backend B produces the pruned layer as

$$S_l^B = \psi_B(W_l, X_l), \qquad M_l^B(s_l^0) = \mathcal{M}_B(S_l^B, s_l^0), \qquad \widehat{W}_l^B(d_l^0) = \mathcal{T}_B(W_l, M_l^B(s_l^0), X_l). \quad (5)$$

This abstraction covers both mask-only and reconstruction-compensated pruning rules (Sun et al., 2024; Frantar & Alistarh, 2023). With the operator fixed at s_l^0, we compare dense and backend-pruned layer outputs on the same input and regard it as the *layer response*:

$$Y_l = F_l(X_l; W_l), \qquad \widehat{Y}_l^B = F_l(X_l; \widehat{W}_l^B(d_l^0)), \qquad \Delta Y_l^B = Y_l - \widehat{Y}_l^B. \quad (6)$$

To compare responses across layers with different activation scales, we use the relative layer-output reconstruction error:

$$e_l^B(d_l^0) = \frac{1}{q_l} \sum_{h=1}^{q_l} \frac{\|\Delta Y_{l,h}^B\|_2^2}{\|Y_{l,h}\|_2^2 + \varepsilon}, \quad (7)$$

where q_l is the number of feature dimensions and $\varepsilon > 0$ is a small constant for numerical stability. The index h denotes feature dimensions, and the norm is taken over calibration samples and token positions. Thus $e_l^B(d_l^0)$ is measured at the input allocation: the probing ratio and backend are fixed, and only the dense–pruned discrepancy is recorded. The denominator scales each feature's reconstruction error by its dense-output energy, reducing the influence of activation magnitude when comparing responses across layers. We use $e_l^B(d_l^0)$ as a local, backend-conditioned response signal, rather than an estimate of full-model pruning loss. The following statements are therefore proxy-level design checks, rather than claims about exact transformer dynamics.

Proposition 2 (Scale stability of the relative-response proxy)**.** *Suppose that under $X_l \mapsto \alpha X_l$ with $\alpha > 0$, the layer output and backend-induced discrepancy scale locally by α. Define $U_l = \frac{1}{m_l} \sum_{h=1}^{m_l} \|\Delta Y_{l,h}^B\|_2^2 / \|Y_{l,h}\|_2^2$. If $\|Y_{l,h}\|_2 > 0$ for all averaged feature dimensions and $U_l \le C_l$ for a finite constant C_l, then*

$$0 \le \rho_l(\alpha) = \frac{1}{m_l} \sum_{h=1}^{m_l} \frac{\alpha^2 \|\Delta Y_{l,h}^B\|_2^2}{\alpha^2 \|Y_{l,h}\|_2^2 + \varepsilon} \le U_l \le C_l.$$

Proposition 2 only indicates that the Eq. (7) proxy is scale-stable under a simplified scaling approximation; it remains a local conditional response rather than a loss estimate.

Because the local response does not describe how the resulting perturbation evolves through later blocks, we also consider an optional downstream modifier. The starting point is the usual first-order propagation picture, which is consistent with prior signal-propagation and dynamical-isometry analyses of deep networks (Poole et al., 2016; Schoenholz et al., 2017; Pennington et al., 2017). Let δh_l denote the output perturbation introduced at layer l, and let J_j be the local Jacobian of block j along the calibration trajectory. Then

$$\delta h_L \approx J_{L:l+1} \delta h_l, \qquad J_{L:l+1} = J_L J_{L-1} \cdots J_{l+1}. \quad (8)$$

Eq. (8) is used as a guide: the effect of pruning layer l depends not only on the local discrepancy but also on the Jacobian gains of later blocks. Because estimating the full product $J_{L:l+1}$ is costly for LLMs, we use a calibration-observable activation ratio as a low-cost heuristic for the optional propagation correction. Let x_j and y_j denote the input and output activations of block j, and let $\mathbb{E}_X$ denote averaging over calibration samples. We define the gain g_j as the output-to-input RMS ratio,

$$g_j = \mathbb{E}_X \left[\frac{\mathrm{rms}(y_j)}{\mathrm{rms}(x_j) + \varepsilon} \right]. \tag{9}$$

The notation $\mathrm{rms}(\cdot)$ denotes the root-mean-square activation magnitude. The statistic g_j is used as an observable proxy for the average perturbation amplification induced by block j. This use of g_j is a calibration-based transfer proxy rather than a verified Jacobian estimate. The local explanation in Appendix C.2 follows the standard signal-propagation simplification that averages perturbations over random directions, often modeled with isotropic covariance (Poole et al., 2016; Schoenholz et al., 2017; Pennington et al., 2017). Let $\Gamma : \mathbb{R}_+ \to \mathbb{R}_+$ be a nonnegative block-level transfer map. We can have:

$$a_l = \begin{cases} 1, & \text{local-only response,} \\ \prod_{j>l} \Gamma(g_j), & \text{propagation-adjusted response.} \end{cases} \tag{10}$$

The suffix product reflects network composition: a perturbation introduced at layer l passes through downstream blocks before reaching the output. The choice of Γ converts calibration statistics into response coefficients; our implementation uses a damped gain map to account for residual identity paths and improve numerical stability; see Appendix C.3. Both variants use the same response coefficient in the constrained allocation objective:

$$r_l = \underbrace{e_l^B(d_l^0)}_{\text{local response}} \cdot \underbrace{a_l}_{\text{optional transfer factor}}. \tag{11}$$

Setting $a_l = 1$ gives the local-only response, while the suffix-product choice adds the propagation correction. Thus r_l should be interpreted as a surrogate coefficient rather than a full-model loss estimate. Appendix C.4 evaluates the incremental effect of the propagation modifier.

3.2 Low-Complexity L2-Regularized Sparsity Optimization

The previous subsection provides the two components needed for refinement: an input allocation d^0 and nonnegative local response coefficients $\{r_l\}_{l=1}^L$. After response collection, we treat r_l as fixed coefficients in the refinement problem. Given a pullback weight $\lambda > 0$, we compute the final feasible sparsity-control decision by solving

$$d^\star = \arg\min_{d \in \mathcal{C}} \sum_{l=1}^{L} r_l (d_l - 1)^2 + \frac{\lambda}{2} \|d - d^0\|_2^2. \tag{12}$$

The constrained update has a marginal-benefit interpretation. For a layer whose keep ratio does not reach its bounds, the KKT condition for Eq. (12) is

$$2r_l(1 - d_l^\star) - \lambda(d_l^\star - d_l^0) = \nu, \tag{13}$$

where ν is the multiplier associated with the global keep-ratio budget. At the input allocation, the regularization derivative is zero, giving the initial marginal surrogate benefit $m_l = 2r_l(1 - d_l^0)$. Thus, for an interior layer,

$$d_l^\star - d_l^0 = \frac{m_l - \nu}{2r_l + \lambda}. \tag{14}$$

Layers with $m_l > \nu$ receive a larger keep ratio, while those with $m_l < \nu$ absorb more pruning; layerwise bounds can limit these changes. This balance characterizes the fixed-response surrogate, not the optimum of end-to-end perplexity.

Theorem 1 (One-dimensional solution map)**.** *Consider Eq.* (12)*. For an interior solution, the Karush–Kuhn–Tucker (KKT) conditions reduce the problem to one scalar budget multiplier and independent per-layer quadratic updates. If this closed-form candidate satisfies the layerwise bounds, it is returned directly. Otherwise, with active box or band constraints, bisection over the same multiplier with per-layer clipping gives the constrained optimizer in $\mathcal{C}$.*

Theorem 1 records the computational form of Eq. (12): it first has an interior closed-form candidate, and falls back to one-dimensional multiplier search when layerwise bounds become active. The derivation is given in Appendix D, and Algorithm 1 summarizes the procedure.

Algorithm 1 Constrained sparsity-control refinement

Require: input allocation d^0, response coefficients r, feasible set $\mathcal{C}$, $\lambda > 0$, tolerance tol
Ensure: $d^\star \in \mathcal{C}$
  Set $A_l \leftarrow 2r_l + \lambda d_l^0$ and $D_l \leftarrow 2r_l + \lambda$ for all layers
  Compute $\nu_0 \leftarrow \big(\sum_l A_l/D_l - L\bar{d}\big)/\sum_l 1/D_l$
  Set the interior candidate $\tilde{d}_l \leftarrow (A_l - \nu_0)/D_l$ for all layers
  if $\tilde{d} \in \mathcal{C}$ **then**
    return $d^\star \leftarrow \tilde{d}$
  end if
  Find ν by scalar bisection on $d_l(\nu) = \Pi_{[\underline{d}_l, \bar{d}_l]}((A_l - \nu)/D_l)$ so that $\sum_l d_l(\nu) = L\bar{d}$ *no backend call*
  return $d^\star \leftarrow d(\nu)$

Algorithm 1 adds limited computation beyond response collection. When the interior candidate satisfies the layerwise bounds, the algorithm returns the closed-form allocation directly. If some bounds are active, the bisection procedure is a one-dimensional numerical bisection over ν: it does not call the pruning backend and requires no additional forward or backward passes. Each step evaluates the clipped closed-form map over L layers and costs $O(L)$. The same map naturally extends to multiple rounds by feeding one round's refined allocation back as the next input allocation; Appendix G gives a small probe showing additional gains when extra response collection is acceptable. The hyperparameters, such as the pullback weight, control refinement strength within a fixed constrained objective rather than defining a new hand-designed allocation rule. Detailed comparisons with allocation-rule search are given in Appendix H. The pullback regularizer discourages large deviations from the input allocation. Optionally, a trust region $\|d - d^0\|_\infty \le \tau$ can further limit extrapolation of the response measured at d^0; we treat it as a safeguard rather than part of the main configuration.

We next analyze the refinement step when the response surrogate is locally informative. This is a conditional local result, not an unconditional performance guarantee. It assumes that, near the input allocation d^0, the surrogate response is aligned with the degradation up to a local quadratic error.

Theorem 2 (Conditional local refinement behavior)**.** *Let $d^\star$ solve Eq. (12), and define $\Phi(d) = \sum_l r_l(d_l - 1)^2$, $F(d) = \Phi(d) + \frac{\lambda}{2}\|d - d^0\|_2^2$, and $\Delta(d) = \mathcal{L}(\widehat{f}_d) - \mathcal{L}(f)$, where $\mathcal{L}$ is the evaluation loss of the pruned model $\widehat{f}_d$ relative to the dense model f. Assume $d^0 \in \mathcal{C}$, and let $\Phi_{\mathcal{C}}^\star = \min_{d \in \mathcal{C}} \Phi(d)$. For (i), additionally assume that there exist $c > 0$ and $\eta < c\lambda/2$ such that the local surrogate–degradation alignment condition*

$$\Delta(d) - \Delta(d^0) \le c\big(\Phi(d) - \Phi(d^0)\big) + \eta\|d - d^0\|_2^2$$

holds at $d = d^\star$. Then the refinement satisfies:

(i) Local non-degradation under alignment: $\Delta(d^\star) - \Delta(d^0) \le -\Big(\frac{c\lambda}{2} - \eta\Big)\|d^\star - d^0\|_2^2$,

(ii) Small updates near surrogate-optimal inputs: $\|d^\star - d^0\|_2^2 \le \frac{2}{\lambda}\big(\Phi(d^0) - \Phi_{\mathcal{C}}^\star\big)$,

(iii) Feasible-direction improvement: $\Phi(d^\star) \le \Phi(d^0) - \gamma$, *if* $\exists\, \gamma > 0,\ \tilde{d} \in \mathcal{C} :\ F(\tilde{d}) \le \Phi(d^0) - \gamma$.

The proof is in Appendix I. The theorem should be interpreted under the stated local surrogate-alignment condition. It does not assert that the response proxy is globally accurate; instead, it explains why the update can be stable for well-aligned allocations and can move less informative allocations toward lower-surrogate feasible directions. Accordingly, Section 4 reports results for both uniform-input refinements and refinements initialized from existing allocators. The main tables focus on LLaMA-1/2 for matched comparisons across input allocations; results on LLaMA-3-8B, Qwen3-0.6B/4B, and Phi-4 are reported in Appendix K.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models, backends, and calibration. We primarily evaluate post-training pruning on selected LLaMA-1 and LLaMA-2 checkpoints commonly used in layerwise pruning studies (Touvron et al., 2023a;b). To examine the method on additional models, we evaluate LLaMA-3-8B, Qwen3-0.6B/4B, and Phi-4(Grattafiori et al., 2024; Yang et al., 2025; Abdin et al., 2024). A separate LLaMA-3-70B scale check is reported in Appendix J. We use WANDA (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023) because they are widely used inner-layer backends and standard baselines for layerwise allocation studies. Our primary unstructured evaluation targets 70% sparsity, where layerwise budget placement is more consequential; a LLaMA-2-7B sweep over 50–85% in Appendix E examines how the benefit varies with sparsity. Responses are collected with the backend used for final pruning.

Baselines and paired evaluation. We compare against uniform allocation and three representative adaptive allocators: OWL, DLP, and AlphaPruning. Within each pair, the model, pruning backend, calibration set, and global sparsity target are fixed; the only intervention is the single-round constrained refinement of the input allocation.

Configuration and metrics. Main experiments use the propagation-adjusted coefficient; other variants are evaluated in Appendix C.4. We run the WikiText-2 perplexity experiments and zero-shot accuracy on BoolQ (Clark et al., 2019), RTE (Wang et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), OpenBookQA (Mihaylov et al., 2018), ARC-Easy, and ARC-Challenge (Clark et al., 2018). on a machine equipped with a single NVIDIA RTX PRO 6000 GPU (96 GB of memory), 25 Intel Xeon Platinum 8470Q vCPUs, and 120 GB of system memory.

Table 1: WikiText-2 perplexity at 70% unstructured sparsity. Each row plugs the listed allocation into the constrained optimizer and reports default → optimized; lower is better.

Inner-layer backend	Plugged allocation	WikiText-2 perplexity: default → optimized (Δ)				
		LLaMA-1 7B	LLaMA-2 7B	LLaMA-1 13B	LLaMA-2 13B	LLaMA-1 30B
WANDA	Uniform	88.86 → 23.34 (-65.52)	76.45 → 26.45 (-50.00)	54.54 → 13.87 (-40.67)	34.13 → 14.74 (-19.39)	17.58 → 11.24 (-6.34)
	OWL	27.94 → 21.42 (-6.52)	35.94 → 24.41 (-11.53)	19.57 → 13.46 (-6.11)	18.00 → 15.66 (-2.34)	11.92 → 9.96 (-1.96)
	DLP	21.18 → 20.63 (-0.55)	24.36 → 22.14 (-2.22)	13.64 → 13.53 (-0.11)	16.19 → 15.03 (-1.16)	9.85 → 9.81 (-0.04)
	AlphaPruning	47.15 → 29.23 (-17.92)	30.10 → 26.57 (-3.53)	14.79 → 13.47 (-1.32)	15.40 → 13.89 (-1.51)	9.71 → 9.46 (-0.25)
SparseGPT	Uniform	26.33 → 20.85 (-5.48)	26.68 → 20.99 (-5.69)	18.97 → 14.25 (-4.72)	20.60 → 13.10 (-7.50)	12.75 → 11.52 (-1.23)
	OWL	20.33 → 18.99 (-1.34)	21.85 → 19.37 (-2.48)	14.28 → 13.02 (-1.26)	16.29 → 14.06 (-2.23)	10.70 → 9.86 (-0.84)
	DLP	18.13 → 18.12 (-0.01)	18.75 → 18.40 (-0.35)	12.89 → 12.48 (-0.41)	13.54 → 13.00 (-0.54)	9.51 → 9.42 (-0.09)
	AlphaPruning	22.09 → 19.76 (-2.33)	21.14 → 19.18 (-1.96)	13.33 → 12.79 (-0.54)	13.81 → 13.05 (-0.76)	9.72 → 9.44 (-0.28)

Table 2: Zero-shot results at 70% unstructured sparsity. Each row plugs the listed allocation into the constrained optimizer and reports average accuracy: default → optimized; higher is better.

Inner-layer backend	Input allocation	Average zero-shot accuracy: default → optimized (Δ)				
		LLaMA-1 7B	LLaMA-2 7B	LLaMA-1 13B	LLaMA-2 13B	LLaMA-1 30B
WANDA	Uniform	0.40 → 0.45 (+0.05)	0.38 → 0.44 (+0.06)	0.41 → 0.49 (+0.08)	0.41 → 0.50 (+0.09)	0.51 → 0.53 (+0.02)
	OWL	0.45 → 0.47 (+0.02)	0.42 → 0.45 (+0.03)	0.47 → 0.49 (+0.02)	0.45 → 0.50 (+0.05)	0.53 → 0.53 (+0.00)
	DLP	0.46 → 0.47 (+0.01)	0.44 → 0.46 (+0.02)	0.49 → 0.50 (+0.01)	0.49 → 0.51 (+0.02)	0.53 → 0.53 (+0.00)
	AlphaPruning	0.46 → 0.47 (+0.01)	0.44 → 0.45 (+0.01)	0.49 → 0.50 (+0.01)	0.49 → 0.50 (+0.01)	0.54 → 0.54 (+0.00)
SparseGPT	Uniform	0.44 → 0.46 (+0.02)	0.44 → 0.46 (+0.02)	0.47 → 0.49 (+0.02)	0.46 → 0.51 (+0.05)	0.53 → 0.54 (+0.01)
	OWL	0.47 → 0.47 (+0.00)	0.46 → 0.47 (+0.01)	0.49 → 0.50 (+0.01)	0.48 → 0.51 (+0.03)	0.54 → 0.54 (+0.00)
	DLP	0.47 → 0.48 (+0.01)	0.48 → 0.48 (+0.00)	0.50 → 0.50 (+0.00)	0.51 → 0.51 (+0.00)	0.54 → 0.54 (+0.00)
	AlphaPruning	0.48 → 0.48 (+0.00)	0.47 → 0.48 (+0.01)	0.49 → 0.50 (+0.01)	0.50 → 0.51 (+0.01)	0.54 → 0.54 (+0.00)

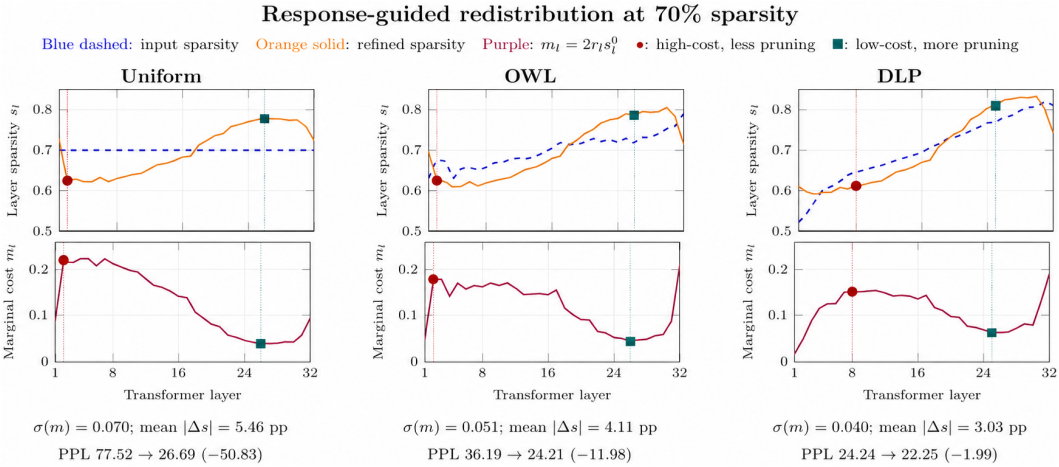

Figure 2: Response-guided sparsity redistribution in a separate local-only LLaMA-2-7B/WANDA diagnostic at 70% target sparsity ($a_l = 1$). Upper panels show input and refined layerwise sparsities, s_l^0 and $s_l^\star$; lower panels show the initial marginal surrogate cost $m_l = 2r_l s_l^0$. For this diagnostic, r_l is the measured local response $e_l^B(d_l^0)$. $\sigma(m)$ is the across-layer standard deviation of m_l, and mean $|\Delta s|$ is the mean absolute change in layerwise sparsity, in percentage points.

4.2 Main Results

Language modeling. Tables 1 and 2 report paired WikiText-2 perplexity (PPL) and seven-task average zero-shot accuracy for LLaMA-family models at 70% unstructured sparsity. Refinement lowers PPL in all 40 displayed pairs, with larger reductions generally occurring from uniform and other higher-PPL inputs, and smaller gains from strong inputs such as DLP. The seven-task average increases in 29 pairs, including all ten uniform-input pairs, and is unchanged at the table's two-decimal precision in the remaining 11. PPL reductions therefore often accompany gains in average zero-shot accuracy, but a large PPL reduction need not produce a comparably large accuracy gain or improve every task. WikiText-2 PPL measures token-level likelihood, whereas zero-shot accuracy depends on discrete, task-specific answer choices.

To complement these performance results with an allocation-level view, Figure 2 shows a separate local-only diagnostic run on LLaMA-2-7B with WANDA at 70% sparsity. The lower panels display the initial marginal surrogate cost $m_l = 2r_l s_l^0$, which combines the measured local response with each layer's input sparsity; the upper panels compare the input and refined allocations. Higher-cost layers are generally pruned less, while lower-cost layers absorb more pruning under the same global budget. Compared with Uniform, DLP has a narrower spread of m_l, smaller allocation changes, and a smaller additional PPL reduction. The figure illustrates how the response-guided surrogate directs redistribution; it does not establish that the displayed cost profiles cause the observed PPL differences. Appendix E extends the LLaMA-2-7B evaluation to 50–85% sparsity using WANDA and SparseGPT with Uniform, OWL, and DLP inputs. Gains are small at 50% and can be larger from 60% onward, particularly from Uniform. With permuted and constant controls under the same constrained solver, measured responses achieve the lowest PPL in all six 70% settings, whereas the 50% differences are small and mixed (Appendix F). To complement the LLaMA-1/2 downstream results, we evaluate OWL-initialized refinement on LLaMA-3-70B at 70% unstructured sparsity. The seven-task average increases from 46.22% to 51.30% with WANDA and from 45.62% to 46.29% with SparseGPT. These results show average zero-shot gains in this large-model setting, although the gain varies by backend and does not imply improvement on every task.

Table 3 compares the proposed plug-and-prune (PnP) refinement starting from a uniform allocation against DSA's released-rule transfer and a target-matching layer-wise BESA configuration. Here, PnP values are the WANDA/Uniform entries from the paired main tables. DSA applies its released allocation rule, while BESA uses a target-matching layer-wise configuration rather than untouched default settings. At the requested 70% sparsity level, PnP achieves lower PPL on all five LLaMA models. Its seven-task average exceeds BESA's in all five cases and achieve better or equal to DSA's.

Table 3: Performance comparison with mask-and-sparsity optimization (BESA) and allocation-rule search (DSA) on LLaMA-family models at a target of 70% unstructured sparsity. Lower WikiText-2 perplexity (PPL) and higher seven-task average are better.

Metric	Method	LLaMA-1 7B	LLaMA-2 7B	LLaMA-1 13B	LLaMA-2 13B	LLaMA-1 30B
WikiText-2 PPL ↓	PnP (WANDA, Uniform)	23.34	26.45	13.87	14.74	11.24
	DSA	24.63	64.78	16.31	36.29	15.08
	BESA	168.58	78.80	53.65	48.48	17.33
Seven-task average ↑	PnP (WANDA, Uniform)	0.45	0.44	0.49	0.50	0.53
	DSA	0.45	0.39	0.47	0.44	0.52
	BESA	0.36	0.37	0.39	0.38	0.45
Actual sparsity (%)	PnP (WANDA, Uniform)	69.99	69.99	69.99	69.99	69.99
	DSA	69.99	69.99	69.99	69.99	69.99
	BESA	70.98	70.27	69.96	70.24	70.23

Table 4: WikiText-2 perplexity on LLaMA-2-7B under mixed N:8 structured pruning. Each entry reports fixed-N baseline → Plug-and-Prune at the same global sparsity; lower is better.

Backend	4:8 (50%)	3:8 (62.5%)	2:8 (75%)
WANDA	$8.334 \rightarrow 7.957$	$38.328 \rightarrow 27.824$	$1753.328 \rightarrow 180.403$
SparseGPT	$8.572 \rightarrow 8.286$	$19.564 \rightarrow 16.497$	$89.021 \rightarrow 74.266$

Because BESA uses a different mask-optimization pipeline and attains sparsity levels ranging from 69.96% to 70.98%, this is a method-level comparison rather than an allocation-only ablation.

Mixed N:8 structured pruning. The preceding experiments focused on unstructured pruning. We also examine whether response-guided budget redistribution helps under structured mask constraints. In fixed-N pruning, every layer retains the same number of weights per group of eight. Mixed N:8, by contrast, allows different integer retained counts across layers while preserving the same global budget as fixed-N. Both response measurement and final mask construction use the selected backend, and the refined allocations preserve valid N:8 masks and the global budget. As shown in Table 4, the proposed PnP reduces PPL relative to the fixed-N baseline at 4:8, 3:8, and 2:8 under both WANDA and SparseGPT for the reported configurations.

Pruning runtime. For the OWL, DLP, and AlphaPruning inputs reported in Tables A5 and A6, the runtime overhead is about 950% with WANDA and 4467% with SparseGPT, where the reconstruction-compensated backend makes response collection more expensive. On LLaMA-1-7B under these input allocations, pruning plus refinement takes 111.8153.1 seconds with WANDA and 438.1482.2 seconds with SparseGPT. Uniform-initialized results and method-level runtime comparisons are reported separately in Table A7

5 Conclusion

We studied layerwise sparsity control as a constrained decision problem in LLM pruning. Plug-and-Prune accepts a uniform or existing layerwise allocation and refines it under an explicit global budget and per-layer constraints. Its backend-aware response objective adapts the allocation to the selected inner-layer pruning method with limited additional computation. Experiments across LLaMA and Qwen3 models with WANDA and SparseGPT show that refinement reduces perplexity most strongly for uniform and weaker input allocations, while gains over already strong inputs are smaller; it also generally improves zero-shot accuracy and, in the LLaMA-2-7B unstructured sparsity sweep, yields limited additional benefit at 50% sparsity but larger gains at higher sparsity.

A limitation is that the response surrogate is calibrated at the input allocation and is not an exact estimator of the full pruning loss. Future work may develop more accurate response signals, identify better-quality allocation and hyper-parameters for control, and explore multi-stage refinements that iteratively update both the surrogate and the allocation to further improve pruning performance.

ETHICS STATEMENT

Our experiments prune existing pretrained language models using established calibration and evaluation datasets; they do not involve new human-subject data collection.

AI USE STATEMENT

In this work, we used generative AI tools to assist with software implementation and debugging, experiment planning and parameter suggestions, organization and preliminary interpretation of experimental results, and language editing and formatting. Technical claims and citations were checked against the corresponding original sources. No synthetic datasets results are reported. The authors reviewed and approved all AI-assisted material and take full responsibility for the final content of the paper, including its methods, experiments, claims, and conclusions.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 Technical Report, 2024.

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

Yuli Chen, Bo Cheng, Jiale Han, Yingying Zhang, Yingting Li, and Shuhao Zhang. DLP: Dynamic Layerwise Pruning in Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7934–7956. PMLR, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2924–2936. Association for Computational Linguistics, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*, 2019.

Elias Frantar and Dan Alistarh. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. In *International Conference on Machine Learning*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell,

Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi (Jack) Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu,

Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu (Sid) Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, 2024.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, 1989.

Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. Layer-Adaptive Sparsity for Magnitude-Based Pruning. In *International Conference on Learning Representations*, 2021.

Lujun Li, Peijie Dong, Zhenheng Tang, Xiang Liu, Qiang Wang, Wenhan Luo, Wei Xue, Qifeng Liu, Xiaowen Chu, and Yike Guo. Discovering Sparsity Allocation for Layer-Wise Pruning of Large Language Models. In *Advances in Neural Information Processing Systems*, 2024a.

Wei Li, Lujun Li, Mark G. Lee, and Shengjie Sun. Adaptive Layer Sparsity for Large Language Models via Activation Correlation Assessment. In *Advances in Neural Information Processing Systems*, 2024b.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The Unreasonable Effectiveness of Random Pruning: Return of the Most Naive Baseline for Sparse Training. In *International Conference on Learning Representations*, 2022.

Xuyang Liu, Zichen Wen, Shaobo Wang, Junjie Chen, Zhishan Tao, Yubo Wang, Tailai Chen, Xiangqi Jin, Chang Zou, Yiyu Wang, Chenfei Liao, Xu Zheng, Honggang Chen, Weijia Li, Xuming Hu, Conghui He, and Linfeng Zhang. Shifting AI Efficiency from Model-Centric to Data-Centric Compression. *CoRR*, abs/2505.19147, 2025.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W. Mahoney, and Yaoqing Yang. AlphaPruning: Using Heavy-Tailed Self Regularization Theory for Improved Layer-wise Pruning of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391. Association for Computational Linguistics, 2018.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable Training of Artificial Neural Networks with Adaptive Sparse Connectivity Inspired by Network Science. *Nature Communications*, 9(1):2383, 2018.

Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. Resurrecting the Sigmoid in Deep Learning through Dynamical Isometry: Theory and Practice. In *Advances in Neural Information Processing Systems*, 2017.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential Expressivity in Deep Neural Networks through Transient Chaos. In *Advances in Neural Information Processing Systems*, 2016.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *Communications of the ACM*, 64(9):99–106, 2021.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep Information Propagation. In *International Conference on Learning Representations*, 2017.

Kele Shao, Keda Tao, Kejia Zhang, Sicheng Feng, Mu Cai, Yuzhang Shang, Haoxuan You, Can Qin, Yang Sui, and Huan Wang. A Survey of Token Compression for Efficient Multimodal Large Language Models. *Transactions on Machine Learning Research*, 2026.

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. *CoRR*, abs/2302.13971, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR*, abs/2307.09288, 2023b.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. Efficient Large Language Models: A Survey. *Transactions on Machine Learning Research*, 2024.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *Advances in Neural Information Processing Systems*, 2019.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. BESA: Pruning Large Language Models with Blockwise Parameter-Efficient Sparsity Allocation. In *International Conference on Learning Representations*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 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*, volume 2026, pp. 76241–76266, 2026.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Kumar Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier Weighed Layerwise Sparsity (OWL): A Missing Secret Sauce for Pruning LLMs to High Sparsity. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57101–57115. PMLR, 2024.

Ofir Zafrir and Haihao Shen. Prune Once for All: Sparse Pre-Trained Language Models. In *NeurIPS Workshop on Efficient Natural Language and Speech Processing*, 2021.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. A Survey on Efficient Inference for Large Language Models. *CoRR*, abs/2404.14294, 2024.

Michael H. Zhu and Suyog Gupta. To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. In *International Conference on Learning Representations, Workshop Track*, 2018.

A Expanded pruning objective and inner-layer backends

Eq. (2) defines the pruned model through a backend operator $\mathcal{P}_B(W_l; d_l)$, while Eq. (3) optimizes the keep-ratio allocation d over $\mathcal{C}$. Expanding $\mathcal{P}_B$, the nested structure can be written as

$$\min_{d\in\mathcal{C}} \mathcal{L}_{\text{prune}}\Big(\widehat{h}_L(d)\Big) \quad \text{s.t.} \quad \widehat{h}_0 = h_0, \quad \widehat{W}_l^B(d_l) = \mathcal{P}_B(W_l; d_l), \quad \widehat{h}_l = F_l\Big(\widehat{h}_{l-1}; \widehat{W}_l^B(d_l)\Big),$$
$$\widehat{W}_l^B(d_l) = \mathcal{P}_B(W_l; d_l), \qquad \mathcal{P}_B(W_l; d_l) \approx \arg\min_{\widehat{W}:\|\widehat{W}\|_0 \le d_l|W_l|} \mathcal{E}_l^B(\widehat{W}; W_l, X_l),$$

for $l = 1, \ldots, L$. Here, X_l denotes the input calibration activations to layer l, and $\mathcal{E}_l^B$ is the backend-specific inner-layer criterion. The approximation sign reflects that practical backends often implement efficient one-shot, greedy, or blockwise procedures rather than solving the inner sparse approximation problem exactly. The main paper focuses on the outer allocation variable d, treating the backend pruning operator as a modular inner-layer procedure.

With weight-error compensation. Some inner-layer methods also compensate retained weights to reduce reconstruction error, rather than only setting pruned weights to zero. A common layerwise form is

$$\min_{\widehat{W}_l} \|W_l X_l - \widehat{W}_l X_l\|_F^2 \quad \text{s.t.} \quad \|\widehat{W}_l\|_0 \le d_l|W_l|,$$

or, equivalently,

$$\widehat{W}_l^B(d_l) = \big(W_l + \Delta_l^B(d_l)\big) \odot M_l^B(d_l),$$

where $M_l^B(d_l)$ is the backend mask at keep ratio d_l and $\Delta_l^B(d_l)$ modifies retained weights to compensate pruning-induced error. This correction is part of $\mathcal{P}_B$ and $\mathcal{E}_l^B$; it does not change the outer allocation variable d.

WANDA. WANDA (Sun et al., 2024) is a representative one-shot inner-layer pruning rule without explicit weight reconstruction. Given calibration activations, it scores each weight by combining magnitude and input activation scale, typically in the form $|W_{l,ij}|\|X_{l,j}\|_2$. For a target keep ratio d_l (equivalently, sparsity ratio $s_l = 1 - d_l$), WANDA prunes the lowest-scoring weights within the chosen granularity, so

$$\mathcal{P}_{\text{WANDA}}(W_l; d_l) = W_l \odot M_l^{\text{WANDA}}(d_l).$$

SparseGPT. SparseGPT (Frantar & Alistarh, 2023) is a representative reconstruction-based backend. It approximates the layerwise reconstruction objective using a calibration Hessian or activation covariance estimated from X_l, and prunes weights while updating the remaining weights through second-order error compensation. In the notation above, $\mathcal{P}_{\text{SparseGPT}}(W_l; d_l)$ implements this compensated pruning procedure for each assigned keep ratio d_l.

B Proof of Proposition 1

The feasible set $\mathcal{C}$ is nonempty by assumption. Under the box constraints and the finite-dimensional budget constraint in Eq. (1), it is closed and bounded, hence compact. Since $R(d; d^0, B)$ is lower-semicontinuous in d on $\mathcal{C}$, the objective in Eq. (4) attains its minimum on $\mathcal{C}$ by the Weierstrass theorem (Boyd & Vandenberghe, 2004). Therefore, for a fixed input allocation d^0 and backend B, there exists at least one minimizer $d^\star \in \mathcal{C}$. □

C Response normalization and downstream propagation

C.1 Proof of Proposition 2

This proposition records an algebraic scale-stability property of the featurewise relative-response proxy under the controlled rescaling model stated in Proposition 2. It should not be read as a proof that transformer blocks are homogeneous; the proxy is evaluated in the pruning experiments.

For feature h, the controlled rescaling model gives

$$\rho_{l,h}(\alpha) = \frac{\alpha^2 \|\Delta Y_{l,h}^B\|_2^2}{\alpha^2 \|Y_{l,h}\|_2^2 + \varepsilon} = \frac{\|\Delta Y_{l,h}^B\|_2^2}{\|Y_{l,h}\|_2^2 + \varepsilon/\alpha^2},$$

where $\alpha > 0$. Since $\|Y_{l,h}\|_2^2 + \varepsilon/\alpha^2 \geq \|Y_{l,h}\|_2^2$, we have

$$0 \leq \rho_{l,h}(\alpha) \leq \frac{\|\Delta Y_{l,h}^B\|_2^2}{\|Y_{l,h}\|_2^2}.$$

Moreover,

$$\left| \rho_{l,h}(\alpha) - \frac{\|\Delta Y_{l,h}^B\|_2^2}{\|Y_{l,h}\|_2^2} \right| = \frac{\|\Delta Y_{l,h}^B\|_2^2}{\|Y_{l,h}\|_2^2} \frac{\varepsilon}{\alpha^2 \|Y_{l,h}\|_2^2 + \varepsilon}.$$

Averaging this inequality over $h = 1, \ldots, m_l$ yields the statement for $\rho_l(\alpha) = m_l^{-1} \sum_h \rho_{l,h}(\alpha)$ and $U_l = m_l^{-1} \sum_h \|\Delta Y_{l,h}^B\|_2^2 / \|Y_{l,h}\|_2^2$. If the featurewise relative errors are bounded by C_l, then $U_l \leq C_l$. Thus the relative response remains controlled by the average featurewise relative perturbation, and its scale-dependent deviation vanishes when the output energy is not dominated by ε. □

C.2 Local-linear explanation of the RMS-transfer proxy

This subsection explains the RMS-transfer proxy used in Eq. (9). Signal-propagation and dynamical-isometry analyses study how signals and perturbations propagate through products of block-level Jacobians, often through averages over random perturbation directions (Poole et al., 2016; Schoenholz et al., 2017; Pennington et al., 2017). We use the same random-direction calculation to motivate a calibration proxy; the proxy itself is not an exact equality for pruning loss.

Consider block j and a small input perturbation δx_j. Under a local-linear approximation along the calibration trajectory,

$$\delta y_j \approx J_j \delta x_j,$$

where J_j is the block Jacobian. Assume, as a local proxy, that the perturbation direction is isotropic with

$$\mathbb{E}[\delta x_j \delta x_j^\top] = \sigma^2 I_{p_j},$$

where p_j is the input dimension. Then the standard random-direction calculation gives

$$\mathbb{E}\|J_j \delta x_j\|_2^2 = \operatorname{tr}\big(J_j^\top J_j \, \mathbb{E}[\delta x_j \delta x_j^\top]\big) = \sigma^2 \|J_j\|_F^2,$$
$$\mathbb{E}\|\delta x_j\|_2^2 = \operatorname{tr}\big(\mathbb{E}[\delta x_j \delta x_j^\top]\big) = \sigma^2 p_j.$$

Therefore,

$$\sqrt{\frac{\mathbb{E}\|J_j \delta x_j\|_2^2}{\mathbb{E}\|\delta x_j\|_2^2}} = \frac{\|J_j\|_F}{\sqrt{p_j}} = \sqrt{\frac{1}{p_j} \sum_{i=1}^{p_j} s_i(J_j)^2},$$

where the singular values are zero-padded to length p_j if necessary. This defines the unobserved RMS Jacobian perturbation gain G_j^{Jac}.

Estimating G_j^{Jac} would require Jacobian-vector statistics for every block. The main text therefore replaces it with the observable activation ratio $G_j^{\mathrm{obs}} := g_j$, computed from a calibration forward pass. This replacement is a proxy design choice: it preserves the intended layerwise transfer trend at low cost, while the end-to-end pruning results test its empirical usefulness. □

C.3 Implementation details

All main experiments use 70% unstructured sparsity unless otherwise stated. Calibration follows OWL/DLP with 128 C4 samples of sequence length 2048. Sparsity optimization estimates local responses and downstream activation gains using 32 calibration samples. For the pullback weight in Eq. (12), we set $\lambda = 1.0$ for OWL and DLP; for AlphaPruning, we use $\lambda = 1.0$ on LLaMA-1-7B and $\lambda = 0.5$ on the other models. We use the general suffix product $a_l = \prod_{j>l} \Gamma(g_j)$ to avoid tying the framework to one particular transfer map. Our implementation instantiates this map with

a damped residual-style form. Let d^0 be the input allocation. For each layer, we fix d_l^0, apply the same backend used in the final pruning run, and measure the resulting output discrepancy:

$$\widehat{W}_l(d_l^0) = \mathcal{P}_{\text{backend}}(W_l, X_l, d_l^0), \qquad e_l = \mathcal{E}_l(\widehat{W}_l(d_l^0); W_l, X_l),$$

where $\mathcal{P}_{\text{backend}}$ is WANDA's activation-weighted mask operator or SparseGPT's compensation-based pruning operator. In both cases, dense layer outputs are cached, the backend-pruned layer is evaluated on the same calibration inputs, and $\mathcal{E}_l$ is the normalized dense-vs-pruned reconstruction loss in Eq. (7). These responses are then combined with propagation weights.

The propagation term is estimated from one calibration forward pass. It follows the same mean-field transfer intuition as the main text: replace the full downstream Jacobian product by a product of scalar block-level transfer proxies. For layer j, let x_j and y_j be its input and output activations and define

$$g_j = \mathbb{E}_X\left[\frac{\text{rms}(y_j)}{\text{rms}(x_j) + \varepsilon}\right].$$

The quantity g_j is not an exact operator norm; it is a calibration-time RMS gain that approximates whether block j tends to expand or contract activation energy. In experiments, we use

$$\Gamma(g_j) = 1 + c\, N_g(g_j), \qquad a_l = \prod_{j>l} \Gamma(g_j), \qquad r_l = e_l a_l.$$

The function N_g is a nonnegative normalization of empirical gains across layers, and $c \geq 0$ damps the propagation strength. The additive baseline 1 reflects the residual identity path, while $cN_g(g_j)$ provides a controlled update-dependent correction. Thus e_l measures backend-specific local response, a_l measures downstream propagation, and r_l is the coefficient used in the sparsity-optimization objective. Other nonnegative choices of Γ can be substituted without changing the constrained optimization map.

C.4 Geometry ablation on uniform allocation

The constrained refinement can use the local response alone ($a_l = 1$); the propagation factor in Eq. (11) is optional. To assess it, we sweep $c \in \{0, 5\times10^{-4}, 10^{-3}, 2\times10^{-3}\}$ in $\Gamma(g_j) = 1+cN_g(g_j)$ from a uniform 70% allocation. Here $c = 0$ is local-only and $c > 0$ adds propagation adjustment. All variants use the same L2 regularization as the main uniform-input runs ($\lambda = 1.0$, except $\lambda = 0.5$ for LLaMA-2-13B). Table A1 compares local-only refinement with the best nonzero propagation setting on WikiText-2 PPL; lower is better, and negative changes indicate improvement. Table A1 reports a separate ablation run.

Table A1: Uniform-input propagation ablation at 70% sparsity.

Model	Backend	Local only	Best nonzero prop.	Change
LLaMA-2-7B	WANDA	26.69	**26.41**	-0.28
LLaMA-2-7B	SparseGPT	20.80	**20.75**	-0.05
LLaMA-2-13B	WANDA	14.77	**14.72**	-0.05
LLaMA-2-13B	SparseGPT	13.21	**13.12**	-0.10

D Proof of Theorem 1

Theorem 1 concerns the L2-regularized allocation problem in Eq. (12). Once the response coefficients r_l are fixed, the only remaining variable is the keep-ratio vector $d \in \mathcal{C}$. The solution first checks an interior closed-form candidate and uses one-dimensional bisection only when layerwise bounds are active.

Proof of Theorem 1. The proof only uses $r_l \geq 0$ and $\lambda > 0$; the estimator used to obtain r_l does not affect the allocation map. Eq. (12) minimizes

$$F(d) = \sum_{l=1}^{L} r_l (d_l - 1)^2 + \frac{\lambda}{2} \sum_{l=1}^{L} (d_l - d_l^0)^2$$

over the convex feasible set $\mathcal{C}$. Because $\lambda > 0$, F is strongly convex, so the minimizer is unique whenever $\mathcal{C}$ is nonempty.

First ignore the box constraints and impose only $\sum_l d_l = L\bar{d}$. With multiplier ν, the Lagrangian is

$$\mathcal{L}(d, \nu) = \sum_{l=1}^{L} r_l (d_l - 1)^2 + \frac{\lambda}{2} \sum_{l=1}^{L} (d_l - d_l^0)^2 + \nu \left(\sum_{l=1}^{L} d_l - L\bar{d} \right).$$

Stationarity gives

$$2r_l(d_l - 1) + \lambda(d_l - d_l^0) + \nu = 0.$$

Thus each layer has the affine multiplier map

$$d_l(\nu) = \frac{2r_l + \lambda d_l^0 - \nu}{2r_l + \lambda}.$$

Enforcing the budget gives

$$\sum_{l=1}^{L} \frac{2r_l + \lambda d_l^0 - \nu}{2r_l + \lambda} = L\bar{d},$$

so the interior multiplier is

$$\nu_0 = \frac{\sum_{l=1}^{L} (2r_l + \lambda d_l^0)/(2r_l + \lambda) - L\bar{d}}{\sum_{l=1}^{L} 1/(2r_l + \lambda)}.$$

If $\tilde{d}_l = d_l(\nu_0)$ satisfies all bounds, the KKT conditions certify it as the global optimum.

When some bounds are active, the KKT conditions give the clipped map

$$d_l(\nu) = \mathrm{clip}\left(\frac{2r_l + \lambda d_l^0 - \nu}{2r_l + \lambda}, \underline{d}_l, \overline{d}_l \right).$$

Each $d_l(\nu)$ is continuous and nonincreasing in ν, so $G(\nu) = \sum_l d_l(\nu)$ is also nonincreasing. Since $\mathcal{C}$ is nonempty, sufficiently small and large multipliers bracket the target budget $L\bar{d}$. Scalar bisection therefore finds a multiplier with $G(\nu) = L\bar{d}$ up to the desired tolerance; if G is flat because of clipping, any multiplier in the flat interval gives the same allocation. The resulting vector satisfies the budget and all layerwise bounds, and the KKT conditions certify it as the unique optimizer. □

E Sparsity-Ratio Ablation

We extend the LLaMA-2-7B evaluation to 50–85% unstructured sparsity with WANDA (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023). For each backend and target, we compare refinement from Uniform, OWL, and DLP inputs while preserving the same global sparsity budget. Figure 3 shows the relative WikiText-2 perplexity (PPL) reduction; Table A2 provides the corresponding PPL pairs.

Table A2: WikiText-2 PPL on LLaMA-2-7B across target sparsities. Each cell reports input PPL → refined PPL; lower is better.

Backend	Input	50%	60%	70%	80%	85%
WANDA	Uniform	7.00 → 6.92	10.91 → 9.47	76.45 → 26.45	2615.41 → 454.39	25786.94 → 5289.79
	OWL	6.93 → 6.92	9.53 → 9.33	35.94 → 24.41	758.54 → 635.47	8526.62 → 4922.10
	DLP	7.06 → 7.05	9.39 → 9.35	24.36 → 22.14	458.91 → 389.53	2251.37 → 1952.42
SparseGPT	Uniform	7.11 → 7.01	10.26 → 9.73	26.68 → 20.99	116.12 → 75.73	302.26 → 200.98
	OWL	7.01 → 7.01	9.48 → 9.34	21.85 → 19.37	83.76 → 75.38	190.28 → 178.43
	DLP	7.14 → 7.13	9.37 → 9.33	18.75 → 18.40	80.81 → 78.35	212.91 → 185.90

At 50% sparsity, refinement yields only marginal changes in WikiText-2 PPL across the three input allocations and both backends, with a relative reduction of at most approximately 1.4%. Improvements become more pronounced from 60% onward, especially for the Uniform input. This pattern is



Figure 3: Relative WikiText-2 PPL reduction on LLaMA-2-7B at 50, 60, 70, 80, and 85% target sparsity. The left and right panels use WANDA and SparseGPT, respectively. Each point is computed within a paired input/refined comparison as $(\mathrm{PPL}_{\mathrm{input}} - \mathrm{PPL}_{\mathrm{refined}})/\mathrm{PPL}_{\mathrm{input}} \times 100\%$; positive values indicate lower PPL after refinement.

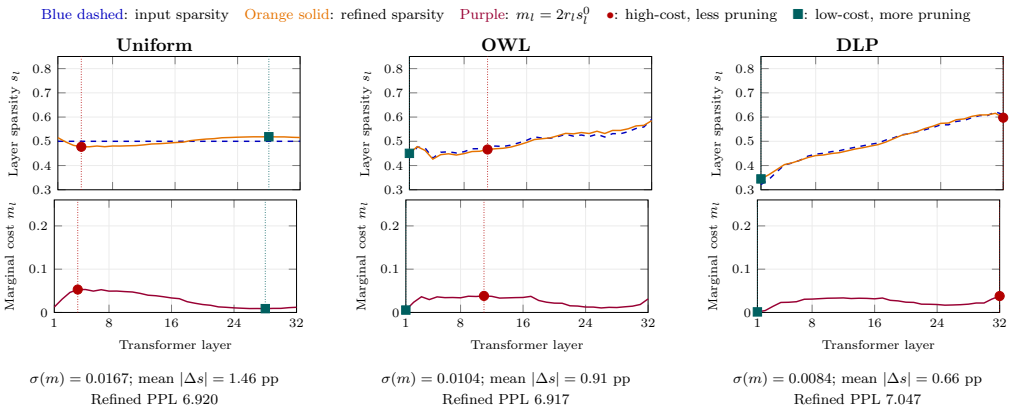

Figure 4: Local-only response and allocation diagnostic for LLaMA-2-7B/WANDA at 50% target sparsity, starting from Uniform, OWL, and DLP. The upper panels show input and refined layerwise sparsities; the lower panels show the initial marginal surrogate cost $m_l = 2r_l s_l^0$. This diagnostic uses $a_l = 1$, $\lambda = 1$.

consistent with layerwise budget allocation becoming increasingly consequential under aggressive pruning: reallocating retained weights can alleviate the burden on layers with higher measured pruning cost. At higher sparsity, OWL and DLP often provide lower-PPL starting allocations, leaving less room for further improvement, although the gains vary by backend and sparsity level. The trend is non-monotonic at 80–85%, and some refined models still exhibit high absolute PPL. Overall, this ablation identifies where refinement is most beneficial for LLaMA-2-7B; it does not establish a universal sparsity threshold, nor does it prove that the smaller gains at the lower end of this range stem from smaller response magnitudes.

For Uniform input, the 50% diagnostic has a smaller across-layer spread of marginal cost and smaller mean allocation change than the separate 70% WANDA diagnostic ($\sigma(m)$: 0.0167 versus 0.070; mean $|\Delta s|$: 1.46 versus 5.46 percentage points).

F Response-Signal Controls

We test whether the correspondence between layers and their measured responses matters for refinement. Within each row of Table A3, we keep the model, backend, input allocation, target sparsity,

Table A3: WikiText-2 PPL under response-signal interventions on LLaMA-2-7B; lower is better. Only the response supplied to the solver changes within each row.

Target	Backend	Input	Measured	Permuted	Constant
70%	WANDA	Uniform	**26.69**	71.66	77.52
		OWL	**24.21**	38.42	38.63
		DLP	**22.25**	26.91	26.68
	SparseGPT	Uniform	**20.80**	27.63	27.32
		OWL	**19.55**	22.79	22.78
		DLP	**18.22**	18.96	19.05
50%	WANDA	Uniform	6.92	7.02	7.00
		DLP	7.05	7.04	7.04
	SparseGPT	Uniform	7.06	7.08	7.08
		DLP	7.14	7.14	7.11

and constrained solver fixed. We supply either the measured response, the same values permuted across layers with one fixed seed (0), or a constant response equal to their layerwise mean. These are separate diagnostic runs, not the endpoints reported in the main paired table.

At 70%, the measured response gives lower PPL than both controls in all six tested configurations, supporting the value of its layerwise correspondence under the same solver. At 50%, the differences are much smaller: the measured response is slightly better than the constant control from Uniform, but not from DLP. Some 50% differences disappear after rounding to two decimals.

G Multiple-round Refinement Probe

The same constrained map can be applied repeatedly by using the previous optimized allocation as the next input allocation and recollecting responses. Table A4 reports a small LLaMA-2-7B WANDA probe at 70% sparsity, including the plain allocation before refinement. Additional rounds reduce PPL for both uniform and OWL inputs in this probe, while each extra round requires another response-collection pass, yet the improvement is decreasing as the solution is getting better.

Table A4: Multiple-round refinement probe on LLaMA-2-7B with WANDA at 70% sparsity.

Input allocation	Default	1st-round	2nd-round	3rd-round
Uniform	77.74	26.56	22.65	22.24
OWL	35.66	24.30	22.52	22.30

H Runtime Comparisons

We separate the cost of constructing an input allocation from the cost of our constrained refinement. Fixed allocation rules such as OWL, DLP, and AlphaPruning compute layer statistics, transform them into scores, and apply a closed-form or rule-based map to obtain an allocation. Their extra allocation-stage cost is therefore dominated by the statistic computation, together with $O(L)$ layer-level computation. When our framework is applied on top of such rules, the same input allocation can be reused as d^0; the added cost is one backend-response collection step plus the allocation solver.

The response collection step evaluates the chosen pruning backend at the probing allocation d^0 and compares dense and pruned layer outputs on calibration inputs. This requires no gradient back-propagation and no optimization over mask variables. After the response coefficients r_l are fixed, Algorithm 1 optimizes only the L-dimensional sparsity-ratio vector. Each bisection step evaluates the scalar map $d_l(\nu)$ for all layers and clips it to the layerwise bounds, costing $O(L)$ arithmetic operations and storing only layer-level scalars. In many cases the interior candidate is already feasible, so the closed-form allocation is returned without bisection. Our experiments used rented cloud GPU instances, so wall-clock times can vary slightly across instances even when the GPU configurations are nominally identical; such variation is negligible for our comparisons.

Tables A5 and A6 report paired pruning-only runtime at 70% sparsity: each plain allocation is compared with its Plug-and-Prune refinement under the same backend. These measurements exclude perplexity evaluation and model saving. Table A7 separately provides same-device pruning-stage times for DSA's released-rule application, target-matching layer-wise BESA, and Uniform-initialized Plug-and-Prune where matched timings are available. DSA's offline rule-search time is not included.

Table A5: Pruning-only runtime on LLaMA-family models with WANDA at 70% sparsity.

Model	Input allocation	Plain (s)	+ Plug-and-Prune (s)	Increase
LLaMA-1 7B	OWL	89.4	111.8	25.1%
LLaMA-1 7B	DLP	132.9	153.1	15.2%
LLaMA-1 7B	AlphaPruning	89.2	124.1	39.1%
LLaMA-2 7B	OWL	89.7	111.3	24.1%
LLaMA-2 7B	DLP	133.4	155.5	16.6%
LLaMA-2 7B	AlphaPruning	89.0	133.7	50.2%
LLaMA-1 13B	OWL	153.7	188.9	22.9%
LLaMA-1 13B	DLP	230.5	251.1	8.9%
LLaMA-1 13B	AlphaPruning	155.1	232.0	49.6%
LLaMA-2 13B	OWL	154.4	176.7	14.4%
LLaMA-2 13B	DLP	230.5	251.0	8.9%
LLaMA-2 13B	AlphaPruning	156.3	230.4	47.4%
LLaMA-1 30B	OWL	341.8	414.3	21.2%
LLaMA-1 30B	DLP	514.4	581.9	13.1%
LLaMA-1 30B	AlphaPruning	346.3	517.2	49.4%

Plain uses the listed allocation without response-aware refinement. Runtimes are pruning-only wall-clock seconds.

Table A6: Pruning-only runtime on LLaMA-family models with SparseGPT at 70% sparsity.

Model	Input allocation	Plain (s)	+ Plug-and-Prune (s)	Increase
LLaMA-1 7B	OWL	279.9	438.1	56.5%
LLaMA-1 7B	DLP	320.4	482.2	50.5%
LLaMA-1 7B	AlphaPruning	279.2	453.0	62.2%
LLaMA-2 7B	OWL	277.3	443.1	59.8%
LLaMA-2 7B	DLP	323.6	484.6	49.8%
LLaMA-2 7B	AlphaPruning	278.3	463.4	66.5%
LLaMA-1 13B	OWL	477.9	739.2	54.7%
LLaMA-1 13B	DLP	552.4	808.8	46.4%
LLaMA-1 13B	AlphaPruning	479.3	780.8	62.9%
LLaMA-2 13B	OWL	465.4	724.1	55.6%
LLaMA-2 13B	DLP	550.0	800.8	45.6%
LLaMA-2 13B	AlphaPruning	472.5	777.4	64.5%
LLaMA-1 30B	OWL	1032.4	1572.6	52.3%
LLaMA-1 30B	DLP	1211.1	1746.4	44.2%
LLaMA-1 30B	AlphaPruning	1034.2	1665.4	61.0%

Plain uses the listed allocation without response-aware refinement. Runtimes are pruning-only wall-clock seconds.

The relative increase depends on the cost of the input allocator and backend. For lightweight rules such as OWL and especially AlphaPruning, the plain pruning pipeline is close to a single score-to-allocation pass, so the fixed response-collection step accounts for a larger fraction of the total runtime. For DLP, the plain allocation stage is already more expensive, and the same response-collection cost is therefore amortized over a larger baseline. SparseGPT has a higher absolute pruning cost than WANDA because of its reconstruction-compensated pruning step; applying Plug-and-Prune on top of SparseGPT also probes this backend at the input allocation, leading to larger relative increases in Table A6.

Table A7 provides same-device runtime context for BESA, DSA, and Uniform-initialized PnP. On the four models with matched timing, WANDA-based PnP takes less pruning-stage time than DSA's released-rule application and BESA's layer-wise run. SparseGPT-based PnP has a different cost

Table A7: Pruning-stage time (seconds) at a target of 70% unstructured sparsity.

Model	PnP input	DSA	BESA	WANDA plain / PnP	SparseGPT plain / PnP
LLaMA-1 7B	Uniform	94.754	273.495	35.319 / 46.303	200.101 / 343.744
LLaMA-2 7B	Uniform	97.100	264.927	35.144 / 45.784	202.772 / 337.213
LLaMA-2 13B	Uniform	169.695	484.620	63.877 / 82.612	373.566 / 586.045
LLaMA-3 8B	Uniform	93.331	288.319	69.507 / 85.231	360.613 / 496.077

profile because response collection also invokes the more expensive backend. These timings do not include DSA's offline rule-search stage.

I Proof of Theorem 2

Let $\widehat{f}_d$ be the model obtained after pruning with allocation d, and let $\Delta(d) = \mathcal{L}(\widehat{f}_d) - \mathcal{L}(f)$ be the corresponding degradation, where $\mathcal{L}$ is the evaluation loss. Let $\Phi(d) = \sum_{l=1}^{L} r_l(d_l - 1)^2$ and

$$F(d) = \Phi(d) + \frac{\lambda}{2}\|d - d^0\|_2^2, \qquad d \in \mathcal{C}.$$

Since $d^\star$ minimizes $F(d)$ over $\mathcal{C}$ and $d^0 \in \mathcal{C}$ is feasible,

$$F(d^\star) \le F(d^0) = \Phi(d^0),$$

where the regularization term vanishes at $d = d^0$. Hence

$$\Phi(d^\star) - \Phi(d^0) \le -\frac{\lambda}{2}\|d^\star - d^0\|_2^2.$$

(i) Local non-degradation under alignment. If the local surrogate–degradation alignment condition holds $d = d^\star$,

$$\Delta(d) - \Delta(d^0) \le c\big(\Phi(d) - \Phi(d^0)\big) + \eta\|d - d^0\|_2^2,$$

then substituting the surrogate decrease above at $d = d^\star$ gives

$$\Delta(d^\star) - \Delta(d^0) \le -\Big(\frac{c\lambda}{2} - \eta\Big)\|d^\star - d^0\|_2^2.$$

(ii) Stability near aligned inputs. Let $\Phi^\star_{\mathcal{C}} = \min_{d\in\mathcal{C}} \Phi(d)$. Since $\Phi(d^\star) \ge \Phi^\star_{\mathcal{C}}$, the same surrogate decrease implies

$$\frac{\lambda}{2}\|d^\star - d^0\|_2^2 \le \Phi(d^0) - \Phi(d^\star) \le \Phi(d^0) - \Phi^\star_{\mathcal{C}}.$$

Rearranging gives

$$\|d^\star - d^0\|_2^2 \le \frac{2}{\lambda}\big(\Phi(d^0) - \Phi^\star_{\mathcal{C}}\big).$$

(iii) Feasible-direction improvement. Suppose there exist $\gamma > 0$ and $\tilde{d} \in \mathcal{C}$ such that

$$\Phi(\tilde{d}) + \frac{\lambda}{2}\|\tilde{d} - d^0\|_2^2 \le \Phi(d^0) - \gamma.$$

By optimality, $F(d^\star) \le F(\tilde{d}) \le \Phi(d^0) - \gamma$. Since the pullback term in $F(d^\star)$ is nonnegative,

$$\Phi(d^\star) \le F(d^\star) \le \Phi(d^0) - \gamma.$$

This proves all three claimed properties. □

J LLaMA-3-70B Scale Check

We investigate whether refinement from an OWL allocation remains beneficial on LLaMA-3-70B at 70% unstructured sparsity and report WikiText-2 perplexity (PPL) and average zero-shot accuracy. The results yield lower PPL and higher average zero-shot accuracy after refinement in this setting.

Table A8: LLaMA-3-70B results at 70% sparsity.

Backend	WikiText-2 PPL	Seven-task average accuracy
WANDA	$39.315 \rightarrow \mathbf{20.489}$	$46.22\% \rightarrow \mathbf{51.30}\%$ (+5.08 pp)
SparseGPT	$42.234 \rightarrow \mathbf{30.490}$	$45.62\% \rightarrow \mathbf{46.29}\%$ (+0.67 pp)

Table A9: WikiText-2 PPL on additional models at a target of 70% unstructured sparsity.

Inner-layer backend	Input allocation	WikiText-2 PPL: input → refined			
		LLaMA-3-8B	Qwen3-0.6B	Qwen3-4B	Phi-4
WANDA	Uniform	116.301 → 74.211	5270.364 → 556.735	104.269 → 63.080	53.438 → 28.893
	OWL	131.499 → 73.700	9117.661 → 611.899	61.290 → 59.923	29.222 → 28.457
	DLP	101.953 → 72.636	885.262 → 605.096	65.790 → 64.119	31.479 → 29.727
	AlphaPruning	180.072 → 113.799	151422.438 → 3439.481	303.394 → 84.130	236.232 → 34.588
SparseGPT	Uniform	45.793 → 37.345	417.808 → 203.975	41.140 → 40.264	23.087 → 19.555
	OWL	48.397 → 37.530	239.953 → 196.716	36.126 → 36.070	18.139 → 17.830
	DLP	40.807 → 36.770	223.546 → 207.047	54.527 → 45.570	17.984 → 17.913
	AlphaPruning	66.427 → 41.793	6995.849 → 312.006	101.082 → 46.639	35.770 → 21.951

AlphaPruning input and refined runs do not always achieve the same actual sparsity; those entries are diagnostic rather than strictly matched comparisons.

K Results on Additional Models

We further evaluate Plug-and-Prune on LLaMA-3-8B, Qwen3-0.6B/4B, and Phi-4 at a target of 70% unstructured sparsity. Table A9 reports all tested input allocations.

For method-level context, Table A10 compares WANDA-based PnP from a uniform input with DSA and BESA. The PnP values are the Uniform/WANDA entries in Table A9.

Table A10: WikiText-2 PPL comparison with DSA and BESA on additional models at a target of 70% unstructured sparsity; lower is better. PnP starts from a uniform allocation.

Metric	Method	LLaMA-3-8B	Qwen3-0.6B	Qwen3-4B	Phi-4
WikiText-2 PPL ↓	PnP (WANDA, Uniform)	74.211	556.735	63.080	28.893
	DSA	115.392	8532.827	64.823	30.629
	BESA	291.867	1895.327	121.969	58.271

PnP obtains the lowest PPL among these three methods on the four additional models. DSA applies its released allocation rule rather than performing a new rule search for each model. BESA uses a target-matching layer-wise configuration and does not attain exactly 70% sparsity in every case; it also optimizes masks as well as sparsity.